Algebraic Topology and Distributed Computing A Primer

Maurice Herlihy[1] and Sergio Rajsbaum[2]

[1] Computer Science Department
Brown University, Providence, RI 02912
herlihy@cs.brown.edu ***
[2] Digital Equipment Corporation, Cambridge Research Lab
One Kendall Square, Cambridge, MA 02139
rajsbaum@crl.dec.com†

Abstract. Models and techniques borrowed from classical algebraic topology have recently yielded a variety of new lower bounds and impossibility results for distributed and concurrent computation. This paper explains the basic concepts underlying this approach, and shows how they apply to a simple distributed problem.

1 Introduction

The problem of coordinating concurrent processes remains one of the central problems of distributed computing. Coordination problems arise at all scales in distributed and concurrent systems, ranging from synchronizing data access in tightly-coupled multiprocessors, to allocating data paths in networks. Coordination is difficult because modern multiprocessor systems are inherently *asynchronous*: processes may be delayed without warning for a variety of reasons, including interrupts, pre-emption, cache misses, communication delays, or failures. These delays can vary enormously in scale: a cache miss might delay a process for fewer than ten instructions, a page fault for a few million instructions, and operating system pre-emption for hundreds of millions of instructions. Coordination protocols that do not take such delays into account run the risk that if one process is unexpectedly delayed, then the remaining processes may be unable to make progress.

Recently, techniques and models borrowed from classical algebraic topology have yielded a variety of new lower bounds for coordination problems. This paper is an attempt to explain the basic concepts and techniques underlying these results. We are particularly interested in making the mathematical concepts accessible to the Computer Science community. Although these concepts are

*** Research partly supported by ONR N00014-91-J-4052, ARPA Order 8225.
† On leave from Instituto de Matemáticas, U.N.A.M., México. Part of this work was done while visiting the Laboratory for Computer Science, MIT. Partly supported by DGAPA Projects.

abstract, they are elementary, being fully covered in the first chapter of Munkres' standard textbook [18].

Our discussion focuses on a class of problems called *decision tasks*, described in Section 2. In Section 3, we show how decision tasks can be modeled using *simplicial complexes*, a standard combinatorial structure from elementary topology. In Section 4, we review the notion of a *chain complex*, which provides an algebraic vocabulary for describing the topological properties of simplicial complexes. In Section 5, we show how these combinatorial and algebraic notions can be applied to prove a variety of lower bounds for a well-known problem in distributed computing, the *k-set agreement* task [7].

2 Model

A set of $n + 1$ sequential threads of control, called *processes*, communicate by applying operations to objects in shared memory. Examples of shared objects include message queues, read/write variables, test-and-set variables, or objects of arbitrary abstract type. Processes are asynchronous: they run at arbitrarily varying speeds. Up to t processes may fail. Because the processes are asynchronous, a protocol cannot distinguish a failed process from a slow process.

To distill the notion of a distributed computation to its simplest interesting form, we focus on a simple but important class of problems called *decision tasks*. We are given a set of $n + 1$ sequential processes $P_0, \ldots, P_n$. Each process starts out with a private *input value*, typically subject to task-specific constraints. The processes communicate for a while, then each process chooses a private *output value*, also subject to task-specific constraints, and then halts.

Decision tasks are intended to model "reactive" systems such as databases, file systems, or flight control systems. An input value represents information entering the system from the outside world, such as a character typed at a keyboard, a message from another computer, or a signal from a sensor. An output value models an effect on the outside world, such as an irrevocable decision to commit a transaction, to dispense cash, or to launch a missile.

Perhaps the simplest example of a decision task is *consensus* [9]. Each process starts with an input value and chooses an output value. All output values must agree, and each output value must have been some process's input value. If the input values are boolean, the task is called *binary consensus*. The consensus task was originally studied as an idealization of the transaction commitment problem, in which a number of database sites must agree on whether to commit or abort a distributed transaction.

A natural generalization of consensus is *k-set agreement* [7]. Like consensus, each process's output value must be some process's input value. Unlike consensus, which requires that all processes agree, k-set agreement requires that no more than k distinct output values be chosen. Consensus is 1-set agreement.

A program that solves a decision task is called a *protocol*. A protocol is *t-resilient* if any non-faulty process will finish the protocol in a fixed number of steps, regardless of failures or delays by up to t other processes. A protocol is *wait-free* if it tolerates failures or delay by all but one of the processes.

