Subconsensus Tasks: Renaming is Weaker than Set Agreement

Eli Gafni[1], Sergio Rajsbaum[2], and Maurice Herlihy[3]

[1] Computer Science Department UCLA Los Angles, CA 90095 eli@cs.ucla.edu
[2] Instituto de Matemáticas, Universidad Nacional Autónoma de México, Ciudad Universitaria, D.F. 04510, Mexico rajsbaum@math.unam.mx
[3] Brown University, Computer Science Department, Providence, RI 02912; mph@cs.brown.edu

Abstract. We consider the the relative power of two important synchronization problems: set agreement and renaming. We show that renaming is strictly weaker than set agreement, in a round-by-round model of computation. We introduce new techniques, including previously unknown connections between properties of manifolds and computation, as well as novel "symmetry-breaking" constructions.

1 Introduction

A *task* in an asynchronous system is a problem where each process starts with a private input value, communicates with the others, and halts with a private output value. A *protocol* is a program that solves the task. In asynchronous systems, it is desirable to design protocols that are *wait-free*: any process that continues to run will halt with an output value in a fixed number of steps, regardless of delays or failures by other processes.

We are interested in the *relative power* of tasks: given two tasks, can one be used to implement the other, or are they incomparable? One way to measure the relative power of tasks is by *consensus numbers* [11]. If a task can solve consensus [7] for n processes is *universal* in a system of n processes in the sense that it can be used to construct a protocol that solves any other task. Moreover, any task that solves consensus for n processes also solves consensus for fewer.

One question that has received substantial attention, and yet is still far from understood, is the relative power of tasks too weak to solve consensus for two processes, the so-called "subconsensus" tasks. For example, it is known that read/write memory cannot solve consensus. After a substantial effort, it was discovered that two other tasks, set agreement [6] and renaming [1], which are both subconsensus tasks, cannot be implemented in read/write memory [3, 10, 13]. It follows that subconsensus tasks have a "fine structure", inaccessible by consensus-based analysis.

In this paper, we shed further light on this fine structure by showing that the renaming task is strictly weaker than the set agreement task, in a natural model of asynchronous computation. a surprising result, since the two tasks are superficially quite dissimilar. We introduce new techniques that may be helpful for other problems: a previously unknown connection between properties of manifolds and computation, as well as novel "symmetry-breaking" constructions.

2 Model

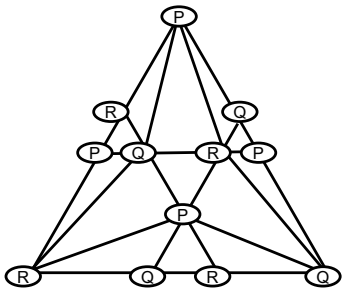

Fig. 1. Standard Chromatic Subdivision

2.1 Combinatorial Topology

We start by reviewing standard terminology from combinatorial topology (mostly taken from Munkres [12]). We employ concepts from combinatorial topology, in a style similar to Attiya and Rajsbaum [2].

A *simplicial complex* $\mathcal{A}$ is a collection of finite non-empty sets, such that if A is an element of $\mathcal{A}$, so is every non-empty subset of A. An element A of $\mathcal{A}$ is called a *simplex*, and an element of A is called a *vertex*. The dimension of A is one less than its number of elements. A subcollection $\mathcal{B}$ of $\mathcal{A}$ is called a *subcomplex* of $\mathcal{A}$ if $\mathcal{B}$ is itself a complex. A subset B of simplex A is itself a simplex called a *face* of A. We refer to an n-dimensional simplex as an n-simplex, and we sometimes use superscripts (for example A^n) to indicate a simplex's dimension. The *dimension* of a complex is the largest dimension of any of its simplexes. An n-dimensional complex $\mathcal{A}$ is *full to dimension* n if every simplex in $\mathcal{A}$ is a face of an n-simplex. All complexes considered here are full to some dimension.

