The Asynchronous Computability Theorem for t-Resilient Tasks

(Preliminary Version)

Maurice Herlihy
Digital Equipment Corporation
Cambridge Research Laboratory
herlihy@crl.dec.com

Nir Shavit
Computer Science Department
Tel-Aviv University
shanir@math.tau.ac.il

Abstract

We give necessary and sufficient combinatorial conditions characterizing the computational tasks that can be solved by N asynchronous processes, up to t of which can fail by halting. The range of possible input and output values for an asynchronous task can be associated with a high-dimensional geometric structure called a simplicial complex. Our main theorem characterizes computability in terms of the topological properties of this complex. Most notably, a given task is computable only if it can be associated with a complex that is simply connected with trivial homology groups. In other words, the complex has "no holes!"

Applications of this characterization include the first impossibility results for several long-standing open problems in distributed computing, such as the "renaming" problem of Attiya et. al., the "k-set agreement" problem of Chaudhuri, and a generalization of the approximate agreement problem.

1 Introduction

A *decision task* is an input/output problem where N asynchronous processes start with input values, communicate either by shared memory or by message-passing, and halt with output values. In a fundamental paper in 1985, Fischer, Lynch, and Paterson [11] showed that there exists a simple Turing-computable task that is not computable in a message-passing system if even one process may fail by halting (or may be infinitely slow). Since then, a long and fruitful line of research has evolved, directed toward providing a computability theory for asynchronous computation. Such a theory could provide the designers of computer networks and multiprocessor architectures with mathematical tools for recognizing when certain protocols are impossible, for evaluating the power of alternative synchronization primitives, or for making explicit the assumptions needed to make a problem solvable.

In this paper, we give the first complete combinatorial characterization of the computational tasks solvable when up to t processes can fail by halting. Our results apply to shared memory when $1 \leq t \leq N$, and to message-passing systems when $1 \leq t < N/2$.[1] We show that one can associate with the ranges of possible inputs and outputs of an asynchronous task a high-dimensional geometric structure called a simplicial complex. Our main theorem characterizes computability in terms of the topological properties of this complex. The most interesting of these properties is that the complex may not have any holes: more formally, it must be simply connected with trivial homology groups.

The first step toward a systematic characterization of the asynchronously computable tasks was taken in 1988, when in a pioneering paper, Biran, Moran, and Zaks [6] gave a graph-theoretic characterization of the tasks that could be solved by a message-passing system in the presence of a single failure. Although the problem has received considerable attention since then, no one has succeeded in extending their approach to encompass more than a single failure.

In this paper, we solve this long-standing open

[1] Message-passing systems where $N/2 \leq t < N$ are uninteresting, because almost all tasks are easily seen to be impossible.

question. For any input/output problem, in shared memory or message passing, we give a necessary and sufficient condition for an algorithm to exist. Applications of this theorem include the first impossibility results for the *K-set agreement* problem of Chaudhuri [8], and a new, nearly-tight bound on the *renaming* problem of Attiya et. al. [4]. We also consider a generalization of the *approximate agreement* problem [5, 9, 10, 15, 22] to arbitrary point-sets.

Every data object can be assigned a *consensus number* [16], which is the maximum number of processes that can solve the consensus task [11] by applying operations to that object. An object's consensus number is a measure of its computational power: If X has consensus number n, and Y consensus number $m < n$, then in a system of n or more concurrent processes, one can construct a wait-free implementation of Y from X, but not vice-versa. It is natural to ask whether an object's consensus number completely characterizes its computational power. We show here for the first time that the answer is *no*. In particular, the set agreement task lies "between" read/write memory (consensus number one) and any object with consensus number two.

Using different techniques than those introduced here, Saks and Zaharoglou [23] have independently proved the impossibility of K-set agreement in the *wait-free* case where $K = t = N - 1$, and Borowsky and Gafni [7] have independently proved impossibility results for both renaming and K-set agreement.

2 Model

Informally, our model of computation consists of a collection of sequential threads of control called *processes* that communicate by reading and writing variables in shared memory. Processes are *sequential* — each process applies a sequence of operations to objects, alternately issuing an invocation and then receiving the associated response. We make no fairness assumptions about processes. A process can halt, or display arbitrary variations in speed. In particular, one process cannot tell whether another has halted or is just running very slowly. Formally, we model objects and processes as automata, using a simplified form of the I/O automata formalism of Lynch and Tuttle [21]. The details are omitted here for lack of space.

The shared memory consists of an array of single-reader/single-writer registers providing atomic *read* and *write* operations with the obvious semantics. From this single-reader/single-writer read/write memory, one can construct an *atomic snapshot* memory. Informally, this is a data structure partitioned into n segments s_i, each of which can be updated (written) by its "owner" process P_i, and all of which can be scanned (read) by any given process in one atomic operation. Each process P_i can thus perform an *update* operation on s_i, replacing all or part of the contents of s_i with a new value, or a *scan* operation on s, returning an "instantaneous" view of the contents of all segments of s. (More precise specifications and implementations of snapshot objects from single-reader/single-writer atomic registers can be found elsewhere [1, 2, 17, 20].)

A *vertex* v is a pair consisting of a value, *value*(v), and a process identifier, *id*(v). An n-dimensional *simplex* (or n-simplex) is a set of $n+1$ vertexes labeled with distinct process identifiers. We usually use superscripts to indicate the dimension of a simplex. S^m is a *subsimplex* of S^n, written $S^m \subset S^n$, if the former's set of vertexes is a subset of the latter's. Let *ids*(S^n) denote the set of $n+1$ distinct process identifiers in the vertexes of S^n, and *values*(S^n) the multiset of values. An n-simplex has a geometric interpretation as the convex hull of a set of $n+1$ affinely-independent points in Euclidian space.

