Set Consensus Using Arbitrary Objects

(Preliminary Version)

Maurice Herlihy
Digital Equipment Corporation
Cambridge Research Laboratory
One Kendall Square
Cambridge, MA 02139
herlihy@crl.dec.com

Sergio Rajsbaum*
MIT Laboratory for Computer Science
545 Technology Square
Cambridge, MA 02139
rajsbaum@theory.lcs.mit.edu.

Abstract

In the (N, k)-consensus task, each process in a group starts with a private input value, communicates with the others by applying operations to shared objects, and then halts after choosing a private output value. Each process is required to choose some process's input value, and the set of values chosen should have size at most k. This problem, first proposed by Chaudhuri in 1990, has been extensively studied using asynchronous read/write memory. In this paper, we investigate this problem in a more powerful asynchronous model in which processes may communicate through objects other than read/write memory, such as test&set variables.

We prove two general theorems about the solvability of set consensus using objects other than read/write registers. The first theorem addresses the question of what kinds of shared objects are needed to solve (N, k)-consensus, and the second addresses the question of what kinds of tasks can be solved by N processes using (M, j)-consensus objects, for $M \leq N$. Our proofs exploit a number of techniques from algebraic topology.

1 Introduction

An asynchronous concurrent system consists of a set of processes that communicate by applying operations to shared objects. An object's *type* specifies the operations it provides and their meanings. Examples of objects include read/write registers, test&set registers, fetch&add registers, and compare&swap registers.

A *task* is a problem where each process starts with a private input value, communicates by applying operations to shared objects, and halts with a private output value. A set of input values defines an initial configuration, while a set of output values defines a final configuration. A task is specified by a set of possible initial configurations, and for each initial configuration, the set of legal final configurations.

Modern multiprocessors are inherently *asynchronous*: processes can be halted or delayed without warning by interrupts, pre-emption, or cache misses. In such environments, it is desirable to design protocols that are *wait-free*: any processes that continues to run will choose an output value in a fixed number of steps, regardless of delays or failures by other processes.

Under what circumstances does a task have a wait-free solution? It is known that the computational power of a concurrent system depends on the types of the shared objects. For example, every object can be assigned a *consensus number* [10] which partly characterizes its computational power: in a system of N or more concurrent processes, it is impossible to construct a wait-free implementation of an object with consensus number N from an object with a lower consensus number. On the other hand, any object with consensus number N is *universal* in a system of N processes: it implements any other concurrent object. More recently, Herlihy and Shavit [11, 12] have given a general combinatorial characterization of the decision tasks that can be solved in read/write memory (which has consensus number one).

Nevertheless, relatively little is known about the computational power of objects with consensus number greater than one but less than N. This ques-

*On leave from Instituto de Matemáticas, U.N.A.M., México. Partly supported by DGAPA Projects.

0

tion has considerable practical interest, since many modern multiprocessors provide such objects (such as test&set or memory-to-register swap) as built-in synchronization primitives. In this paper, we make some progress on this basic question by investigating the circumstances under which one can solve the (N, k)-*consensus* task using objects other than read/write registers. In the (N, k)-consensus task, each of N processes starts with an integer input value, and each chooses some process's input value, such that the number of values chosen has size at most k. (When $k = 1$, this problem is the well-known *consensus* problem [6].) The (N, k)-consensus problem was first posed in 1990 by Chaudhuri [4] (who called it *k-set agreement*), along with a conjecture that it could not be solved in read/write memory. This conjecture was finally proved by three independent research teams in 1993 [2, 11, 15]. When the specific values of N and k are unimportant, we will refer to this problem as *set consensus*.

In this paper we prove two general theorems about the solvability of set consensus using objects other than read/write registers. The first theorem addresses the question of what kinds of shared objects are needed to solve set consensus. Any protocol employing shared objects of any kind has a characteristic geometric structure, called a *simplicial complex* [11]. The topological properties of this complex are determined by the types of the shared objects. In our first result, we show that a protocol cannot implement set consensus if the protocol's associated complex lacks holes of sufficiently small dimension (i.e., if certain low-dimensional homology groups are trivial).

The second theorem addresses the question of what kinds of tasks can be solved by N processes using (M, j)-consensus objects, for $M \leq N$. We show that the complex associated with any N-process protocol using (M, j)-consensus objects starting from a fixed set of inputs has no holes of dimension less than $j \cdot \lfloor (N-1)/M \rfloor$ (i.e., these low-order homology groups are trivial). When $j = 1$, then M is the object's *consensus number* [10]. This theorem gives an intriguing topological interpretation to consensus numbers. At one extreme, when $M = 1$, the complex has no holes at all (i.e., it has trivial homology in all dimensions), reflecting a prior result of Herlihy and Shavit [11]. For consensus numbers above one, however, the complex may have holes. If the consensus number is low, then holes appear only in higher dimensions, but as the consensus number grows, the holes spread into increasingly lower dimensions. Finally, when the consensus number $M = N$, the complex becomes disconnected.

Together, these theorems imply, for example, that it is impossible to implement $(n+1, k)$-consensus using (m, j)-consensus objects if $n/k > m/j$, an observation also made by Borowsky and Gafni [3]. One intriguing consequence of this result is that there exist objects whose computational powers are *incomparable*: there is no wait-free implementation of X by Y, and vice-versa. For example, $(2, 1)$-consensus cannot implement $(6, 2)$-consensus because $5/2 > 2/1$. Conversely, Herlihy and Shavit [11] have shown that one cannot implement $(2, 1)$-consensus using $(N, 2)$-consensus for any N.

It should be noted that our theorems extend to any variation of (N, k)-consensus which also places a *lower* bound on the number of values chosen. For example, a related task might require that if processes have unique inputs, then the number of values chosen lies between ℓ and $k \geq \ell \geq 1$.