Let $\mathcal{A}$ and $\mathcal{B}$ be complexes. A *simplicial map* $\mathcal{M} : \mathcal{A} \rightarrow \mathcal{B}$ carries vertexes of $\mathcal{A}$ to vertexes of $\mathcal{B}$ so that every simplex of $\mathcal{A}$ maps to a simplex of $\mathcal{B}$. An n-complex is *colored* if there exists a map id sending each vertex to $\{0, \ldots, n\}$ such that each n-simplex is labeled with $n+1$ distinct colors. If A is a colored simplex, $ids(A)$ denotes its set of colors. If $\mathcal{A}$ and $\mathcal{B}$ are colored complexes, then a simplicial map $\mathcal{M} : \mathcal{A} \rightarrow \mathcal{B}$ is *color-preserving* if $id(v) = id(\mathcal{M}(v))$ for every vertex v in $\mathcal{A}$.

For a simplex S, $\chi(S)$, the *standard chromatic subdivision*, is defined as follows. Each vertex of $\chi(S)$ is a pair (s, F), where F is a face of S, and s a vertex of F. A set of vertexes of $\chi(S)$ define a simplex if for each pair (s, F) and (s', F'), $id(s)$ and $id(s')$ are distinct, and either $F \subseteq F'$, or $F' \subseteq F$. As s ranges over the vertexes of S, the vertexes (s, S) define the *central simplex* of $\chi(S)$. (The standard chromatic subdivision is the colored analog of the *standard barycentric subdivision* [12, p.96].)

A complex is *strongly connected* if any two n-simplexes can be joined by a "chain" of n-simplexes in which each pair of neighboring simplexes has a common $(n-1)$-dimensional face. A complex $\mathcal{M}$ is a *manifold with boundary* (sometimes called a *pseudomanifold with boundary*) if it is strongly connected, and each $(n-1)$-simplex in $\mathcal{M}$ is contained in precisely one or two n-simplexes. An $(n-1)$ simplex in $\mathcal{M}$ is an *interior* simplex if it is contained in two n-simplexes, and a *boundary* simplex otherwise. The *boundary* subcomplex of $\mathcal{M}$, denoted $\partial\mathcal{M}$, is the subcomplex induced by its boundary simplexes. For brevity, we will use "manifold" as an abbreviation for "manifold with boundary".

2.2 Distributed Computation

We are given a set of $n+1$ process ids $\{0,\ldots,n\}$. An initial or final state of a process is modeled as a vertex $v=(P,w)$, a pair consisting of a process id P and a value w. We say the vertex is *colored* with the process id. A set of $n+1$ mutually compatible initial or final states is modeled as an n-simplex $A^n=(a_0,\ldots,a_n)$.

A *task specification* is given by a colored *input complex* $\mathcal{I}$, a colored *output complex* $\mathcal{O}$, and a recursive relation Δ carrying each m-simplex of $\mathcal{I}$ to a set of m-simplexes of $\mathcal{O}$, for each $0 \leq m \leq n$. Δ has the following interpretation: if the $(m+1)$ processes named in S^m start with the designated input values, and the remaining $n-m$ processes fail without taking any steps, then each simplex in $\Delta(S^m)$ corresponds to a legal final state of the non-faulty processes. A *protocol* is a program in which processes receive their inputs, communicate via shared objects, and eventually return with mutually compatible decision values. Any protocol has an associated *protocol complex* $\mathcal{P}$, in which each vertex is labeled with a process id and that process's final state (called its *view*). Each simplex thus corresponds to an equivalence class of executions that "look the same" to the processes at its vertexes. The protocol complex corresponding to executions starting from a fixed simplex S^m is denoted $\mathcal{P}(S^m)$. We typically use $\mathcal{P}$ to denote both a protocol and its complex. A protocol *solves* a task if there exists a color-preserving simplicial *decision map* $\delta : \mathcal{P} \rightarrow \mathcal{O}$ such that for each simplex $R^m \in \mathcal{P}(S^m)$, $\delta(R^m) \in \Delta(S^m)$. Note that a protocol can also be considered as a task.

3 Tasks

In the *immediate snapshot* protocol [3], the processes share an array of registers, one for each process, which for brevity we refer to as the *immediate snapshot memory*. This memory provides a single operation: *write-read*, which writes the process state to its register and takes an instantaneous snapshot of the others. Logically, the processes act as if the round is divided into *phases*, where in each phase a set of processes is chosen, disjoint from any set chosen in an earlier phase. The processes in the set simultaneously write to their own registers, and then read all the others. The protocol complex for the one-round immediate snapshot is just the standard chromatic subdivision. An algorithm for implementing immediate snapshot from asynchronous read/write registers is given by Borowsky and Gafni [3, 5].