A *simplicial complex* (or complex) is a set of simplexes $\mathcal{C}$ satisfying the following property: If $S^m \in \mathcal{C}$ and $S^\ell \subset S^m$ then $S^\ell \in \mathcal{C}$. *Subcomplexes* are defined in the obvious way. Define the *dimension* of a complex to be the highest dimension of any simplex it contains. An n-complex is *pure* if every simplex it contains is a subsimplex of some n-simplex. All complexes considered in this paper are pure. As with simplexes, we usually use superscripts to indicate the dimension of a complex.

An $(n+1)$-process *decision task* $\langle \mathcal{I}^n, \mathcal{O}^n, \Delta \rangle$ is given by an *input* complex $\mathcal{I}^n$, an *output* complex $\mathcal{O}^n$, and a recursive (computable) map Δ carrying each m-simplex of $\mathcal{I}^n$, where $0 \leq m \leq n$, to an m-subcomplex of $\mathcal{O}^n$ in a way that (1) preserves process identifiers: for all S^m in $\mathcal{I}^n$, $ids(S^m) = ids(\Delta(S^m))$, and (2) is coherent: if $S^\ell \subset S^m$, then $\Delta(S^\ell) \subset \Delta(S^m)$. Simplexes in $\mathcal{I}^n$ and $\mathcal{O}^n$ are called *input* and *output simplexes.*

Informally, each input simplex S^m corresponds to a set of $m+1$ possible input values, and the input complex is the collection of all legal sets of inputs. The complex $\Delta(S^m)$ is the collection of all legal output configurations in response to the set of input values S^m. A *solo* execution by a set of processes U is one where all processes in U complete the protocol before any other process takes a step. Intuitively, $\Delta(S^m)$ for $m < n$ is the set of possible outputs of solo executions by the processes in *ids*(S^m).

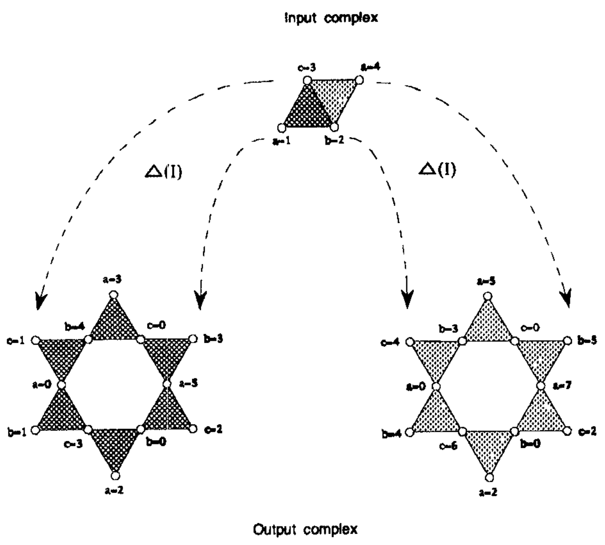

Figure 1: The FETCH&ADD task.

Perhaps the simplest decision task is *consensus* [11]. Each process starts with a binary input value and chooses a binary output value. All output values must agree, and the output value must have been someone's input. The input complex consists of simplexes of the form $(\langle P_0, b_0\rangle, \ldots, \langle P_n, b_n\rangle)$, where the b_i independently range over $\{0, 1\}$. (This complex is topologically equivalent to an n-sphere.) The output complex consists of two disjoint n-simplexes, corresponding to decision values 0 and 1.

Figure 1 shows two input simplexes and their corresponding output complexes for the *fetch-and-add* task, in which each of $n+1$ processes atomically adds an integer input to a shared register, initially zero, and returns the register's previous contents. The input and output complexes are infinite, although the output complex corresponding to any specific input simplex is finite. Note that vertexes with the same labels are the same and should be mentally "glued together." They are drawn as distinct only for legibility. Figure 2 shows a three-process *renaming* task [4] using four names. (This output complex is topologically equivalent to a Klein bottle.)

A protocol *solves* a decision task $\langle \mathcal{I}^n, \mathcal{O}^n, \Delta\rangle$ in an execution if the following conditions hold. Let $u_0, \ldots, u_n$ be the input values of $P_0, \ldots, P_n$, and let $P_i, \ldots, P_j$ be the processes that finish the protocol with values $v_i, \ldots, v_j$. We require that (1) no process takes an infinite number of steps without choosing an output value, and (2) the simplex $(S^n = \langle P_0, u_0\rangle, \ldots \langle P_n, u_n\rangle)$ is a simplex of $\mathcal{I}^n$ and $(\langle P_i, v_i\rangle, \ldots \langle P_j, v_j\rangle)$ is a simplex in $\Delta(S^n)$. An execution is *t-faulty* if at most t processes fail. A protocol is *t-resilient* if it solves a decision task in every t-faulty execution. A protocol is *wait-free* if it is $(n-1)$-resilient.

Many complexes of interest have a simple but important topological property: they have no "holes". We now introduce the formal machinery needed to make this notion precise. A more complete discussion may be found in any standard text on algebraic topology (e.g., [13, 14, 19, 24]).

An *orientation* for a simplex is a collection of orderings for the vertexes, consisting of one particular ordering and all even permutations of it. An *oriented n-simplex* is an n-simplex S^n together with an orientation. By convention, $(v_0, \ldots, v_n)$ denotes the oriented n-simplex with vertexes $v_0, \ldots, v_n$ oriented in the even permutations of $0, \ldots, n$. An *oriented complex* is a simplicial complex in which each simplex is provided with an orientation.

Let $\{S^d_0, \ldots, S^d_\ell\}$ be the set of oriented d-simplexes in an oriented complex C^n. A *d-chain* is a formal sum of the form $\sum_{i=0}^{\ell} \lambda_i \cdot S^d_i$, where λ_i is an integer.[2] When writing chains, we typically omit d-simplexes with zero coefficients, unless they are all zero, when we simply write 0. We also write $1 \cdot S^d$ as S^d and $-1 \cdot S^d$ as $-S^d$. We identify $-S^d$ as S^d with the opposite orientation.