3 Combinatorial Structures

Formally, an initial or final state of a process is a *vertex*, $\mathbf{v} = \langle P_i, v_i \rangle$, a pair consisting of a process id and a value (either input or output). A set of $d+1$ mutually compatible initial or final process states is modeled as a *d-dimensional simplex*, (or d-simplex). A simplex is *properly colored* if each vertex is labeled with a distinct process id. The complete set of possible initial (or final) process states is represented by a set of properly colored simplexes, closed under containment, called a *simplicial complex* (or complex). The *dimension* of $\mathcal{C}$ is the dimension of a simplex of largest dimension in $\mathcal{C}$. Where convenient, we use superscripts to indicate dimensions of simplexes and complexes. The k-th *skeleton* of a complex, $skel^k(\mathcal{C}^n)$, is the subcomplex consisting of all simplexes of dimension k or less. The set of process ids associated with simplex S^n is denoted by $ids(S^n)$, and the set of values by $vals(S^n)$. S^m is a (proper) *face* of S^n if the vertexes of S^m are a (proper) subset of the vertices of S^n. If $\mathcal{K}$ and $\mathcal{L}$ are complexes, a *simplicial map* $\phi : \mathcal{K} \to \mathcal{L}$ carries vertexes of $\mathcal{K}$ to vertexes of $\mathcal{L}$ so that simplexes are preserved.

It is often convenient to visualize vertexes, simplexes, and complexes as point sets in Euclidian space. A vertex is simply a point, and an n-simplex is the convex hull of $n+1$ affinely-independent[5] vertexes. A complex is represented by a set of (geometric) simplexes arranged so that that each pair of simplexes intersects either in a common face, or not at all. The point set occupied by such a complex is called its *polyhedron*. Although we use this geometric interpretation for illustrations and informal discussions, we do not use it in our formal treatment.

A *decision task* for $n+1$ processes is given by an *input complex* $\mathcal{I}$, an *output complex* $\mathcal{O}$, and a map Δ carrying each input n-simplex of $\mathcal{I}$ to a set of n-simplexes of $\mathcal{O}$. This map associates with each initial state of the system (an input n-simplex) the set of legal final states (output n-simplexes). It is convenient to extend Δ to simplexes of lower dimension: when $n-t \leq m < n$, $\Delta(S^m)$ is the set of legal final states in executions where only the indicated $m+1$ processes take steps.

For example, the input complex for binary consensus is constructed by assigning independent binary values to $n+1$ processes. We call this complex the *binary n-sphere*, because its polyhedron is homeomorphic to an n-sphere (exercise left to the reader). The output complex consists of two disjoint n-simplexes, corresponding to decision values 0 and 1. Figure 1 illustrates the input and output complexes for two-process binary consensus.

As an example of an interesting output complex, consider the *renaming task* [1], in which each process is given a unique input name taken from a large name space, and must choose a unique output name taken from a much smaller name space. Figure 2 shows the output complex for the three-process renaming task

[5] $\mathbf{v}_0, \ldots, \mathbf{v}_n$ are affinely independent if $\mathbf{v}_1 - \mathbf{v}_0, \ldots, \mathbf{v}_n - \mathbf{v}_0$ are linearly independent.