In the k-*Set Agreement* task [6], each process starts with a private input value, communicates with the others, and then halts after choosing a private output value. Each process is required to choose some process's input, and at most k distinct values may be chosen. For brevity we use *set agreement* as shorthand for $(n+1)$-process n-Set Agreement, since this is the easiest set agreement task known to be impossible for $n+1$ processes using shared read/write memory [3, 9, 10, 13, 14].

Set agreement has a particularly simple interpretation in the simplicial model. A protocol $\mathcal{P}$, starting from an input n-simplex S^n, solves set agreement if and only if there exists a (necessarily not color-preserving) simplicial map

$$\delta : \mathcal{P}(S^n) \to S^n$$

that (1) carries each $\mathcal{P}(S^m)$ to S^m, for dimensions m in $0, \ldots, n$, and (2) does not map any n-simplex of $\mathcal{P}(S^n)$ *onto* S^n.

The *Renaming* task [1] can be formulated in several equivalent ways. Processes are issued unique input names from a large name space, and must choose unique output names taken from a smaller name space. To rule out trivial solutions, the protocol must be *anonymous* [10]): the value chosen must depend on the protocol execution, not on the specific process id. Formally, a protocol complex $\mathcal{P}(S^n)$ is *symmetric* if any permutation π of the process ids induces a simplicial map $\pi : \mathcal{P}(S^n) \to \mathcal{P}(S^n)$. All complexes considered in this paper are symmetric. Recall that the protocol has a decision function $\delta : \mathcal{P}(S^n) \to \mathcal{O}$, where $\mathcal{O}$ is the output complex. The protocol is *anonymous* if

$$\pi(\delta(\mathcal{P}(S^n))) = \delta(\pi(\mathcal{P}(S^n)))$$

that is, relabeling process ids does not change the protocol's decisions.

In the *Weak Symmetry-Breaking* (WSB) task, tasks have no input values and Boolean output values. In every execution in which all $n+1$ processes participate, at least one process decides *true* and at least one process decides *false*. Like renaming, any protocol that implements WSB must be anonymous.

Definition 1. *A task $\mathcal{M}$ is a* manifold task *if for every input m-simplex S, for m in $0, \ldots, n$, $\mathcal{M}(S)$ is a manifold of dimension m and $\mathcal{M}(\partial S) = \partial\mathcal{M}(S)$.*

The immediate snapshot task is an example of a manifold task. A manifold task is a special case of a *divided image* [2].

4 Round-by-Round Computations

We study tasks in the *round-by-round model* [8], where processes execute in a sequence of asynchronous rounds. Each round is associated with an object (which could be an immediate snapshot memory, or another object). In each round, each process applies one operation to that round's object. Once a process has finished a round, it never revisits that round's objects. The result of one round are carried to the next not though objects, but through processor-local state.

It is known that for read/write memory alone, the round-by-round model is equivalent to the usual model in which processes can access any object at any time [4, 5]. We

conjecture that the two models are equivalent for tasks as well, but the question remains open.

The principal attraction of the round-by-round model is that manifold tasks are closed under composition: given a set of tasks whose one-round protocol complexes are manifolds, the result of their multi-round composition is also a manifold. For example, consider the effect of iterating the immediate snapshot protocol. Running it once yields the standard chromatic subdivision, and running it multiple times yields an *iterated subdivision*, where each simplex R in $\chi^{r-1}(I)$ is replaced with a copy of $\chi(R)$, yielding the subdivision $\chi^r(I)$.

Although the protocol complex for the multi-round snapshot complex is a subdivision of the input simplex, this property is not true of multi-round manifold tasks in general. For example, the Moebius task introduced below has a boundary complex equivalent to the boundary of a subdivided simplex, but the complex interior has a "hole" (that is, non-trivial homology), so the multi-round composition of this task cannot be a subdivided simplex. Nevertheless, a multi-round composition of a manifold task is manifold task.

Lemma 1. *If single-round protocol $\mathcal{R}$ is a manifold task, and $\mathcal{A}$ is a manifold, then $\mathcal{R}(\mathcal{A})$ is a manifold, and $\mathcal{R}(\partial\mathcal{A}) = \partial\mathcal{R}(\mathcal{A})$.*

Proof. Let R^{n-1} be an $(n-1)$-simplex in $\mathcal{R}(\mathcal{A})$. There are several cases to consider. Suppose R^{n-1} is an interior simplex of $\mathcal{R}(A^n)$ for some n-simplex $A^n \in \mathcal{A}$. Because $\mathcal{R}(A^n)$ is a manifold with boundary, there exist exactly two n-simplexes R_0^n and R_1^n in $\mathcal{R}(A^n)$ such that $R^{n-1} = R_0^n \cap R_1^n$.