Let $S^d = (v_0, \ldots, v_d)$ be an oriented d-simplex.

[2]Alternatively, the d-th *chain group* of C^n is the free abelian group on the set $\{S^d_0, \ldots, S^d_\ell\}$.

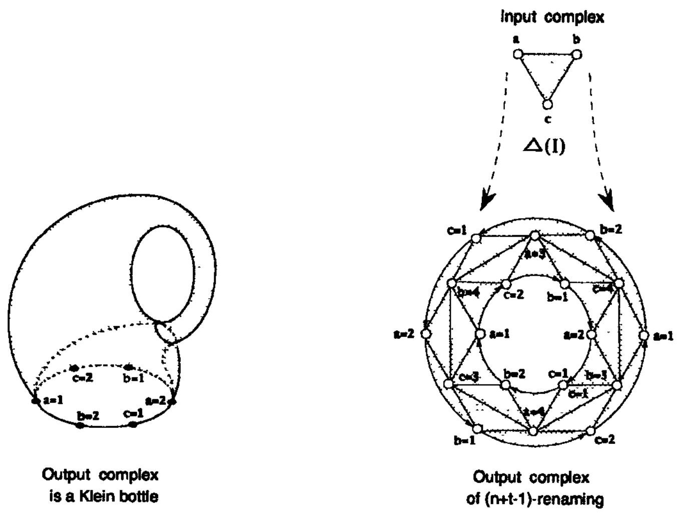
Input complex
a
b
c
Δ(I)
Output complex is a Klein bottle
Output complex of (n+t-1)-renaming

Figure 2: The RENAMING task.

Define $face_i(S^d)$, the i^{th} *face* of S^d, to be the $(d-1)$-simplex $(v_0, \dots, \hat{v}_i, \dots, v_d)$, where circumflex denotes omission. Define the *boundary* operator, ∂, on simplexes to be the chain:

$$\partial S^d = \sum_{i=0}^{d} (-1)^i \cdot face_i(S^d)$$

The boundary operator extends additively to chains: $\partial(C_1^d + C_2^d) = \partial C_1^d + \partial C_2^d$. It is easily verified that $\partial\partial C^d = 0$ for every d-chain C^d.

A d-chain C^d is a *d-cycle* if $\partial C^d = \emptyset$, and it is a *d-boundary* if there exists a C^{d+1} such that $\partial C^{d+1} = C^d$. Every boundary in a complex is a cycle, but not necessarily vice-versa.

Definition 2.1 The set of d-cycles for a complex define an abelian group, called the d-th *homology group* for that complex. If every d-cycle is a d-boundary, then the d-th homology group is the trivial (single-element) group. For brevity, we say a complex C^n has *trivial d-th homology* if its d-th homology group is trivial, and it has *trivial homology* if it has trivial d-th homology for $1 \leq d \leq n$.

Informally, if C^n has trivial homology, then C^n has no "holes" of dimension d: every d-cycle is a d-boundary, and can therefore be "filled in." For example, the fetch-and-add complex shown above has a non-trivial first homology group because the cycle bounding the hole is not a boundary.

A *edge path* in a complex C is a sequence of vertexes $v_0, \dots, v_\ell$ such that each successive pair v_i, v_{i+1} (not necessarily distinct) lie on a simplex. C^n is *path connected* if every pair of vertexes is connected by an edge path. An edge path is an *edge loop* if $v_0 = v_1$. The vertex v_0 is called the *base point* of the loop. A loop is *trivial* if it consists only of its base point. A loop is *null-homotopic* if it can be reduced to a trivial loop by applying the following kinds of transformations: (1) if v_i, v_{i+1}, v_{i+2} lie on a common simplex, then the subsequence v_i, v_{i+1}, v_{i+2} can be replaced with the subsequence v_i, v_{i+2}, and vice-versa, and (2) the subsequence v_i, v_i can be replaced by v_i, and vice-versa.

Definition 2.2 A complex C is *simply connected* if it is path connected and all edge loops are null-homotopic.

The set of edge loops in a complex with a fixed base point define a group called the *fundamental group* of the complex. If a complex is simply connected, then its fundamental group is trivial. It is a standard result that any simply-connected complex has trivial first homology ([19, Th. 18.1] or [14, Ch. 12]) (The converse, however, is false [14, p.150].)

Let $\mathcal{A}^n$ and $\mathcal{B}^n$ be complexes. A *simplicial map* $\mu : \mathcal{A}^n \rightarrow \mathcal{B}^n$ carries vertexes of $\mathcal{A}^n$ to vertexes of $\mathcal{B}^n$ such that every simplex of $\mathcal{A}^n$ maps to a simplex of $\mathcal{B}^n$ (possibly of lesser dimension). Simplicial maps preserve cycles and boundaries: if C is a cycle, so is $\mu(C)$, and if $C = \partial(D)$, then $\mu(C) = \partial\mu(D)$. We will exploit this property of simplicial maps to derive a number of impossibility results for specific problems.

$$\begin{tikzcd}
\mathcal{F}^n \arrow[r, "\delta"] & \mathcal{O}^n \\
S^n \arrow[u, "\Phi"] \arrow[ur, "\Delta"'] &
\end{tikzcd}$$

Figure 3: Commutative Diagram for Task Solution

3 Full Information Protocols

Informally, a full-information protocol is one in which each process repeatedly exchanges its entire history with each of the others until it has "seen enough" to choose a decision value. Full-information protocols have a regular structure that makes them easier to analyze than arbitrary protocols.