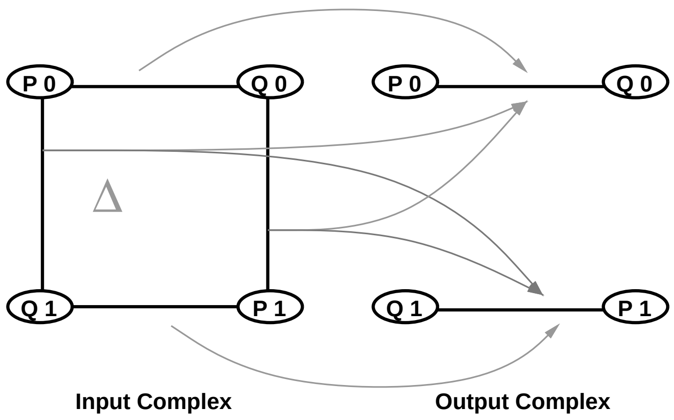

Fig. 1. Input and Output Complexes for 2-Process Consensus

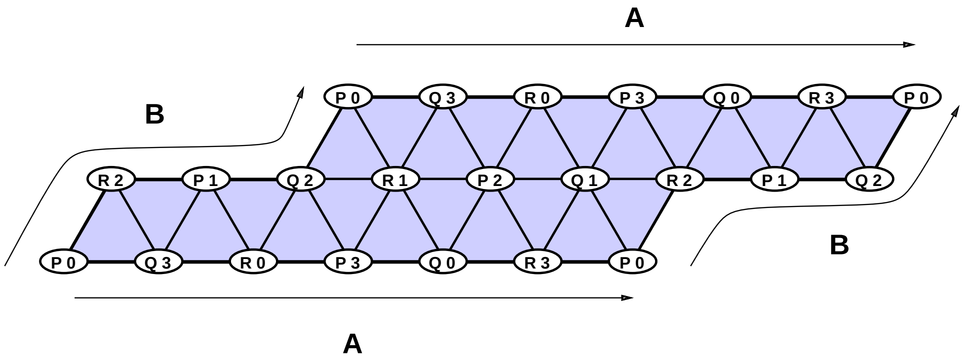

Fig. 2. Output Complex for 3-Process Renaming with 4 Names

using four output names. Notice that the two edges marked A are identical, as are the two edges marked B. By identifying these edges, we can see that this complex has a polyhedron homeomorphic to a torus.

At the end of a protocol, the process's *local state* is its view of the computation: its input value followed by the sequence of operations (including arguments and results) applied to shared objects. It is convenient to view a process as executing a protocol for a fixed number of steps, and then choosing its output value by applying a task-specific *decision map* δ to its local state.

We can treat any protocol as an "uninterpreted" protocol simply by treating each process's local state as its decision value (i.e., omitting the task-specific decision map δ). This uninterpreted protocol itself defines a complex $\mathcal{P}$, called its *protocol complex*. Each vertex $\mathbf{v}$ in this complex is labeled with a process id and a local state such that there exist some execution of the protocol in which process $id(\mathbf{v})$ finishes the protocol with local state $val(\mathbf{v})$. A simplex

$T^m = (\mathbf{t}_0, \ldots, \mathbf{t}_m)$ is in this complex if there is an execution of the protocol in which each process $id(\mathbf{t}_i)$ finishes the protocol with local state $val(\mathbf{t}_i)$ (i.e., the vertexes of any simplex are compatible local states). For an input simplex S^m, let $\mathcal{P}(S^m)$ be the subcomplex of $\mathcal{P}$ generated by executions in which only the processes in $ids(S^m)$ take steps, starting with input values from $vals(S^m)$.