Suppose R^{n-1} is a simplex of $\mathcal{R}(A^{n-1})$, for some interior simplex $A^{n-1} \in \mathcal{A}$. There are exactly two n-simplexes A_0^n and A_1^n in A^n such that $A^{n-1} = A_0^n \cap A_1^n$. Because R^{n-1} is a boundary simplex of $\mathcal{R}(A_0^n)$, there exists exactly one n-simplex R_0^n in $\rho(A_0^n)$ such that $R^{n-1} \subset R_0^n$. Similarly, because R^{n-1} is a boundary simplex of $\rho(A_1^n)$, there exists exactly one n-simplex R_1^n in $\rho(A_1^n)$ such that $R^{n-1} \subset R_1^n$. It follows that there exist exactly two n-simplexes R_0^n and R_1^n in $\rho(A^n)$ such that $R^{n-1} = R_0^n \cap R_1^n$.

Suppose R^{n-1} is a simplex of $\mathcal{R}(A^{n-1})$, for some boundary simplex $A^{n-1} \in \mathcal{A}$. Because A^{n-1} is a boundary simplex, there is exactly one n-simplex A^n in $\mathcal{A}$ such that $A^{n-1} \subset A^n$. Because R^{n-1} is a boundary simplex of $\mathcal{R}(A^n)$, there exists exactly one n-simplex R^n in $\mathcal{R}(A^n)$ such that $R^{n-1} \subset R^n$. It follows that R^{n-1} is a boundary simplex of $\mathcal{R}(\mathcal{A})$. The last item implies that $\mathcal{R}(\partial\mathcal{A}) \subseteq \partial\mathcal{R}(\mathcal{A})$.

If $R^{n-1} \in \partial\mathcal{R}(\mathcal{A})$, then $R^{n-1} \in \partial\mathcal{R}(A^n)$, for some A^n in $\mathcal{A}$. Because $\mathcal{R}$ is a manifold task, $\partial\mathcal{R}(A^n) = \mathcal{R}(\partial A^n)$ R^{n-1} is also in $\mathcal{R}(\partial A^n)$, and therefore $\partial\mathcal{R}(\mathcal{A}) \subseteq \mathcal{R}(\partial\mathcal{A})$.

It remains to check $\mathcal{R}(\mathcal{A})$ is strongly connected. Pick two n-simplexes R_0 and R_1 of $\mathcal{R}(\mathcal{A})$. Let $R_0 \in \mathcal{R}(A_0)$ and $R_1 \in \mathcal{R}(A_1)$. Because $\mathcal{A}$ is a manifold, it is strongly connected, and there exists a chain of n-simplexes $B_0, \ldots, B_k$ such that $A_0 = B_0$, $A_1 = B_k$, and each B_i and B_{i+1} share an $(n-1)$-dimensional face. We argue by induction on k. When k is zero, the result follows because $\mathcal{R}(A_0) = \mathcal{R}(A_1)$ is a manifold.

Assume the result for chains of length $k-1$. Let S be an $(n-1)$-simplex on the common boundary of $\mathcal{R}(A_{k-1})$ and $\mathcal{R}(A_k)$, and let S_0 (S_1) be the unique n-simplex

in $\mathcal{R}(A_{k-1})$ ($\mathcal{R}(A_k)$) having S as a face. By the induction hypothesis, there exists a chain from R_0 to S_0, and from S_1 to R_1. Since S_0 and S_1 share a common $(n-1)$-dimensional face, we have constructed a chain from R_0 to R_1. □

Corollary 1. *The result of composing manifold tasks is a manifold task.*

5 Structural Results

5.1 Set Agreement

Theorem 1. *No manifold task can solve set agreement.*

Proof. Let $\mathcal{M}$ be a manifold task, and let S^n be an input n-simplex. A *Sperner labeling* is defined as follows[4]. Define a map carrying each vertex of $\mathcal{M}(S^n)$ to a vertex of S^n. For each face S^m of S^n, where the dimension m ranges from 0 to n, map each vertex of $\mathcal{M}(S^m)$ with one of the vertexes in S^m. Sperner's Lemma states that there must exist at least one n-simplex in $\mathcal{M}(S^n)$ that maps *onto* S^n (that is, maps to $n+1$ distinct vertexes).