Definition 3.1 Following Frederickson and Lynch [12], an *S-expression* is either the symbol $\perp$, an input value v, or a list $(v_0 \dots v_n)$ where each v_i is an S-expression. We use $s[i]$ to denote the i-th component of a compound S-expression s.

Definition 3.2 The *full-information complex* $\mathcal{F}^n$ for an input simplex S^n is defined as follows. Each vertex is labeled with an S-expression. Each n-simplex R^n of $\mathcal{F}^n$ is constructed inductively by a sequence $R^n_0, \dots, R^n_\ell$ where $R^n_0 = (\perp \dots \perp)$, $R^n_\ell = R^n$, and each R^n_k is constructed from R^n_{k-1} by replacing some i-th component in one of two ways: (1) if $R^n_{k-1}[i] = \perp$, then $R^n_k[i] = v_i$, where v_i is P_i's input value in S^n, or (2) if $R^n_{k-1}[i] \neq \perp$, then $R^n_k[i] = R^n_j$, where $j < k$ and $R^n_j[i] = R^n_{k-1}[i]$.

Note that the full-information complex is infinite.

A t-resilient protocol is a *full-information protocol* if it has the form shown in Figure 3. Let δ be a map from S-expressions to output values. An S-expression s is a *halting state* if $\delta(s)$ is defined. The processes share an array V. P_i starts by writing its input value to $V[i]$, and then it executes a sequence of *rounds*. At round i, it repeatedly scans the array until at least $n - t + 1$ processes have either (1) written out a halting state, or (2) completed a round greater than or equal to $i - 1$. A process executing this loop is said to be *waiting at a barrier*. The process then scans the shared array V, creating local copy v, and updates $V[i]$ to $(v[0] \dots v[n])$. If $\delta(v)$ is defined, it returns that value, and otherwise it starts round $i + 1$.

Lemma 3.1 *The output simplexes computed by any execution of a full-information protocol lie in* $\mathcal{F}^n$.

```
V[i] := input
round := 0
loop
  round := round+1
  /* barrier step */
  repeat {
    v := scan(V)
    } until (at least n-1 processes have
            either halted or reached round-1)
   /* scan step */
   v := scan(V)
   /* update step */
   V[i] := (P_i, v)
   /* check for halting state */
   if (delta(v) is defined)
     then return delta(v)
```

Figure 4: A Full-Information Protocol

Theorem 3.2 *A decision task* $\langle \mathcal{I}^n, \mathcal{O}^n, \Delta \rangle$ *is t-solvable if and only if there exists a t-resilient full-information protocol* $\langle \mathcal{I}^n, \mathcal{F}^n, \Phi \rangle$*, and a simplicial map δ, called a* decision map*, such that the diagram in Figure 2 commutes in the following sense: for every input simplex S^m, $n - t \leq m \leq n$,* $\delta : \Phi(S^m) \to \Delta(S^m)$.

To show impossibility, we will focus on the decision map δ. This decision map is a simplicial map from the full-information complex to the task's output complex. Simplicial maps preserve certain basic topological properties, and therefore we will demonstrate the impossibility of constructing a decision map by showing that any full-information complex is topologically incompatible with the tasks' output complexes.

The theorem also gives a sufficient condition for solvability. The condition is existential, in the sense that one cannot take an arbitrary task specification and construct a matching implementation. Indeed, by adapting a theorem of Toueg and Jayanti [18], one can show that the problem of deciding whether an arbitrary decision task has a t-resilient implementation is undecidable. Of course, if all complexes are finite, one can always find a decision map (if one exists) by enumeration and testing. Notice that the sufficient condition of Biran, Moran and Zaks [6] for 1-resilient tasks is similar in nature: a task is 1-solvable if and only if an associated graph satisfies certain properties, but there is no general method for determining whether an arbitrary graph satisfies those properties.

In a *message-passing* system, processors communicate by sending and receiving asynchronous messages. A simulation of a message passing algorithm with $t < n/2$ faults in shared memory with $t < n/2$ faults is an elementary exercise. Conversely, Attiya, Bar-Noy, and Dolev [3, Th. 9] have shown how to translate any wait-free shared-memory algorithm into a message-passing algorithm with $t < n/2$ faults. Consequently, the Asynchronous Computability Theorem holds for message-passing systems when $t < n/2$.

4 Topological Properties

We new derive some basic topological properties of t-resilient protocols. First, we show that "*a full-information protocol has no holes.*" More precisely, if Φ is a t-resilient full-information protocol, then for every input simplex S^q, $n-t \leq q \leq n$, $\Phi(S^q)$ is simply connected with trivial homology. Second, we show that $\Phi(S^q)$ must include a subcomplex with a particular regular geometric structure.

Definition 4.1 A property $\wp$ is *simplicial* if it satisfies the following conditions. (1) $\wp$ holds for any single n-simplex, and (2) If $\mathcal{A}^n$ and $\mathcal{B}^n$ are intersecting complexes such that $\wp$ holds for $\mathcal{A}^n$, $\mathcal{B}^n$, and $\mathcal{A}^n \cap \mathcal{B}^n$, then $\wp$ holds for $\mathcal{A}^n \cup \mathcal{B}^n$.

Being simply connected is a simplicial property (from the *Seifert/Van Kampen* Theorem [14, 4.12]), as is having trivial homology (from the Mayer-Vietoris sequence).

For the remainder of this section, let $n-t \leq q \leq n$. We will now show that for every input simplex S^q, and any simplicial property $\wp$, $\Phi(S^q)$ satisfies $\wp$.

Definition 4.2 An output simplex R^q is *reachable* from protocol state s if there is some execution starting from s in which each process in $ids(R^q)$ chooses its corresponding value from R^q. The *reachable complex* from s is the complex of reachable simplexes from s.