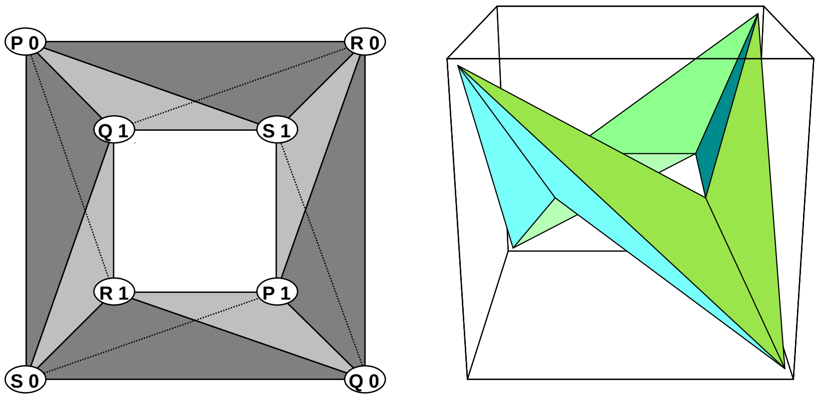

Fig. 3. Two Views of Protocol Complex for Single-Round Test-And-Set Protocol

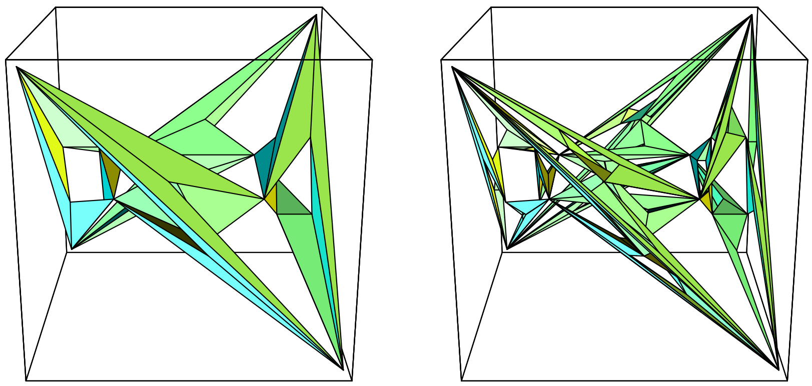

Fig. 4. Protocol Complexes for Multi-Round Test-And-Set Protocols

For example, consider a system in which asynchronous processes communicate by applying *test-and-set*[6] operations to shared variables. Figure 3 shows two views of the protocol complex for a a four-process one-round protocol in which processes P and Q share one test-and-set variable, and R and S share another. Both variables are initialized to 0, and each process executes a single test-and-set

[6] Recall that *test-and-set* atomically writes a 1 to a variable and returns the variable's previous contents.

operation and halts. This complex consists of four tetrahedrons, corresponding to the four possible outcomes of the two test-and-set operations. The left-hand side shows a schematic view of the complex, where each vertex is labeled with a process id and the result of the operation, while the right-hand side shows the same complex in three-dimensional perspective. Figure 4 shows two more protocol complexes for protocols in which the processes respectively iterate two and three-round test-and-set protocols, using fresh, 0-initialized variables for each round.

What does it mean for a protocol to solve a decision task? Recall that a process chooses a decision value by applying a decision map δ to its local state when the protocol is complete. Expressed in our terminology, a protocol solves a decision task $\langle \mathcal{I}, \mathcal{O}, \Delta \rangle$ if and only if there exists a simplicial map $\delta : \mathcal{P} \rightarrow \mathcal{O}$ such that for all $S^m \in \mathcal{I}$, and all $T^m \in \mathcal{P}(S^m)$, $\delta(T^m) \in \Delta(S^m)$, for $n - t \leq m \leq n$. This definition is just a formal way of stating that every execution of the protocol must yield an output value assignment permitted by the decision task. This might seem like a roundabout way to formulate such an obvious property, but it has an important and useful advantage. We have moved from an operational notion of a decision task, expressed in terms of computations unfolding in time, to a purely combinatorial description.

We are now ready to describe our strategy for proving impossibility results. To show that a decision task has no protocol in a given model of computation, it is enough to show that no decision map δ exists. Because decision maps are simplicial, they preserve topological structure. If we can show that a class of protocols generates protocol complexes that are "topologically incompatible" with the task's output complex, then we have established impossibility.

4 Algebraic Structures

So far, our model is entirely combinatorial. To analyze the topological structure of simplicial complexes, however, we now need to introduce some algebraic concepts. Our discussion closely follows that of Munkres [18, Section 1.13], which the reader is encouraged to consult for more details.

Let $\mathcal{K}$ be an n-dimensional simplicial complex, and $S^q = (\mathbf{s}_0, \ldots, \mathbf{s}_q)$ a q-simplex of $\mathcal{K}$. An *orientation* for S^q is an equivalence class of orderings on $\mathbf{s}_0, \ldots, \mathbf{s}_q$, consisting of one particular ordering and all even permutations of it. For example, an orientation of a 1-simplex $(\mathbf{s}_0, \mathbf{s}_1)$ is just a direction, either from $\mathbf{s}_0$ to $\mathbf{s}_1$, or vice-versa. An orientation of a 2-simplex $(\mathbf{s}_0, \mathbf{s}_1, \mathbf{s}_2)$ can be either "clockwise," as in $(\mathbf{s}_0, \mathbf{s}_1, \mathbf{s}_2)$, or "counterclockwise," as in $(\mathbf{s}_0, \mathbf{s}_2, \mathbf{s}_1)$ (see Figure 5). By convention, simplexes are oriented in increasing subscript order unless explicitly stated otherwise.

A *q-chain* of $\mathcal{K}$ is a formal sum of oriented q-simplexes: $\sum_{i=0}^{\ell} \lambda_i \cdot S_i^q$, where each λ_i is an integer, and the S_i^q range over the q-simplexes of $\mathcal{K}$. When writing chains, we typically omit q-simplexes with zero coefficients, unless they are all zero, when we simply write 0. We write $1 \cdot S^q$ as S^q and $-1 \cdot S^q$ as $-S^q$. We identify $-S^q$ with S^q having the opposite orientation. The q-chains of $\mathcal{K}$ form a free Abelian group $C_q(\mathcal{K})$, called the q-th *chain group* of $\mathcal{K}$. For technical reasons, it is convenient to define $C_{-1}(\mathcal{K})$ to be the (infinite cyclic) group of integers under addition.