Now suppose we can solve set agreement by some repeated composition of $\mathcal{M}$ and one-round immediate-snapshot protocols. By Corollary 1, the resulting complex is itself a manifold. The decision map δ induces a Sperner labeling: for each face S^m of S^n, for m in $0, \ldots, n$, δ carries each vertex of $\mathcal{M}(S^m)$ to a vertex of S^m. It follows that there is some n-simplex of $\mathcal{M}(S^n)$ that maps on to every vertex of S^n, contradicting the definition of the set agreement task. □

5.2 Renaming vs Weak Symmetry-Breaking

Lemma 2. *Renaming implements Weak Symmetry-Breaking.*

Proof. The $n+1$ processes call a Renaming protocol to choose unique names in the range $0, \ldots, 2n-1$. Each process then chooses *true* if its chosen name is even, and *false* if odd. At least process must choose *true* and at least one *false*. This protocol is anonymous. □

Lemma 3. *Weak Symmetry-Breaking implements Renaming.*

Proof. Attiya et al. [1] give a renaming algorithm for $n+1$ processes and $2n+1$ names with the property that if $k \leq n+1$ processes actually participate, then they are assigned names at most $2k-1$ names. Consider the algorithm illustrated in Figure 2. Create two instances of this renaming algorithm: R_0 and R_1. The processes first execute a weak symmetry-breaking task. The $k > 0$ processes that decide *true* each take a name from R_0, choosing at most $2k-1$ names in the range $0, \ldots, 2k-2$. The other $n-k+1$ processes that decide *false* each take an intermediate name from R_1, choosing at most $2n-2k+1$ names in the range $0, \ldots, 2n-2k$. Each process in the second group

[4] Other authors call this construct a *Sperner coloring*, but we have already used "coloring" to mean labeling with process ids.

```
// data fields
WSB wsb = new WSB();
Renaming R0 = new Renaming();  // renaming task instance
Renaming R1 = new Renaming();  // renaming task instance
// algorithm
int chooseName(int n) {        // there are n+1 processes
  boolean side = wsb.choose();
  int name;
  if (side) {
    return R0.choose();
  } else {
    return (2*n) - R1.choose();
  }
}
```

Fig. 2. Implementing Renaming from WSB

chooses a name by subtracting its intermediate name from $2n$, yielding a name in the range $2n-2k-1,\ldots,2n-1$. Since these ranges do not overlap, this algorithm solves renaming for $n+1$ processes and $2n$ names. □

Corollary 2. *Renaming is equivalent to Weak Symmetry-Breaking.*

5.3 Set Agreement vs Weak Symmetry-Breaking

We now show set agreement solves weak symmetry-breaking. Assume we have a protocol that solves set agreement. We also need an instance of the Attiya et al. [1] protocol that assigns to each of $n+1$ processes a unique name in the range $0,\ldots,2n+1$. This protocol uses only read/write memory, and is anonymous. Consider the protocol shown in Figure 3.

The processes first call the Attiya protocol to choose ids in the range $0,\ldots,2n-1$. This step ensures that the protocol as a whole is anonymous. The processes that choose names in the range $0,\ldots,n$ call the first set agreement object's decide method, using its own anonymous id as input. The process then reads through the array, returning *true* if it finds its own id, and *false* otherwise. The processes that choose names in the range $n+1,\ldots,2n-2$ do the same using the other set agreement object and the other output array, except that they return inverted values.

Lemma 4. *Some process decides* true.

Proof. At least one process goes to the first set agreement object. Of these, the process whose id is first to be written to the output array decides *true*. □

Lemma 5. *Some process decides* false.