A protocol state is *critical* for a property $\wp$ if $\wp$ does not hold, but every group of $n-t+1$ processes includes at least one process whose next step will leave the protocol in a state where $\wp$ henceforth holds. Such a process is called a *critical process.* A critical state is *maximal* if every process is either critical, waiting at a barrier, or halted with a decision value.

Lemma 4.1 *Let $\wp$ be a property that must eventually become true in every t-faulty execution, and let s be a state where $\wp$ does not hold. Any full-information protocol has a maximal critical state for $\wp$.*

Lemma 4.2 *In executions where zero failures occur, the reachable complex is a single simplex.*

Our proof strategy is the following. Since the reachable complex eventually shrinks to a single simplex, it eventually henceforth satisfies $\wp$. We run the protocol to a maximal critical state for $\wp$, and we derive a contradiction in the style of Fischer, Lynch, and Paterson [11], using an analysis of the possible interactions among the pending *update* and *scan* operations to show that the reachable complex could not have violated $\wp$ to begin with.

More formally, let S^q be an input simplex and $\wp$ any simplicial property. We can run any protocol to a maximal critical state for $\wp$. Let C be the set of critical processes. Let U be a subset of m critical processes in C. We define $\mathcal{C}_U^q$ to be the reachable complex immediately after each process in U takes a step. For brevity, $\mathcal{C}_i^q = \mathcal{C}_{\{P_i\}}^q$, and $\mathcal{C}^q = \cup \mathcal{C}_i^q$ for $P_i \in C$. We will show that $\mathcal{C}^q$ is really the reachable complex, and then that $\mathcal{C}^q$ satisfies $\wp$, and hence the desired contradiction.

Lemma 4.3 *Let C be the set of critical processes in a maximal critical state and let*

$$\mathcal{C}^q = \bigcup_{i \in C} \mathcal{C}_i^q.$$

Then $\mathcal{C}^q$ is the reachable complex in the maximal critical state.

Lemma 4.4 *In a maximal critical state, let W be a set of critical processes about to perform operations such that each pair of operations commutes. (For example, all scans or all updates.) Let $U_0, \ldots, U_\ell$ be subsets of W such that each $|U_i| = m$ and for each distinct U_i and U_j, $\{P_i\} = U_i - U_j$. Let*

$$\mathcal{D}^q = \bigcup_{i=0}^{\ell} \mathcal{C}_{U_i}^q.$$

We claim that $\mathcal{D}^q$ satisfies $\wp$.

Proof: We argue by induction on ℓ and reverse induction on m.

We now prove the base case for m. When $m = |W|$, the only such complex is $\mathcal{C}_W^q$, which satisfies the lemma because the current state is critical. We assume the result for $m+1$ and $1 \leq \ell \leq \binom{|W|}{m}$ and prove it for m and $\ell = 0$. The only such complex is $\mathcal{C}^q_{U_0}$, which also satisfies the lemma because the current state is critical.

We assume the result for $(\ell - 1)$ sets of size m, and will prove it for ℓ sets.

Let $\mathcal{D}^q = \cup_{i=0}^{\ell}\mathcal{C}^q_{U_i}$, where $U_0, \ldots, U_\ell$ are process sets of size m satisfying the conditions above. $\mathcal{D}^q = \mathcal{A}^q \cup \mathcal{B}^q$ where $\mathcal{A}^q = \mathcal{C}^q_{U_\ell}$ and $\mathcal{B}^q = \cup_{i=0}^{\ell-1}\mathcal{C}^q_{U_i}$.

Claim 4.5 *For any sets U_i and U_j, $\mathcal{C}^q_{U_i} \cap \mathcal{C}^q_{U_j} = \mathcal{C}^q_{U_i \cup U_j}$.*

Proof: We first show that $\mathcal{C}^q_{U_i \cup U_j} \subseteq \mathcal{C}^q_{U_i} \cap \mathcal{C}^q_{U_j}$. By definition, each simplex in $\mathcal{C}^q_{U_i \cup U_j}$ is reachable in some execution where every process in $U_i \cup U_j$ takes a step before any other process takes a step. The operations commute, so we can reorder them so every process in U_i moves before any process in U_j, yielding $\mathcal{C}^q_{U_i \cup U_j} \subset \mathcal{C}^q_{U_i}$. By a symmetric argument, $\mathcal{C}^q_{U_i \cup U_j} \subset \mathcal{C}^q_{U_j}$, yielding $\mathcal{C}^q_{U_i \cup U_j} \subseteq \mathcal{C}^q_{U_i} \cap \mathcal{C}^q_{U_j}$.

Conversely, each simplex in $\mathcal{C}^q_{U_i} \cap \mathcal{C}^q_{U_j}$ is reachable by at least two executions: one where every process in U_i takes a step before any other process, and one where every process in U_j takes a step before any other process. Moreover, these executions must be indistinguishable to each process, and therefore $\mathcal{C}^q_{U_i} \cap \mathcal{C}^q_{U_j} \subseteq \mathcal{C}^q_{U_i \cup U_j}$. ■

Therefore,

$$\mathcal{A}^q \cap \mathcal{B}^q = \bigcup_{i=0}^{\ell-1}(\mathcal{C}^q_{U_i} \cap \mathcal{C}^q_{U_\ell}) = \bigcup_{i=0}^{\ell-1}\mathcal{C}^q_{U_i \cup U_\ell}$$

We now prove the induction step for ℓ. Let $V_i = U_i \cup U_\ell$.

$$V_i = (U_i - U_\ell) \cup (U_\ell - U_i) \cup (U_i \cap U_\ell).$$

The first two terms have size one, and the third $m - 1$, so $|V_i| = m + 1$.

$$V_i - V_j = (U_i \cup U_\ell) - (U_j \cup U_\ell) = \{P_i\}$$

The $V_0, \ldots, V_{\ell-1}$ thus satisfy the conditions for the induction hypothesis for $m+1$, and therefore $\mathcal{A}^q \cap \mathcal{B}^q$ satisfies $\wp$. Moreover, $\mathcal{A}^q$ satisfies $\wp$ by construction, and so does $\mathcal{B}^q$ by the induction hypothesis for $\ell - 1$. Because $\wp$ is simplicial, $\mathcal{A}^q \cup \mathcal{B}^q$ also satisfies $\wp$. ■

Corollary 4.6 *$\bigcup_{i \in W} \mathcal{C}^q_i$ satisfies $\wp$.*

We have just shown that if the critical processes are poised to perform operations that commute, say, all scans or all updates, then the simplicial property $\wp$ must already hold, contradicting our assumptions. Therefore, in any maximal critical state for $\wp$, some processes must be about to scan, and others about to update. We now show that this assumption also leads to a contradiction.