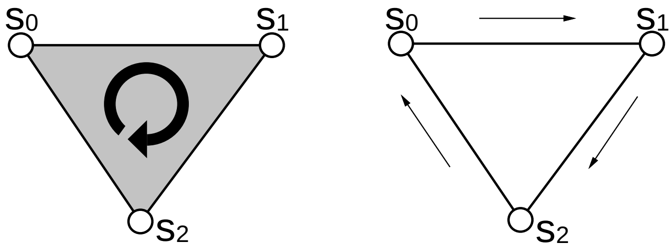

Fig. 5. An Oriented Simplex and its Boundary

Let $S^q = (\mathbf{s}_0, \ldots \mathbf{s}_q)$ be an oriented q-simplex. Define $face_i(S^q)$, the i^{th} *face* of S^q, to be the $(q-1)$-simplex $(\mathbf{s}_0, \ldots, \hat{s}_i, \ldots, \mathbf{s}_q)$, where circumflex denotes omission. The *boundary* operator $\partial_q : C_q(\mathcal{K}) \to C_{q-1}(\mathcal{K})$, $q > 0$, is defined on simplexes:

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

and extends additively to chains: $\partial(\alpha_0 + \alpha_1) = \partial\alpha_0 + \partial\alpha_1$. The zero-dimensional boundary operator[7] $\partial_0 : C_0(\mathcal{K}) \to C_{-1}(\mathcal{K})$ is defined by $\partial_0(\mathbf{s}) = 1$, for each vertex $\mathbf{s}$. The boundary operator has the important property that applying it twice causes chains to vanish:

$$\partial_{q-1}\partial_q \alpha = 0. \qquad (1)$$

The boundary operator is illustrated in Figure 5. Henceforth, we usually omit subscripts from boundary operators.

We illustrate these concepts with an example. Let $S^2 = (\mathbf{s}_0, \mathbf{s}_1, \mathbf{s}_2)$ be an oriented 2-simplex (a "solid" triangle), and $\dot{\mathcal{S}}^1$ the complex of its proper faces (a "hollow" triangle). The complex $\dot{\mathcal{S}}^1$ includes three 0-simplexes (vertexes): $\mathbf{s}_0$, $\mathbf{s}_1$, and $\mathbf{s}_2$, and three 1-simplexes: $S_i^1 = face_i(S^2)$, $0 \leq i \leq 2$. The boundaries for each of these simplexes are shown in Figure 7. The 0-th chain group of $\dot{\mathcal{S}}^1$, $C_0(\dot{\mathcal{S}}^1)$, is generated by the $\mathbf{s}_i$, meaning that all 0-chains have the form

$$\lambda_0 \cdot \mathbf{s}_0 + \lambda_1 \cdot \mathbf{s}_1 + \lambda_2 \cdot \mathbf{s}_2,$$

where the λ_i are integers. The first chain group, $C_1(\dot{\mathcal{S}}^1)$, is generated by the S_i^1, and all 1-chains have the form

$$\lambda_0 \cdot S_0^1 + \lambda_1 \cdot S_1^1 + \lambda_2 \cdot S_2^1,$$

[7] Munkres [18] calls this operator an *augmentation*, and denotes it by ϵ.

where the S_0^1 each have standard orientation. Since $\dot{\mathcal{S}}^1$ contains no simplexes of higher dimension, the higher chain groups are trivial.

A q-chain α is a *boundary* if $\alpha = \partial\beta$ for some $(q+1)$-chain β, and it is a *cycle* if $\partial\alpha = 0$. The boundary chains form a group $B_q(\mathcal{K}) = im(\partial_{q+1})$, and the cycles form a group $Z_q(\mathcal{K}) = ker(\partial_q)$. Equation 1 implies that $B_q(\mathcal{K})$ is a subgroup of $Z_q(\mathcal{K})$, and their quotient group is called the q^{th} *homology group*:[8]

$$H_q(\mathcal{K}) = Z_q(\mathcal{K})/B_q(\mathcal{K}).$$

Informally, homology groups measure the extent to which a complex has holes. Any non-zero element of $H_q(\mathcal{K})$ is a q-cycle but not a q-boundary, corresponding to the intuitive notion of a q-dimensional "hole". Conversely, if $H_q(\mathcal{K}) = 0$ (the trivial single-element group), then every q-cycle is a q-boundary, so $\mathcal{K}$ has no "holes" of dimension q. $H_0(\mathcal{K}) = 0$ if and only if $\mathcal{K}$ is connected. If $H_q(\mathcal{K}) = 0$ for every q, we say that $\mathcal{K}$ is *acyclic*.