Proof. There are two cases to consider: (1) all processes go to the first set-agreement, or (2) some go to the first and some to the second. In the first case, if all processes

```
int[2][n] output; // two (n+1)-element arrays, initially -1
ABDPR rename;     // ABDPR renaming algorithm
SetAgree sa[2];   // set agreement protocol objects
boolean choose(int me) {
  int id = rename.choose(me); // anonymous id
  if (id < n+1) {
    output[0][id] = sa[0].decide(id);
    foreach (int i in output[0]) {
      if (i == id) return true;
    }
    return false;
  } else {
    output[1][id] = sa[1].decide(id);
    foreach (int i in output[1]) {
      if (i == id) return false;
    }
    return true;
}
```

Fig. 3. Implementing WSB from set agreement

decide *true*, then all $n+1$ inputs were chosen as decision values, violating the Set Agreement specification. In the second case, the processes that go to the second set-agreement object, the process whose id is first to be written decides *false*. □

Corollary 3. *Set Agreement solves Weak Symmetry-Breaking.*

This result shows that any task that solves Set Agreement also solves renaming, ruling out the possibility that the two tasks are incomparable.

We next introduce a new task, called the *Moebius* task, because in two dimensions its one-round protocol complex is a Moebius strip. (Of course, in higher dimensions, the complex is not a Moebius strip, although it is still a non-orientable manifold with boundary.)

This task is defined in even dimensions, so let $n = 2N$. Consider a system of processes, $P_0, \ldots, P_{2N}$. Let $S_0, \ldots, S_{2N}$ be $2N+1$ disjoint $(2N)$-simplexes, and let S_{ij} be the face of S_i opposite vertex P_j.

Let $X_i = \chi(S_i)$, the standard chromatic subdivision of S_i, and let $X_{ij} = \chi(X_{ij})$.

We call X_{ii} the *external face* of X_i (even though it is technically a complex), and the X_{ij}, for $i \neq j$, the *internal faces*.

Let U_j be the set of process ids in X_{ij}: 0 to $2N+1$ except for j. Let $\pi_j : U_j \to U_j$ send the index with rank r in U_j to the index with rank $r + N \bmod 2N$.

Identify each internal face X_{ij} with $X_{\pi_j(i)j}$. Because $\pi_j(\pi_j(i)) = i$, each $(2N-1)$-simplex in an external face lies in exactly two $(2N)$-simplexes, so the result is a manifold. (This also why the construction only works in even dimensions.)

The Moebius task itself is defined as follows. Let S^n be the input n-simplex. The Moebius complex $\mathcal{M}(S^n)$ is defined so that its boundary complex $\partial\mathcal{M}(S^n)$ is a subdivision of the boundary complex ∂S^n of the input simplex. For each proper face S^m

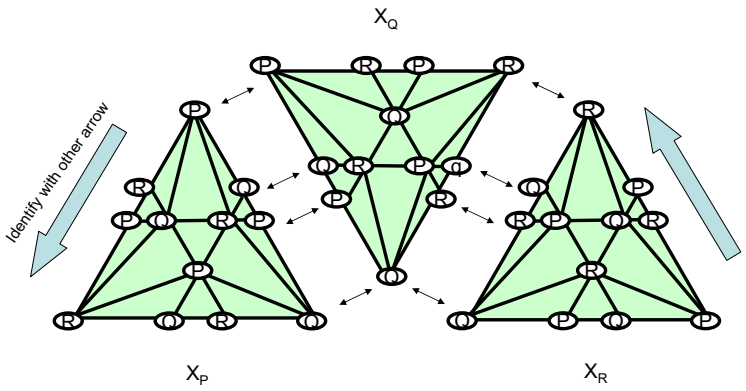

Fig. 4. One-round Moebius task protocol complex for 3 processes)

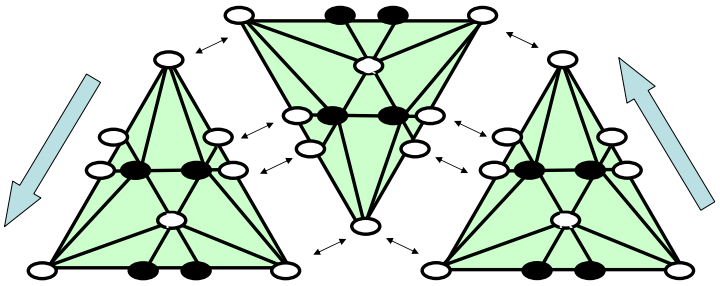

Fig. 5. Weak Symmetry-breaking from one-round Moebius task protocol

of S^n, for $m < n$, the task specification map Δ carries S^m to the m-simplexes in the unique face of $\partial\mathcal{M}(S^n)$ with the same set of process ids. Finally, Δ maps S^n to all n-simplexes of $\mathcal{M}(S^n)$.