As before, let $\mathcal{C}^q_i$ be the reachable complex immediately after critical process P_i moves.

Lemma 4.7 *Let X be an arbitrary set of critical processes in any maximal critical state for $\wp$. We claim that $\mathcal{A}^q = \bigcup_{i \in X} \mathcal{C}^q_i$ satisfies $\wp$.*

Proof: Divide X into two sets, R and W, such that every process in R is about to scan the shared variables, and each process in W is about to update its variable. We argue by double induction, first on $m = |X| = |R| + |W|$, the total number of critical processes, and then on $\ell = |R|$, the number of critical processes about to scan.

When $m = 0$, $\mathcal{A}^q$ is empty, so assume the result for $m-1$ processes and arbitrary ℓ. For m processes and $\ell = 0$, the result follows from Corollary 4.6. We now consider the induction step for ℓ.

Assume $\mathcal{A}^q$ satisfies $\wp$, where $|R \cup W| = m$, $|R| = \ell - 1$, and $P_r \notin X$ is a critical process about to scan. We now show that $\mathcal{C}^q_r \cup \mathcal{A}^q$ satisfies $\wp$. If no failures are possible, then the result follows from Lemma 4.2. Otherwise, notice that the simplexes in $\mathcal{C}^q_r \cap \mathcal{A}^q$ are reachable only in executions where all processes other than P_r finish their last scans before P_r updates its variable. The simplexes reachable by such executions form the reachable complex for an $(m - 1)$-process full-information protocol with one fewer failure. By the induction hypothesis for $m-1$, $\mathcal{C}^q_r \cap \mathcal{A}^q$ satisfies $\wp$. Moreover, $\mathcal{C}^q_r$ satisfies $\wp$ because P_r is critical, and so does $\mathcal{A}^q$ by the induction hypothesis for $\ell - 1$. We have shown that $\wp$ is satisfied by $\mathcal{C}^q_r$, $\mathcal{A}^q$, and $\mathcal{C}^q_r \cap \mathcal{A}^q$, and the result follows because $\wp$ is a simplicial property. ■

Theorem 4.8 *If $\langle \mathcal{I}^n, \mathcal{F}^n, \Phi\rangle$ is a t-resilient full-information protocol, then for every input simplex S^q where $q \geq n - t$, $\Phi(S^q)$ satisfies $\wp$.*

Corollary 4.9 *If $\langle \mathcal{I}^n, \mathcal{F}^n, \Phi\rangle$ is a t-resilient full-information protocol, then for every input simplex S^q where $q \geq n - t$, $\Phi(S^q)$ is simply connected with trivial homology.*

We now show that full-information complexes have a regular geometric structure that can be exploited to prove a variety of impossibility results. In particular, any n-dimensional full-information complex includes a subcomplex, called a *span*, that "looks" like a t-simplex that has been subdivided into a set of smaller t-simplexes in a regular way. (Such a subdivision is called a *triangulation.*) This span has a nice recursive structure: if the span is a triangulation of S^t then each vertex of S^t corresponds to a solo execution of some set of $n-t$ processes, all vertexes along an edge of S^{t-1} to solo executions by some set of $n-t+1$ processes, and so on. Our notion of a span is essentially a generalization of the spanning trees of Biran, Moran, and Zaks [6] when $t = 1$.

A span is a *pseudomanifold*: every $(n-1)$ simplex is a subsimplex of either one or two n-simplexes. The subcomplex $\tilde{\mathcal{A}}^n$ of $\mathcal{A}^n$ generated by the $(n-1)$-simplexes contained in exactly one n-simplex is called the *boundary subcomplex*. Once we prove that spans exist, then we can exploit the many theorems of algebraic topology (such as Sperner's Lemma [19, Lemma 5.5]) that apply to pseudomanifolds.

Definition 4.3 The *barycentric subdivision* $bary(S^n)$ of input simplex S^n is the complex whose vertexes are the subsimplexes in S^n. A simplex $(S_0, \ldots, S_n)$ (where the S_i are subsimplexes of S^n) is in $bary(S^n)$ if for some permutation $i_0, \ldots, i_n$ of $0, \ldots n$, $S_{i_0} \subset \ldots \subset S_{i_n} \subset S^n$. Let $bary^\ell(S^n)$ denote the result of ℓ successive barycentric subdivisions.

Let T be the set of $n-t$ processes with indexes greater than $t+1$, S_T^{n-t-1} the subsimplex of S^n with process ids in T, and S^ℓ a subsimplex of S^n containing no process ids from T.

Definition 4.4 A *span* for S^ℓ is a simplicial map $\sigma : bary^r(S^\ell) \to \Phi(S^\ell \cup S_T^{n-t-1})$ for some $r \geq 0$, such that if $\ell > 0$, then for $0 \leq i \leq t$, σ restricted to $bary^r(face_i(S^\ell))$ is a span for $face_i(S^\ell)$.