Our arguments make extensive use of concepts, such as simplicial complexes and homology groups, taken from undergraduate-level algebraic topology. We believe this topological approach has a great deal of promise for the theory of distributed and concurrent computation, and that it merits further investigation. It has already yielded a number of results, including, for read/write memory, a general characterization of tasks having wait-free solutions [11, 12], the impossibility of set consensus [11], and the impossibility of renaming [1] with a small number of names [11]. In the synchronous model, Chaudhuri, Herlihy, Lynch, and Tuttle [5] used this approach to give tight upper and lower bounds on the complexity of solving set consensus using message-passing. This paper is the first application of these techniques to objects other than read/write registers.

2 Model

Our model is the same as [11].

An initial or final state of a process is modeled as a *vertex*, a pair consisting of a process id and a value (either input or output). A set of $d+1$ mutually compatible initial or final states is modeled as a *d-dimensional simplex*, (or d-simplex). It is convenient to visualize a vertex as a point in Euclidian space, and a simplex as the convex hull of a set of affinely-independent vertexes, the higher-dimensional analogue of a solid triangle or tetrahedron. The complete set of possible initial (or final) states is represented by a set of simplexes, closed under intersection, called a *simplicial complex* (or complex). A *principal* simplex in a complex $\mathcal{C}$ is one not contained in a simplex of higher dimension. The *dimension* of $\mathcal{C}$ is the smallest dimension of any principal simplex. Where

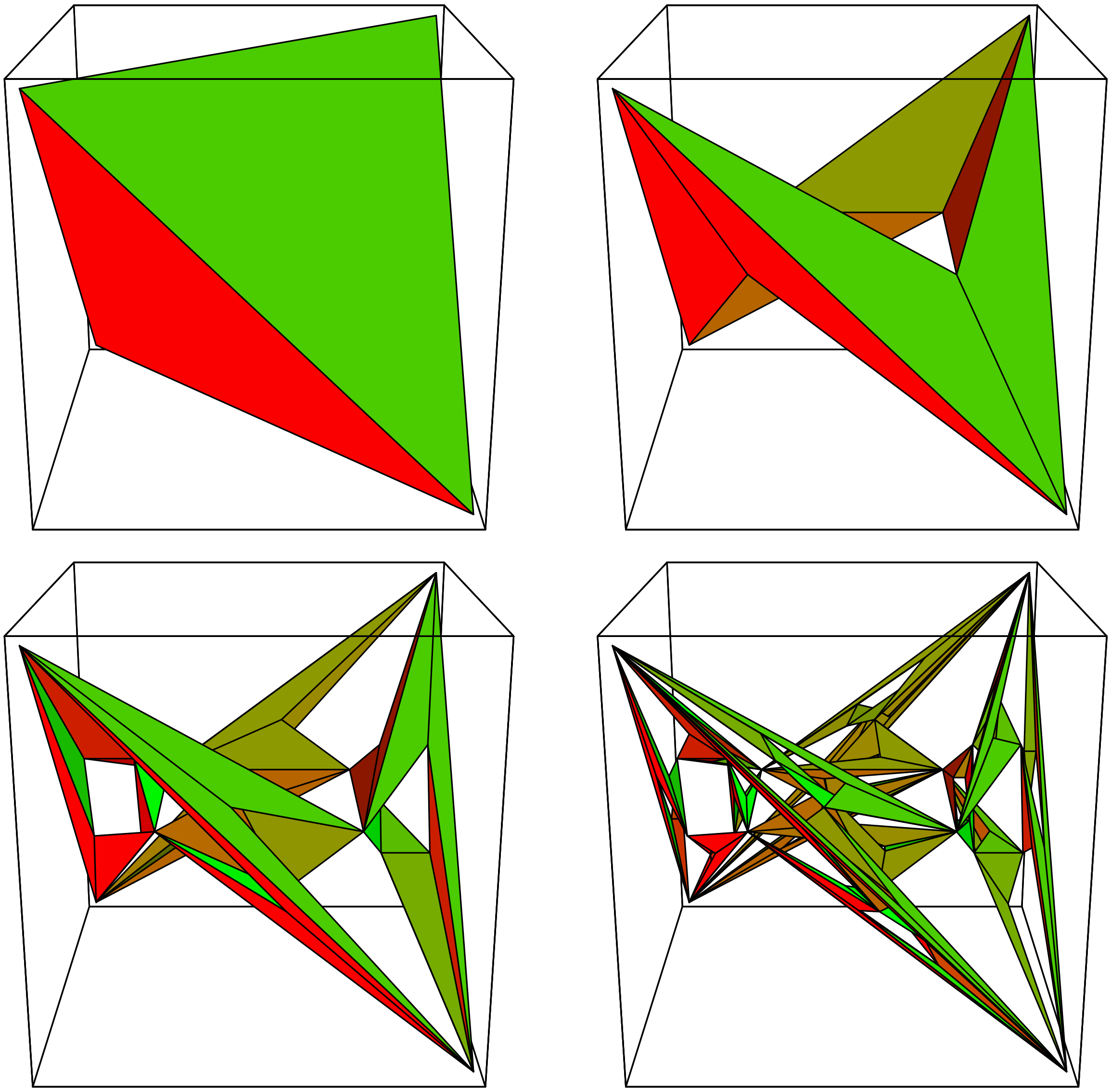

Figure 1: Full-Information Complexes for multi-round Test&Set Protocols

convenient, we use superscripts to denote the dimension of a simplex or complex.

A *task specification* for $N = n+1$ processes is given by an input complex $\mathcal{I}^n$, an output complex $\mathcal{O}^n$, and a map Δ carrying each input simplex of $\mathcal{I}^n$ to a set of simplexes of $\mathcal{O}^n$. This map associates with each initial state of the system (an input simplex) the set of legal final states (output simplexes). When $m < n$, $\Delta(S^m)$ indicates the legal final states in executions where only $m+1$ out of $n+1$ processes take steps (the rest fail before taking any steps). A *solution* to a task is a protocol in which the processes communicate by applying operations to objects in shared memory, and eventually halt with mutually compatible decision values. A *wait-free* solution is one which tolerates the failure of up to n out of $n+1$ processes.

Any protocol that solves a task can also be associated with a *full-information complex*, 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.