Theorem 2. *The Moebius task cannot solve Set Agreement.*

Proof. The one-round Moebius task is a manifold task, so composing the Moebius task with itself, with read/write rounds, or with any other manifold task yields a manifold task (Corollary 1). So by Corollary 1, the Moebius task cannot solve Set Agreement. □

To show this task solves weak symmetry breaking, we color the edges with black and white pebbles (that is, *true* or *false* values) so that no simplex is monochrome. For the central simplex of each X_i, color each node black except for the one labeled with P_i. For the central simplex of each external face X_{ii}, color the central $(2N-2)$-simplex black. The rest are white.

Every $(2N-1)$-simplex X in X_i intersects both a face, either internal or external, and a central $(2N-1)$-simplex. If X intersects an internal face, then the vertexes on that face are white, and the vertexes on the central simplex are black. If X intersects the internal face, then it intersects the white node of the central simplex of X_i, and a black node of the central simplex of X_{ii}.

Theorem 3. *Set agreement implements renaming but not vice-versa.*

Proof. Set agreement solves weak symmetry-breaking (Corollary 3) which solves renaming (Corollary 2).

On the other hand, if renaming solves set agreement, then so does WSB, and so does the Moebius task, contradicting Theorem 1. □

References

1. Hagit Attiya, Amotz Bar-Noy, Danny Dolev, David Peleg, and Rüdiger Reischuk. Renaming in an asynchronous environment. *J. ACM*, 37(3):524–548, 1990.
2. Hagit Attiya and Sergio Rajsbaum. The combinatorial structure of wait-free solvable tasks. *SIAM J. Comput.*, 31(4):1286–1313, 2002.
3. E. Borowsky and E. Gafni. Generalized FLP impossibility result for t-resilient asynchronous computations. In *Proceedings of the 1993 ACM Symposium on Theory of Computing*, pages 206–215, May 1993.
4. Elizabeth Borowsky and Eli Gafni. Immediate atomic snapshots and fast renaming. In *PODC '93: Proceedings of the twelfth annual ACM symposium on Principles of distributed computing*, pages 41–51, New York, NY, USA, 1993. ACM Press.
5. Elizabeth Borowsky and Eli Gafni. A simple algorithmically reasoned characterization of wait-free computation (extended abstract). In *PODC '97: Proceedings of the sixteenth annual ACM symposium on Principles of distributed computing*, pages 189–198, New York, NY, USA, 1997. ACM Press.
6. S. Chaudhuri. Agreement is harder than consensus: Set consensus problems in totally asynchronous systems. In *Proceedings Of The Ninth Annual ACM Symposium On Principles of Distributed Computing*, pages 311–234, August 1990.

7. M. Fischer, N.A. Lynch, and M.S. Paterson. Impossibility of distributed commit with one faulty process. *Journal of the ACM*, 32(2), April 1985.
8. Eli Gafni. Round-by-round fault detectors (extended abstract): unifying synchrony and asynchrony. In *PODC '98: Proceedings of the seventeenth annual ACM symposium on Principles of distributed computing*, pages 143–152, New York, NY, USA, 1998. ACM Press.
9. Maurice Herlihy and Nir Shavit. The asynchronous computability theorem for t-resilient tasks. In *STOC '93: Proceedings of the twenty-fifth annual ACM symposium on Theory of computing*, pages 111–120, New York, NY, USA, 1993. ACM Press.
10. Maurice Herlihy and Nir Shavit. The topological structure of asynchronous computability. *J. ACM*, 46(6):858–923, 1999.
11. M.P. Herlihy. Wait-free synchronization. *ACM Transactions On Programming Languages and Systems*, 13(1):123–149, January 1991.
12. J.R. Munkres. *Elements Of Algebraic Topology*. Addison Wesley, Reading MA, 1984. ISBN 0-201-04586-9.
13. M. Saks and F. Zaharoglou. Wait-free k-set agreement is impossible: The topology of public knowledge. In *Proceedings of the 1993 ACM Symposium on Theory of Computing*, pages 101–110, May 1993.
14. Michael E. Saks and Fotios Zaharoglou. Wait-free k-set agreement is impossible: The topology of public knowledge. *SIAM J. Comput.*, 29(5):1449–1483, 2000.