Lemma 4.10 *If S^ℓ is an input simplex with no ids from T, then S^ℓ has a span.*

Proof: (*Sketch.*) We make use of the following classical theorems of algebraic topology. (1) Any complex has a "realization" as a point set in Euclidian space [19, Th. 14.2]. (2) The Hurewicz Isomorphism Theorem ([19, Th. 23.1]) implies that if a complex $\mathcal{A}^n$ is simply connected with trivial homology, then any continuous map from the ℓ-sphere to $\mathcal{A}^n$ can be extended to a continuous map of the $(\ell+1)$-disk to $\mathcal{A}^n$. (3) The Simplicial Approximation Theorem ([13, Th. 5.11]) states that any continuous map from $\mathcal{A}^n$ to $\mathcal{B}^n$ can be "approximated" by a simplicial map from $bary^r(\mathcal{A}^n)$ to $\mathcal{B}^n$, for some $r \geq 0$.

Spans are constructed by induction on ℓ. We (temporarily) treat all simplexes as subsets of Euclidian space (allowing maps to be continuous). If $\mathcal{A}$ is a complex, let $|\mathcal{A}|$ denote the associated point set. The spans for the faces of S^ℓ define a continuous map $\phi : |bary^r(\tilde{S}^\ell)| \to |\Phi(S^\ell \cup S_T^{n-t-1})|$. Because $|bary^r(\tilde{S}^\ell)|$ is homeomorphic to an $(\ell-1)$-sphere, and $|\Phi(S^\ell \cup S_T^{n-t-1})|$ is simply connected with trivial homology, the Hurewicz Isomorphism Theorem implies that ϕ can be extended to a continuous map ϕ' from the ℓ-disk $|bary^r(S^\ell)|$ to $|\Phi(S^\ell \cup S_T^{n-t-1})|$. By the Simplicial Approximation Theorem, there exists a simplicial map σ carrying $bary^R(S^\ell)$ for some $R \geq r$, to $\Phi(S^\ell \cup S_T^{n-t-1})$ that agrees with ϕ on vertexes in $bary^r(\tilde{S}^\ell)$. The map σ is a span for S^ℓ. ■

5 Impossibility Results

5.1 Set Agreement

In the *K-set agreement task*, each of $n+1$ processes starts with an arbitrary input value in a private register, and halts after returning an output value. In every t-resilient execution the output values must satisfy the following conditions: *Validity*: every output value is some process's input value, and *Consistency*: the set of output values has cardinality at most K. This problem was first proposed by Chaudhuri [8] in 1989, along with a conjecture that it has no t-resilient solution for $K \leq t$. We now show that this conjecture is correct.

Definition 5.1 The *carrier* of a vertex v in $bary^r(S^m)$ is the smallest-dimensional subsimplex $S^\ell \subset S^m$ such that $v \in bary^r(S^\ell)$.

Lemma 5.1 (Sperner's Lemma) *If χ is a map sending each vertex v of $bary^r(S^m)$ to a vertex in its carrier, then there is at least one m-simplex $R^m = \{r_0, \ldots, r_m\}$ in $bary^r(S^m)$ such that the $\chi(r_i)$ are all distinct.*

A protocol for the t-set agreement task is *canonical* if, in every t-faulty execution, no process chooses the input from the $n-t$ processes with the highest ids. Any t-set agreement protocol Φ can easily be transformed into a canonical protocol Φ'. Each process tags its input with with its process id, runs Φ, writes its decision value (including the originator's id) to a shared array *prefer*, and waits until $n-t+1$ processes have written their values. The process then decides the value from *prefer* tagged with the least id.

Theorem 5.2 *The K-set agreement task has no t-faulty solution for $K \leq t$.*

Proof: It is enough to show that no canonical t-resilient t-set agreement protocol exists. By way of contradiction, let Φ be such a protocol. Pick an input simplex $S^n = S^t \cup S_T^{n-t-1}$ in which each process P_i has a distinct input v_i. By Lemma 4.10, there exists a span $\sigma : bary^r(S^t) \rightarrow \Phi(S^t \cup S_T^{n-t-1})$. Let $S^t = (s_0, \ldots, s_t)$. Define $\chi : bary^r(S^t) \rightarrow S^t$ as follows: $\chi(v) = s_i$, where $value(\sigma(v)) = v_i$. Each $value(v)$ is chosen by some process in a solo execution by the processes in $ids(S^t) \cup T$. Because the protocol is canonical, however, $value(v)$ cannot be an input value for any process in T, so χ must carry each vertex v to a vertex in its carrier. The map χ is thus a Sperner labeling for S^t, and Sperner's Lemma implies that there exists a simplex in $bary^r(S^t)$ whose vertexes are mapped to all $t+1$ inputs. This simplex corresponds to an execution that yields $t+1$ distinct decision values, a contradiction. ∎

It is easily shown that one cannot solve two-process consensus using an arbitrary object that solves 2-set agreement. It follows that the computational power of set agreement lies "between" that of read/write memory and any object with consensus number two.

Corollary 5.3 *An object's computational power is* not *completely characterized by its consensus number.*

5.2 Renaming

Another open problem that has attracted considerable attention is the *renaming* problem of Attiya *et al.* [4]. Each process is given a unique *input name* taken from a range $0 \ldots N$ where $N \gg n$. Each process then chooses a unique *output name* taken from a smaller range $0 \ldots K$. To rule out trivial solutions (such as P_i chooses output name i), we require that the name chosen by a process depend only on its input name and the execution, and not on the process's id.

For $n+1$ processes, it is known that the renaming task has a t-resilient solution for $n+t+1$ or more output names, and none for $n+2$ or fewer [4]. We now narrow this gap by showing that there is no solution for $n+t-1$ or fewer output names, leaving open the question whether there is a solution for exactly $n+t$ names.