For example, Figure 1 shows full-information complexes for four simple four-process protocols. In the first (degenerate) protocol, each process halts without communicating with any other. Because each process has only one possible decision value, the full-information complex consists of a single simplex. Because there are four processes, this simplex is a tetrahedron. In the second protocol, the first two processes share a test&set variable, as do the second two. This complex consists of four tetrahedrons, corresponding to the four possible outcomes of the two test&set operations. In the next two protocols (encompassing 4^2 and 4^3 tetrahedrons), the processes respectively iterate two and three-round test&set protocols, using fresh variables for each round. This sequence of pictures is suggestive: full-information complexes generated by protocols in which processes communicate by test&set variables may have an arbitrary number of "holes" (unlike protocols employing only read/write variables), but they still appear to remain connected. Our theorems confirm and generalize this intuition.

A *simplicial map* carries vertexes of one complex to vertexes of another so that simplexes are preserved. Let $\mathcal{F}^n$ be the full-information complex for a protocol Π. If S is an input simplex of dimension less than or equal to n, let $\mathcal{F}^n(S) \subset \mathcal{F}^n$ denote the complex of final states reachable from the initial state S. Π solves the decision task $\langle \mathcal{I}^n, \mathcal{O}^n, \Delta \rangle$ if and only if there exists a simplicial map $\delta : \mathcal{F}^n \to \mathcal{O}^n$, called a *decision map*, such that for every input simplex S, $\delta(\mathcal{F}^n(S)) \subset \Delta(S)$. We prove our impossibility results by exploiting the topological properties of the full-information complex and the output complex to show that no such map exists.

Many complexes of interest have a simple but important topological property: they have no "holes" in certain dimensions. Formally, this concept is captured by the notion of a *homology group*. (Readers completely unfamiliar with the notion may wish to consult any one of a number of standard textbooks [7, 8, 9, 13, 16].) For our purposes, it suffices to note that an n-dimensional complex $\mathcal{C}^n$ has $n+1$ homology groups, $H_0(\mathcal{C}^n), \ldots, H_n(\mathcal{C}^n)$, one for each dimension. For $i > 0$, if $H_i(\mathcal{C}^n) = 0$, the trivial single-element group, then $\mathcal{C}^n$ has no holes of that dimension. When $i = 0$, there are two alternative ways to define $H_0(\mathcal{C}^n)$. In this paper, $H_0(\mathcal{C}^n)$ always denotes the *reduced* 0-th homology group [16, p.172], which is trivial if and only if $\mathcal{C}^n$ is connected.

Our principal tool for computing homology groups for full-information complexes is the *Mayer-Vietoris sequence* [16, p.186]. A sequence of groups G_i is said to be *exact* if there exist group homomorphisms $\phi_i : G_i \to G_{i+1}$ such that the image of ϕ_i is the kernel of ϕ_{i+1}. It is easily checked that if $G_{i+1} = G_{i-1} = 0$, then $G_i = 0$. The Mayer-Vietoris sequence states that if $\mathcal{A}$ and $\mathcal{B}$ are complexes of dimension n such that $\mathcal{A} \cap \mathcal{B} \neq \emptyset$,

$$\cdots \to H_q(\mathcal{A} \cap \mathcal{B}) \to H_q(\mathcal{A}) \oplus H_q(\mathcal{B}) \to H_q(\mathcal{A} \cup \mathcal{B}) \to H_{q-1}(\mathcal{A} \cap \mathcal{B}) \to \cdots$$

is an exact sequence.

A complex is *connected* if there is a path of edges (1-simplexes) between every pair of vertices. It is *simply connected* if it is connected and every closed path can be continuously deformed to a point. (Any simply connected complex has a trivial first homology group ([8, Ch. 12], but not vice-versa [8, p.150].)

3 Impossibility

Consider a system of $N = n+1$ processes sharing read/write variables and a collection of objects O. Fix an input simplex S^n with distinct input values for each process, and let $\mathcal{F}(S^n)$ be the full information complex of a protocol Π in which the processes start in initial state S^n and communicate via objects O. If U is a subset of the set of process ids in S^n, let $\mathcal{F}(U)$ be the full information complex for executions of the protocol in which only the processes in U take steps. Our first main result is the following.

Theorem 3.1 *Let Π be a protocol such that*

- *for $|U| \geq c$, $\mathcal{F}(U)$ is connected,*

- *for* $|U| \geq 2c$, $\mathcal{F}(U)$ *is simply connected, and*
- *for* $|U| > q \cdot c$, $H_q(\mathcal{F}(U)) = 0$,

then Π *cannot solve* $(n+1, \lfloor n/c \rfloor)$*-consensus.*

The argument is a generalization of the argument used to prove impossibility of (N, k)-consensus in read/write memory. Because the full-information complex for read/write protocols has no holes, one can show that it contains an n-dimensional subdivided simplex whose decision values define a Sperner coloring (defined below), and hence that some simplex is colored with $n+1$ distinct colors, making $(n+1, n)$-consensus impossible. If the complex has higher-dimensional holes, then we can still construct a subdivided simplex with a Sperner coloring, except that we need a higher-dimensional complex to avoid the holes.

The proof of the following lemma (omitted) is based on the Hurewicz Isomorphism Theorem [16, p398], which states that if $\mathcal{C}$ is simply connected with $H_q(\mathcal{C}) = 0$ for $q < k$, then any continuous map of a $(k-1)$-sphere to $\mathcal{C}$ can be extended to a continuous map of the k-disk.

Lemma 3.2 *Let* S^k *be a* k*-simplex for* $k \leq \lfloor n/c \rfloor$, $\dot{S}^k$ *the boundary of* S^k *(the set of faces), and* $\sigma(\dot{S}^k)$ *an iterated barycentric subdivision. Any simplicial map of the subdivided boundary*

$$\phi : \sigma(\dot{S}^k) \rightarrow \mathcal{F}(S^n)$$

can be extended to the interior:

$$\Phi : \tau(S^k) \rightarrow \mathcal{F}(S^n)$$

where τ *is an iterated barycentric subdivision refining* σ.