For example, $H_0(\dot{\mathcal{S}}^1)$ is trivial, because $\dot{\mathcal{S}}^1$ is connected. $H_1(\dot{\mathcal{S}}^1)$ is non-trivial: the chain ∂S^2 is a cycle (because $\partial\partial S^2 = 0$), but not a boundary (because S^2 is not a simplex of $\dot{\mathcal{S}}^1$). It can be shown that $H_1(\dot{\mathcal{S}}^1)$ is infinite cyclic, generated by the equivalence class of ∂S^2 [18, 31.8].

The *chain complex* $C(\mathcal{K})$ is the sequence of groups and homomorphisms $\{C_q(\mathcal{K}), \partial_q\}$. Let $C(\mathcal{K}) = \{C_q(\mathcal{K}), \partial_q\}$ and $C(\mathcal{L}) = \{C_q(\mathcal{L}), \partial'_q\}$ be chain complexes for simplicial complexes $\mathcal{K}$ and $\mathcal{L}$. A *chain map* ϕ is a family of homomorphisms.

$$\phi_q : C_q(\mathcal{K}) \to C_q(\mathcal{L}),$$

that commute with the boundary operator: $\partial'_q \circ \phi_q = \phi_{q-1} \circ \partial_q$. (In dimension -1, ϕ_{-1} is just the identity map.) Chain maps thus preserve cycles and boundaries.

Recall that a simplicial map from $\mathcal{K}$ to $\mathcal{L}$ carries vertexes of $\mathcal{K}$ to vertexes of $\mathcal{L}$ so that every simplex of $\mathcal{K}$ maps to a simplex of $\mathcal{L}$. Any simplicial map ϕ induces a chain map $\phi_\#$ from $C(\mathcal{K})$ to $C(\mathcal{L})$: when $\phi(S^q)$ is of dimension q, $\phi_\#(S^q) = \phi(S^q)$, otherwise $\phi_\#(S^q) = 0$. Note, however, that not all chain maps are induced by simplicial maps. Henceforth, we abuse notion by omitting subscripts and sharp signs from chain maps. In this paper, we define chain maps by giving their values on simplexes (the chain group generators) and extending additively.

If $\phi, \psi : C(\mathcal{K}) \to C(\mathcal{L})$ are chain maps, then a *chain homotopy* from ϕ to ψ is a family of homomorphisms

$$D_q : C_q(\mathcal{K}) \to C_{q+1}(\mathcal{L}),$$

such that

$$\partial'_{q+1} D_q + D_{q-1}\partial_q = \phi_q - \psi_q.$$

Very roughly, if two chain maps are homotopic, then one can be deformed into the other; see Munkres [18] for intuitive justification for this definition.

[8] Strictly speaking, these are the *reduced* homology groups [18, p.71].

Definition 1. An *acyclic carrier* from $\mathcal{K}$ to $\mathcal{L}$ is a function Σ that assigns to each simplex S^q of $\mathcal{K}$ a non-empty subcomplex of $\mathcal{L}$ such that (1) $\Sigma(S^q)$ is acyclic, and (2) if S^p is a face of S^q, then $\Sigma(S^p) \subseteq \Sigma(S^q)$.

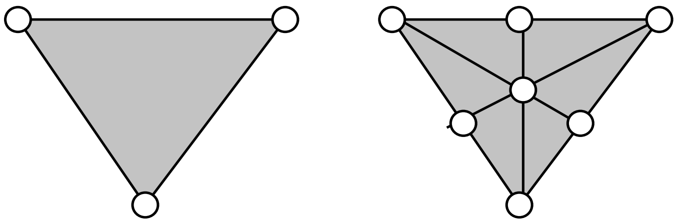

Fig. 6. The Complex $\mathcal{S}^2$ and a Subdivision

For example, a *subdivision* of a complex is an acyclic carrier. Informally, a complex is subdivided by partitioning each component simplex into smaller simplexes, as illustrated in Figure 6. Here, the acyclic carrier maps the 2-simplex S^2 to the entire subdivided complex, and each face $\mathcal{S}^1_i$ to the corresponding subdivided face. While every subdivision is an acyclic carrier, an acyclic carrier need not be a subdivision.