Theorem 5.4 *There is no t-resilient renaming protocol for $n+1$ processes, $2n-1$ or more input names, and $n+t-1$ or fewer output names.*

(Because of space limitations, we summarize the proof for the wait-free case.) Let $\mathcal{E}^\ell$ be the complex generated by a single ℓ-simplex, where vertexes are labeled with $0, \ldots, \ell$.

Theorem 5.5 *[Spanier, 8.D.2] Let $\phi : \mathcal{A}^n \rightarrow \mathcal{E}^n$ be a simplicial map. For $0 \leq i \leq n$, the number of simplexes of $\tilde{\mathcal{A}}^n$ mapped to $face_i(\mathcal{E}^n)$ has the same parity as the number of simplexes of $\mathcal{A}^n$ mapped to $\mathcal{E}^n$.*

Let Φ be a wait-free renaming protocol where input names range from 0 to at least $2n-2$, and output names from 0 to $2n-2$. Pick $n+1$ input names so that all processes choose output names of the same parity in solo executions. Let $\sigma : bary^r(S^\ell) \rightarrow \Phi(S^\ell)$ for $r \geq 0$ be a span. If ϕ is a function on $bary^r(S^\ell)$, $\phi(P_i, s_i)$ is shorthand for $\phi(v)$, where $\sigma(v) = \langle P_i, s_i \rangle$. Define $\phi : bary^r(S^\ell) \rightarrow \mathcal{E}^{\ell+1}$ as follows: $\phi(P_i, s_i)$ is the vertex $(i + (name(s_i) \bmod 2) \pmod{\ell+2})$.

Lemma 5.6 *The number of simplexes of $bary^r(S^\ell)$ mapped to $face_i(\mathcal{E}^{\ell+1})$ is odd for $i = \ell+1$, and even otherwise.*

The proof is an inductive argument based on Theorem 5.5.

When $\ell = n$, ϕ sends the vertexes of at least one simplex T^n of $bary^r(S^n)$ to distinct vertexes of $\mathcal{E}^n$, implying that the names at vertexes of T^n are either all even or all odd. Since there $n+1$ processes but only $2n-1$ names, we have a contradiction.

5.3 Approximate Agreement

Let K be a point set in Euclidian space $\mathbf{R}^\ell$, and let $hull(K)$ be the convex hull of K. In the *generalized approximate agreement* task, each P_i starts with an input x_i in K, and halts with an output y_i in K satisfying (1) *agreement*: all y_i lie within some fixed ϵ of one another, and (2) *validity*: all y_i lie within the convex hull of the x_i. This problem has been studied in the special case where $K = \mathbf{R}$ in the Byzantine [9, 10, 22] and fail-stop models [5, 15]. Here we consider only the wait-free case.

Definition 5.2 K has a *hole* of radius r around point x if x is in $hull(K)$, and that no point of K lies within distance r of x.

Theorem 5.7 *The generalized approximate agreement task has no wait-free solution if K has a hole of radius ϵ.*

References

[1] Y. Afek, H. Attiya, D. Dolev, E. Gafni, M. Merritt, and N. Shavit. Atomic snapshots. Ninth ACM Symposium on Principles of Distributed Computing, 1990.

[2] J. Anderson. Composite registers. In *Proceedings of the 9th ACM Symposium on Principles of Distributed Computing*, pages 15–30, August 1990.

[3] H. Attiya, A. Bar-Noy, and D. Dolev. Sharing memory robustly in message-passing systems. In *Proceedings of the 9th Annual ACM Symposium on Principles of Distributed Computing*, pages 377–408, August 1990.

[4] H. Attiya, A. Bar-Noy, D. Dolev, D. Koller, D. Peleg, and R. Reischuk. Achievable cases in an asynchronous environment. In *Proceedings of the 28th IEEE Symposium on Foundations of Computer Science*, pages 337–346, October 1987.

[5] H. Attiya, N. Lynch, and N. Shavit. Are wait-free algorithms fast? In *Proceedings of the 31st Annual Symposium on the Foundations of Computer Science*, October 1990.

[6] O. Biran, S. Moran, and S. Zaks. A combinatorial characterization of the distributed tasks which are solvable in the presence of one faulty processor. In *Proceedings of the 7th Annual ACM Symposium on Principles of Distributed Computing*, pages 263–275, August 1988.

[7] 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.

[8] 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.

[9] D. Dolev, N.A. Lynch, S.S. Pinter, E.W. Stark, and W.E. Weihl. Reaching approximate agreement in the presence of faults. *Journal of the ACM*, 33(3):499–516, July 1986.

[10] A. Fekete. Asymptotically optimal algorithms for approximate agreement. In *Proceedings of the 5th Annual ACM Symposium on Principles of Distributed Computing*, August 1986.

[11] 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.

[12] G.N. Frederickson and N.A. Lynch. Electing a leader in a synchronous ring. *Journal of the ACM*, 34(1):98–115, January 1987.

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

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

[15] M.P. Herlihy. Impossibility results for asynchronous PRAM. In *Proceedings of the 2nd Annual Symposium on Parallel Algorithms and Architectures*, July 1991.

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

[17] A. Israeli and M. Li. Bounded time-stamps. In *Proceedings of the 28th IEEE Symposium on Foundations of Computer Science*, pages 371–382, October 1987.

[18] P. Jayanti and S. Toueg. Some results on the universality, impossibility, and decidability of consensus. In *Proceedings of WDAG 92*, 1992.

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

[20] 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.

[21] N.A. Lynch and M.R. Tuttle. An introduction to input/output automata. Technical Report MIT/LCS/TM-373, MIT Laboratory for Computer Science, November 1988.

[22] S. Mahaney and F.B. Schneider. Inexact agreement: Accuracy, precision, and graceful degradation. In *Proceedings of the 4th Annual ACM Symposium on Principles of Distributed Computing*, August 1985.

[23] 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.

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