Let $Q_i = P_{i \cdot c}$, for $0 \leq i \leq \lfloor n/c \rfloor$, and let

$$G_i = \{P_{i \cdot c}, \ldots, P_{(i+1) \cdot c - 1}\}$$

for $0 \leq i < \lfloor n/c \rfloor$. We call each Q_i a *principal process.* Let W be a set of $\ell + 1$ principal processes, and let $S^\ell(W)$ be a simplex with vertices labeled with process identifiers from W. Abusing notation, we use Q_i to denote the vertex labeled with Q_i in $S^\ell(W)$, and $\mathcal{F}(Q_i)$ the unique vertex of the full-information complex associated with the solo execution of Q_i.

Definition 3.1 A *span* for $S^\ell(W)$ is a simplicial map $\phi : \sigma(S^\ell(W)) \rightarrow \mathcal{F}^n(S^n)$, with the following properties.

- If $W = \{Q_{i_0}, \ldots, Q_{i_\ell}\}$, for $i_0 < \ldots < i_\ell$, then ϕ carries $S^\ell(W)$ to $\mathcal{F}(G_{i_0} \cup \cdots \cup G_{i_{\ell-1}} \cup \{Q_{i_\ell}\})$. (In particular, if $W = \{Q_i\}$, then $\phi(S^0(\{Q_i\}))$ is $\mathcal{F}(Q_i)$.)
- ϕ restricted to each face of $S^\ell(W)$ is a span for that face.

Lemma 3.3 *If* U *is the set of all* $\lfloor n/c \rfloor + 1$ *principal processes, then there exists a span* $\Phi : \sigma(S^{\lfloor n/c \rfloor}(U)) \rightarrow \mathcal{F}^n(S^n)$.

Proof: We show, by induction on ℓ, that every ℓ-subsimplex $S^\ell(W)$ has a span ϕ_W, and that these spans agree on their common intersections. When $W = \{Q_i\}$, define $\phi_{\{Q_i\}}$ to send $S^0(\{Q_i\})$ to $\mathcal{F}(Q_i)$.

Suppose $W = \{Q_i, Q_j\}$ for $i < j$. Because $\mathcal{F}^c(G_i \cup \{Q_j\})$ is connected, there exists a path from the vertex $\mathcal{F}(Q_i)$ to $\mathcal{F}(Q_j)$ in $\mathcal{F}^c(G_i \cup \{Q_j\})$. We use this path to define a subdivision σ and a simplicial map

$$\phi_W : \sigma(S^1(W)) \rightarrow \mathcal{F}^c(G_i \cup \{Q_j\})$$

such that $\phi_W(Q_i) = \phi_{\{P_i\}}(Q_i)$, and $\phi_W(Q_j) = \phi_{\{P_j\}}(Q_j)$. Notice that any two different maps $\phi_{\{P_i, P_j\}}$ and $\phi_{\{P_i, P_k\}}$ agree on the intersection of their domains.

Inductively, assume the claim holds for every $\ell < \lfloor n/c \rfloor$, and consider the simplex $S^{\lfloor n/c \rfloor}(U)$. Every face of this simplex has a span, and the spans agree on their common intersections, so together these spans induce a simplicial map ϕ from a subdivision of $\dot{S}^{\lfloor n/c \rfloor}(U)$ into $\mathcal{F}(S^n)$. Because $H_q(\mathcal{F}(S^n)) = 0$ for $q < \lfloor n/c \rfloor$, Lemma 3.2 implies that ϕ can be extended to a simplicial map

$$\Phi : \tau(S^{\lfloor n/c \rfloor}(U)) \rightarrow \mathcal{F}^n(G_0 \cup \cdots \cup G_{\lfloor n/c \rfloor - 1} \cup \{Q_{\lfloor n/c \rfloor}\})$$

■

Like all known impossibility proofs for (N, k)-consensus, our proof relies on Sperner's Lemma. Let $bary^r(S^k)$ denote the r-th barycentric subdivision of simplex S^k.

Definition 3.2 The *carrier* of a vertex $\vec{v}$ in $bary^r(S^k)$ is the smallest-dimensional $S^l \subseteq S^k$ containing $\vec{v}$.

Lemma 3.4 (Sperner's Lemma) *If* χ *is a map sending each vertex* $\vec{v}$ *of* $bary^r(S^k)$ *to a vertex in its carrier, then there is at least one* k*-simplex* $R^k = \langle \vec{r}_0, \ldots, \vec{r}_k \rangle$ *in* $bary^r(S^k)$ *such that the* $\chi(\vec{r}_i)$ *are all distinct.*

We are now ready to prove Theorem 3.1.

Proof: Color each vertex $\vec{v}$ of $\Phi(\sigma(S^{\lfloor n/c \rfloor}(U)))$ with the value that the process decides at the end of the execution. Let p_i be the value decided by P_i. If we "reset" each $p_j \in G_i$ to $p_{i \cdot c}$, then the result is a Sperner coloring of $\sigma(S^{\lfloor n/c \rfloor}(U))$, and therefore some simplex has all $\lfloor n/c \rfloor + 1$ colors. It follows that some simplex in the original coloring must have had $\lfloor n/c \rfloor + 1$ distinct colors. Since the colors are distinct, Φ carries these vertexes in $\sigma(S^{\lfloor n/c \rfloor}(U))$ to vertexes labeled with distinct processes in $\mathcal{F}(S^n)$, and therefore there is some execution in which $\lfloor n/c \rfloor + 1$ processes all choose distinct values, and therefore $(n+1, \lfloor n/c \rfloor)$-consensus is impossible. ∎

4 Topology of Set Consensus

In this section we characterize the tasks that can be solved by $n+1$ processes using a combination of (m,j)-consensus objects and read/write objects. An (m,j)-consensus object can be accessed by at most m processes, and the set of values returned by the object is of size at most j. For our bounds to be meaningful, we assume that j is as small as possible; e.g. if at least j processes access the object then there are executions where j different values are returned. Under this assumption, an (m,j)-consensus object is not an $(m,j+1)$-consensus object.

Because an (m,j)-consensus object is nondeterministic, we assume that the choice of responses is controlled by an adversary. We will show that an adversary can force any protocol Π using (m,j)-consensus objects to have a full-information complex with no holes in the lower dimensions. Without loss of generality (for impossibility results), we can constrain the adversary to choose responses according to any convenient strategy consistent with the (m,j)-consensus specification. If the constrained adversary can force the full information protocol not to have holes in the lower dimensions, then so can any stronger adversary. We henceforth restrict our attention to the following deterministic adversary: each of the first j processes receives its own input back, and the remaining $m-j$ processes get the very first process's input. A process can tell if it is one of the first j processes (it gets its own input back), and otherwise it observes the identity of the first process to access the object.

Fix an input simplex S^n. Let $\mathcal{C}^n(m,j)$ be the full-information complex for any protocol using (m,j)-consensus objects and starting in S^n. The main result of this section is the following: $\mathcal{C}^n(m,j)$ is simply connected with

$$H_q(\mathcal{C}^n(m,j)) = 0 \text{ for } q < j \cdot \left\lfloor \frac{n}{m} \right\rfloor.$$

For brevity, we refer to $H_q(\mathcal{C}^n(m,j))$ for $q < j \cdot \lfloor n/m \rfloor$ as the *low-order* homology groups.

The proof of this claim is by contradiction. Assume that some low-order homology group is initially non-trivial. When the protocol is finished, the reachable complex is a single simplex, all of whose homology groups vanish. Because the low-order groups eventually henceforth vanish, there exists a *critical state* in which some low-order group is non-trivial, but any step by any process will cause them to vanish henceforth. We then derive a contradiction by showing that the low-order groups of the reachable complex from the critical state must already be trivial.

4.1 Reachable Complexes

Starting from the initial state defined by the input simplex S^n, we can run the protocol and after some time "freeze" the system. The *state* at this moment specifies the local state of each process and the contents of the shared memory. All the executions of the system starting at this state define a complex as follows.

Definition 4.1 A simplex R^m of the full information complex $\mathcal{C}^n$ is *reachable from state* s in a history if there is some execution starting from s in which each process in $ids(R^m)$ is decided and has the view specified in R^m. The *reachable complex* from state s, $\mathcal{F}(s)$ is the complex of reachable simplexes from s.

Our approach is based on an inductive application of the Mayer-Vietoris sequence. Let $\mathcal{F}$ be a subcomplex of the reachable complex, and W an arbitrary index set. Consider a set $\{\mathcal{C}_i : i \in W\}$ of subcomplexes of $\mathcal{F}$ that *cover* $\mathcal{F}$: $\mathcal{F} = \cup_{i \in W} \mathcal{C}_i$. Define, for every $U \subseteq W$,

$$\mathcal{C}_U = \bigcap_{i \in U} \mathcal{C}_i.$$

Define $\mathcal{C}_\emptyset = \mathcal{F}$.

Lemma 4.1 *For any subsets U and V of W,*

$$\mathcal{C}_U \cap \mathcal{C}_V = \mathcal{C}_{U \cup V}.$$

Proof:

$$\mathcal{C}_U \cap \mathcal{C}_V = (\bigcap_{i \in U} \mathcal{C}_i) \bigcap (\bigcap_{i \in V} \mathcal{C}_i) = \bigcap_{i \in U \cup V} \mathcal{C}_i = \mathcal{C}_{U \cup V}.$$

∎

Definition 4.2 A real-valued map θ on natural numbers is *slowly decreasing* if

$$\theta(u+1) \le \theta(u) \le \theta(u+1) + 1.$$

We now reduce the problem of showing that $\mathcal{F}$ has trivial low-order homology groups to showing that property for intersections of the $\mathcal{C}_i$.

For each u, $1 \leq u \leq |W|$, let $U_0^{(u)}, \ldots, U_\ell^{(u)}$ be subsets of W such that each $|U_i^{(u)}| = u$, and for each distinct $U_i^{(u)}$ and $U_j^{(u)}$, $\{i\} = U_i^{(u)} - U_j^{(u)}$. (Equivalently, there is a set U, $|U| = u - 1$, such that $U_i^{(u)} = U^{(u)} \cup \{i\}$.) We omit superscripts whenever convenient.

Lemma 4.2 *Let $\theta(u)$ be a slowly-decreasing map such that*

(i) if $\theta(|U|) \geq 0$, then $\mathcal{C}_U \neq \emptyset$, and

(ii) $H_q(\mathcal{C}_U) = 0$ for $q < \theta(|U|)$,

then for $1 \leq u \leq |W|$ and $0 \leq \ell \leq |W| - u$,

$$H_q(\bigcup_{i=0}^{\ell} \mathcal{C}_{U_i^{(u)}}) = 0 \text{ for } q < \theta(u).$$

Proof: We argue by reverse induction on u and induction on ℓ. In the base case for u, when $u = |W|$, we have $\ell = 0$ and only one set $U_0 = W$;

$$H_q(\mathcal{C}_W) = 0 \text{ for } q < \theta(|W|).$$

follows from the hypothesis (ii).

Assume the claim for sets of size $u+1$, we prove it for u such that $\theta(u) > 0$. The base case for $\ell = 0$ also follows from hypothesis (ii). We assume the claim for $\ell - 1$ and show it for ℓ.

Consider the Mayer-Vietoris sequence for $\mathcal{C}_{U_\ell}$ and $\cup_{i=0}^{\ell-1} \mathcal{C}_{U_i}$:

$$\begin{aligned} \cdots \quad &\to H_q(\mathcal{C}_{U_\ell}) \oplus H_q(\textstyle\bigcup_{i=0}^{\ell-1} \mathcal{C}_{U_i}) \to H_q(\bigcup_{i=0}^{\ell} \mathcal{C}_{U_i}) \\ &\to H_{q-1}(\mathcal{C}_{U_\ell} \cap (\textstyle\bigcup_{i=0}^{\ell-1} \mathcal{C}_{U_i})) \to \cdots \end{aligned}$$

By hypothesis (ii), $H_q(\mathcal{C}_{U_\ell}) = 0$ for $q < \theta(u)$, and by the induction hypothesis for ℓ,

$$H_q(\bigcup_{i=0}^{\ell-1} \mathcal{C}_{U_i}) = 0 \text{ for } q < \theta(u).$$

Let $V_0, \ldots, V_{\ell-1}$ be sets such that $V_i = U_\ell \cup U_i$.

$$\mathcal{C}_{U_\ell} \cap (\bigcup_{i=0}^{\ell-1} \mathcal{C}_{U_i}) = \bigcup_{i=0}^{\ell-1} (\mathcal{C}_{U_\ell} \cap \mathcal{C}_{U_i}) = \bigcup_{i=0}^{\ell-1} \mathcal{C}_{V_i},$$

where the last equality follows from Lemma 4.1. Notice that $|V_i| = u+1$, and for each distinct V_i and V_j, $\{i\} = V_i - V_j$, so by the induction hypothesis for u:

$$H_q(\bigcup_{i=0}^{\ell-1} \mathcal{C}_{V_i}) = 0 \text{ for } q < \theta(u+1),$$

or

$$H_{q-1}(\bigcup_{i=0}^{\ell-1} \mathcal{C}_{V_i}) = 0 \text{ for } q < \theta(u+1) + 1.$$

Because θ is slowly decreasing, $\theta(u) \leq \theta(u+1) + 1$, and thus these groups are trivial for $q < \theta(u)$.

Since $|V_i| = u + 1$, by hypothesis (i) $\bigcup_{i=0}^{\ell-1} \mathcal{C}_{V_i}$ is not empty. From exactness of the Mayer-Vietoris sequence, $H_q(\cup_{i=0}^{\ell} \mathcal{C}_{U_i}) = 0$ for $q < \theta(u)$. ■

4.2 Critical States

Let $\wp$ be an arbitrary property of a state.

Definition 4.3 A state s is *critical* for a property $\wp$ if $\wp$ does not hold for s, but any step by any undecided process will enter a state s' where $\wp$ henceforth holds, i.e., for any state s'' reachable from s' (including s' itself), $\wp$ holds for s''.

A *final state* of a protocol is one in which every process is halted. Thus if s is a final state then $\mathcal{F}(s)$ is a single simplex.

Lemma 4.3 *Let $\wp$ be a property that holds in every final state of a protocol. If the property does not hold in a state, then some execution from that state leads to a critical state for $\wp$.*

Proof: The *execution tree* at a state s is defined as follows. The root is s. For each state s' in the tree, the children correspond to the states reached by the execution of each pending operation in s'.

Starting in the state s where the property is false, consider the tree of executions starting in s. Color red the nodes of this tree where the property holds and black the rest. Every leaf has red color and s has black color. Therefore there exists a black node s' with all of its children colored red. This state s' is critical. ■

4.3 Pending Operations

A process which has not decided in s has a *pending* operation, which it is about to perform. Since we consider only deterministic protocols, P's pending operation is uniquely defined. Let P_i be an undecided process at a state s, about to execute operation p_i, and W the set of indexes of pending operations. Define $\mathcal{C}_i(s)$, $i \in W$, to be the reachable complex immediately after P_i executes p_i. A simplex S is in $\mathcal{C}_i(s)$ if there exists an execution from s, starting with operation p_i by P_i, such that for each vertex $\langle Q, \nu \rangle \in S$, process Q has view ν.

Define $\mathcal{C}_U(s) = \bigcap_{i \in U} \mathcal{C}_i(s)$, for any $U \subseteq W$, and $\mathcal{C}_\emptyset(s) = \mathcal{F}(s)$, the entire reachable complex. Intuitively, each vertex in $\mathcal{C}_U(s)$ corresponds to an execution starting in s in which no process can tell which process in U went first. More formally, if a vertex $\langle Q, \nu \rangle$ is in $\mathcal{C}_U$, then for each $i \in U$ there is an execution starting with p_i in which Q's view is ν. Observe that Lemma 4.1 holds for the $\mathcal{C}_U(s)$.

Lemma 4.4 *Let s be a state of the protocol, and W be the set of indexes of pending operations in s. Then the $\mathcal{C}_i(s)$ cover $\mathcal{F}(s)$: $\mathcal{F}(s) = \cup_{i \in W} \mathcal{C}_i(s)$.*

Lemma 4.5 *Let $|U| = u$. For any state s,*

$$j \cdot \left\lfloor \frac{dim(\mathcal{C}_U(s))}{m} \right\rfloor \geq j \cdot \left\lfloor \frac{n}{m} \right\rfloor - u + 1.$$

Proof: A process P_i is *potentially disabled* with respect to $\mathcal{C}_U(s)$ if there is an execution α in which every process that decides chooses a vertex in $\mathcal{C}_U(s)$, but there is no extension of α in which P_i decides a vertex in $\mathcal{C}_U(s)$. The dimension of $\mathcal{C}_U(s)$ is just a way of counting the minimal number of processes that cannot become disabled with respect to $\mathcal{C}_U(s)$. We now undertake a case analysis to analyze how processes become disabled.

We may assume without loss of generality that all read/write registers are single-reader and single-writer, since they are sufficient to construct wait-free implementations of multi-reader/multi-writer registers (e.g., [14]). Suppose U includes a process P about to write a register, while Q is about to read that register. Q is disabled, since it can observe the relative order of the two operations. Each such read/write conflict disables one process.

A *used* (m, j)-consensus object is one that has already been accessed by an operation in s. If the object has already been accessed a times, and it is accessed c times by operations in U, the largest number of operations that can become disabled is c (when $a = j - 1$ and $c \geq 2$).

A *new* (m, j)-consensus object is one that has never been accessed. If a new object is accessed by more than j processes in U, then a maximum of $m - j + 1$ processes may become disabled: the first j receive their own values, and the remaining j learn the identity of the first process to access the object. Of the j that receive their own values, only $j - 1$ may continue to run, since any process that observes the results of all j observes that the remaining processes in U did not go first.

Therefore, the most effective way to disable processes is to have at least $j + 1$ processes access each consensus variable, disabling at most $\lfloor u/(j+1) \rfloor \cdot (m - j + 1)$ processes:

$$\begin{aligned} dim(\mathcal{C}_U(s)) &\geq n + 1 - \left\lfloor \frac{u}{j+1} \right\rfloor (m - j + 1) \\ &> n - \left\lfloor \frac{u}{j+1} \right\rfloor m. \end{aligned}$$

Dividing both sides by m, taking floors, and multiplying by j yields:

$$j \cdot \left\lfloor \frac{dim(\mathcal{C}_U(s))}{m} \right\rfloor \geq j \cdot \left(\left\lfloor \frac{n}{m} \right\rfloor - \left\lfloor \frac{u}{j+1} \right\rfloor \right)$$

The result follows from the identity

$$j \cdot \left\lfloor \frac{u}{j+1} \right\rfloor \leq u - 1.$$

■

Lemma 4.6 *If, for all $U \subseteq W$,*

$$H_q(\mathcal{C}_U(s)) = 0 \text{ for } q < j \cdot \left\lfloor \frac{dim(\mathcal{C}_U(s))}{m} \right\rfloor,$$

then

$$H_q(\mathcal{F}(s)) = 0 \text{ for } q < j \cdot \left\lfloor \frac{dim(\mathcal{F}(s))}{m} \right\rfloor.$$

Proof: The function

$$\theta(u) = j \cdot \lfloor n/m \rfloor - u + 1$$

is slowly decreasing, and by Lemma 4.5,

$$j \cdot \left\lfloor \frac{dim(\mathcal{C}_U(s))}{m} \right\rfloor \geq \theta(|U|).$$

If $\theta(|U|) \geq 0$, then $j \cdot \lfloor dim(\mathcal{C}_U(s))/m \rfloor \geq 0$, hence $dim(\mathcal{C}_U(s)) \geq 0$ and $\mathcal{C}_U(s)$ is non-empty. The hypothesis implies that

$$H_q(\mathcal{C}_U(s)) = 0 \text{ for } q < \theta(|U|),$$

so $\mathcal{C}_U(s)$ satisfies the conditions of Lemma 4.2:

$$H_q(\bigcup_{i=0}^{\ell} \mathcal{C}_{U_i}(s)) = 0 \text{ for } q < \theta(|U_i|).$$

When U_i is a singleton set, $\cup_{i=0}^{\ell} \mathcal{C}_i(s) = \mathcal{F}(s)$, and

$$H_q(\mathcal{F}(s)) = 0 \text{ for } q < \theta(1) = j \cdot \left\lfloor \frac{dim(\mathcal{F}(s))}{m} \right\rfloor.$$

■

Lemma 4.7 $H_q(\mathcal{C}_U(s)) = 0$ *for* $q < j \cdot \left\lfloor \frac{dim(\mathcal{C}_U(s))}{m} \right\rfloor$.

Proof: **(Sketch)** Suppose not. Pick the smallest n for which the lemma fails to hold. Define the predicate

$$\wp(s) = (\forall U) H_q(\mathcal{C}_U(s)) = 0 \text{ for } q < j \cdot \left\lfloor \frac{dim(\mathcal{C}_U(s))}{m} \right\rfloor$$

where U ranges over all sets of processes with pending operations. In any final state, $\mathcal{C}_U(s) = \mathcal{C}_\emptyset(s)$, which is a single simplex, so $\wp$ holds. By Lemma 4.3 there exists a state s^* critical for $\wp$ for the protocol Π. Let U^* be a set that violates $\wp$.

A set of operations is *commuting* if executing them in any order returns the same results to the calling processes and leave the objects in the same state. A set of operations is *weakly commuting* if applying them in any order leaves the objects in the same state (although the operations may return different results depending on the order). For example, pending read and write operations to the same object are weakly commuting, as are pending operations to a used (m, j)-consensus object. Pending operations on distinct objects commute. Commuting implies weakly commuting.

Suppose we execute the operations in U in some order. A *winner set* is the set of process ids from any simplex in $\mathcal{C}_U(s)$, and a *loser set* is the complement of a winner set in U. A winner set is *maximal* if it is not contained in any larger winner set, and the complementary loser set is *minimal.* Winner sets are closed under intersection, and loser sets under union. Any winner set can be expressed as the intersection of maximal winner sets, any loser set as the union of minimal loser sets.

We consider two cases: (1) the pending operations in U are weakly commuting, and (2) some object's pending operations in U are not weakly commuting.

In the first case, for each winner set W_i and complementary L_i, let s_i be the state reached from s^* by executing the operations in W_i followed by the operations in L_i. Let Π_i be the protocol starting in s_i identical to Π except that the processes in L_i do not participate. Let $\mathcal{D}_i$ be the reachable complex for Π_i in s_i.

$$\mathcal{C}_{U^*}(s^*) = \bigcup \mathcal{D}_i$$

where the index i ranges over all *maximal* winner sets W_i.

We claim that for any winner set W_i (not just maximal ones),

$$H_q(\mathcal{D}_i) = 0 \text{ for } q < j \cdot \left\lfloor \frac{dim(\mathcal{D}_i)}{m} \right\rfloor.$$

If L_i is empty, then $\Pi_i = \Pi$, and $\mathcal{F}(s_i) = \mathcal{D}_i$, and the claim follows because s^* is critical. Otherwise it follows because n is minimal. From Lemma 4.6,

$$H_q(\mathcal{C}_{U^*}(s^*)) = 0 \text{ for } q < j \cdot \left\lfloor \frac{dim(\mathcal{C}_{U^*}(s^*))}{m} \right\rfloor,$$

and $\mathcal{C}_{U^*}(s^*)$ satisfies $\wp$, a contradiction.

In the second case, there exists an object x whose set of pending operations $U_x \subseteq U$ is not weakly commuting. Each ordering of U_x defines a set of winners and losers as above. For each maximal winner set W_i (and minimal L_i), let Π_i be the protocol starting in s^* identical to Π except that the processes in L_i do not participate. Let $\mathcal{D}^{(i)}(s^*)$ be the reachable complex for Π_i from s^*, $\mathcal{D}_j^{(i)}(s^*)$ the reachable complex for Π_i in executions in which $P_j \in U^* - L_i$ takes the first step, and

$$\mathcal{D}_V^{(i)}(s^*) = \bigcap_{j \in V} \mathcal{D}_j^{(i)}(s^*).$$

Because the operations in U_x do not commute, each loser set L_i is non-empty, so Π_i is a protocol for strictly fewer than $n+1$ processes. Because n is minimal, each $\mathcal{D}_V^{(i)}$ satisfies $\wp$:

$$H_q(\mathcal{D}_V^{(i)}) = 0 \text{ for } q < j \cdot \left\lfloor \frac{dim(\mathcal{D}_V^{(i)})}{m} \right\rfloor.$$

Let $V_i = U^* - L_i$. The $\mathcal{D}_{V_i}^{(i)}(s^*)$ cover $\mathcal{C}_{U^*}(s^*)$ for maximal winner sets W_i, so Lemma 4.6 implies that

$$H_q(\mathcal{C}_{U^*}(s^*)) = 0 \text{ for } q < j \cdot \left\lfloor \frac{dim(\mathcal{D}_{V_i}^{(i)}(s^*))}{m} \right\rfloor,$$

and $\mathcal{C}_{U^*}(s^*)$ satisfies $\wp$, a contradiction. ∎

We are now ready to present the paper's second main result.

Theorem 4.8 *$\mathcal{C}^n(m, j)$ is simply connected with*

$$H_q(\mathcal{C}^n(m, j)) = 0 \text{ for } q < j \cdot \left\lfloor \frac{n}{m} \right\rfloor.$$

Proof: By Lemma 4.7, for all $U \subseteq W$,

$$H_q(\mathcal{C}_U(s)) = 0 \text{ for } q < j \cdot \left\lfloor \frac{dim(\mathcal{C}_U(s))}{m} \right\rfloor,$$

so by Lemma 4.6,

$$H_q(\mathcal{C}^n(m, j)) = 0 \text{ for } q < j \cdot \left\lfloor \frac{n}{m} \right\rfloor.$$

The proof that $\mathcal{C}^n(m, j)$ is simply connected is similar to the proof of Lemma 4.2, replacing the Mayer-Vietoris sequence with the with Siefert/Van Kampen theorem ([8, 4.12]). ∎

5 Conclusions

We have presented for the first time combinatorial properties of an asynchronous system in which processes communicate by objects more powerful than read/write registers. We have proved two general theorems about the solvability of set consensus.

The first theorem characterizes a wide class of protocols that *cannot* solve set consensus. Suppose a protocol has a full information complex $\mathcal{F}(U)$ when only the processes in U participate. If $\mathcal{F}(U)$ is connected for $|U| \geq c$, simply connected for $|U| \geq 2c$, and has $H_q(\mathcal{F}(U)) = 0$ for $q < |U|/c$, then that protocol cannot solve $(n+1, \lfloor n/c \rfloor)$-consensus. The proof of the result is general enough to point out a property that *any* solvable task should satisfy: there will always be at least one execution where the number of decided values is at least $\lfloor n/c \rfloor + 1$.

The second theorem states that the full information complex $\mathcal{C}^n(m,j)$ for any protocol using (m,j)-consensus objects is simply connected for $n > 2\lfloor m/j \rfloor$, with no "holes" in the lower dimensions:

$$H_q(\mathcal{F}^n(m,j)) = 0 \text{ for } q < j \cdot \left\lfloor \frac{n}{m} \right\rfloor .$$

One implication of these two theorems is that $(n+1, k)$-consensus is impossible if $k \leq j \cdot \lfloor n/m \rfloor$, which implies $n/k > m/j$.

References

[1] Hagit Attiya, Amotz Bar-Noy, Danny Dolev, David Peleg, and Rudiger Reischuk. Renaming in an asynchronous environment. *Journal of the ACM*, July 1990.

[2] 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*, May 1993.

[3] E. Borowsky and E. Gafni. The implication of the Borowsky-Gafni simulation on the set consensus hierarchy. Technical Report 930021, UCLA Computer Science Dept., 1993.

[4] S. Chaudhuri. Agreement is harder than consensus: set consensus problems in totally asynchronous systems. In *Proceedings of the Ninth Annual ACM Symosium on Principles of Distributed Computing*, pages 311–234, August 1990.

[5] S. Chaudhuri, M.P. Herlihy, N. Lynch, and M.R. Tuttle. A tight lower bound for k-set agreement. In *Proceedings of the 34th IEEE Symposium on Foundations of Computer Science*, October 1993.

[6] 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.

[7] P.J. Giblin. *Graphs, Surfaces, and Homology.* Chapman and Hill, London and New York, 1981. Second edition.

[8] M.J. Greenberg and J.R. Harper. *Algebraic Topology: A First Course.* Mathematics Lecture Notes Series. The Benjamin/Cummings Publishing Company, Reading MA, 1981.

[9] M. Henle. *Combinatorial Introduction to Topology.* W.H. Freeman and Company, San Francisco, 1979.

[10] M.P. Herlihy. Wait-free synchronization. *ACM Transactions on Programming Languages and Systems*, 13(1):123–149, January 1991.

[11] M.P. Herlihy and N. Shavit. The asynchronous computability theorem for t-resilient tasks. In *Proceedings of the 1993 ACM Symposium on Theory of Computing*, May 1993.

[12] M.P. Herlihy and N. Shavit. A simple constructive computability theorem for wait-free computation. In *Proceedings of the 1994 ACM Symposium on Theory of Computing*, May 1994.

[13] S. Lefschetz. *Introduction to Topology.* Princeton University Press, Princeton, New Jersey, 1949.

[14] M. Li, J. Tromp, and P.M. Vitányi. How to share concurrent wait-free variables. Technical Report CT-91-02, University of Amsterdam, Amsterdam, Netherlands, March 1991.

[15] 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*, May 1993.

[16] E.H. Spanier. *Algebraic Topology.* Springer-Verlag, New York, 1966.