A homomorphism $\phi : C_q(\mathcal{K}) \to C_q(\mathcal{L})$ is *carried* by Σ if each simplex appearing with a non-zero coefficient in $\phi(S^m)$ is in the subcomplex $\Sigma(S^m)$.

Theorem 2 [Acyclic Carrier Theorem]. *Let Σ be an acyclic carrier from $\mathcal{K}$ to $\mathcal{L}$.*

(1) *If ϕ and ψ are two chain maps from $C(\mathcal{K})$ to $C(\mathcal{L})$ that are carried by Σ, then there exists a chain homotopy of ϕ to ψ that is also carried by Σ.*
(2) *There exists a chain map from $C(\mathcal{K})$ to $C(\mathcal{L})$ that is carried by Σ.*

A proof of this theorem can be found in Munkres's text [18, 13.3]. The following lemma is an immediate consequence of the definitions.

Lemma 3. *If $\phi, \psi : C(\mathcal{K}) \to C(\mathcal{L})$ are both carried by Σ, and for each S^q in $\mathcal{K}$, $q = \dim(S^q) = \dim(\Sigma(S^q))$, then $C_{q+1}(\Sigma(S^q)) = 0$, $D_i = 0$ for all i, and ϕ and ψ are equal chain maps.*

Returning to our example, the rotation map $\rho : \dot{\mathcal{S}}^1 \to \dot{\mathcal{S}}^1$ defined by $\rho(\mathbf{s}_i) = \mathbf{s}_{i+1 \bmod 3}$ induces a chain map $\rho : C(\dot{\mathcal{S}}^1) \to C(\dot{\mathcal{S}}^1)$, shown in Figure 7. To verify that ρ is a chain map, it suffices to check that $\rho(\partial S^1_i) = \partial\rho(S^1_i)$. The identity map $\iota : \dot{\mathcal{S}}^1 \to \dot{\mathcal{S}}^1$ also induces a chain map $\iota : C(\dot{\mathcal{S}}^1) \to C(\dot{\mathcal{S}}^1)$. We now show that ι and ρ are chain homotopic, by displaying both an acyclic carrier Σ, and an explicit chain homotopy D. $\Sigma(\mathbf{s}_i)$ is the complex consisting of $S^1_{i-1 \bmod 3}$ and its vertexes, and $\Sigma(S^1_i)$ is the subcomplex of $\dot{\mathcal{S}}^1$ containing S^1_i, $\rho(S^1_i)$, and their vertexes. Both ι and ρ are carried by Σ, and both $\Sigma(\mathbf{s}_i)$ and $\Sigma(S^1_i)$ are acyclic

	$\mathbf{s}_0$	$\mathbf{s}_1$	$\mathbf{s}_2$	S_0^1	S_1^1	S_2^1
∂	1	1	1	$\mathbf{s}_2 - \mathbf{s}_1$	$\mathbf{s}_2 - \mathbf{s}_0$	$\mathbf{s}_1 - \mathbf{s}_0$
ρ	$\mathbf{s}_1$	$\mathbf{s}_2$	$\mathbf{s}_0$	$-S_1^1$	$-S_2^1$	S_0^1
D	$-S_2^1$	$-S_0^1$	S_1^1	0	0	0
ϕ	$\mathbf{s}_0$	$\mathbf{s}_1$	$\mathbf{s}_2$	$S_0^1 + \partial S^2$	$S_1^1 - \partial S^2$	$S_2^1 + \partial S^2$

Fig. 7. Maps used in Extended Example

(being contractible). The chain homotopy D is given in Figure 7. It is easily verified that

$$(D\partial + \partial D)(S) = (\iota - \rho)(S).$$

Although every simplicial map induces a chain map, some chain maps are not induced by any simplicial map. Consider the chain map ϕ given in Figure 7. Notice that $\phi(\partial S^2) = 4 \cdot \partial S^2$, so this map "wraps" the triangle boundary around itself four times, something no simplicial map could do. This map is not chain homotopic to ι or ρ, which "wrap around" only once.

5 An Application

Recall that in the k-set agreement task [7], each process is required to choose some process's input value, and the set of values chosen should have size at most k. We first give a theorem specifying an algebraic property that prevents a protocol from solving k-set agreement, and then we apply this theorem to a variety of different models of computation. The arguments presented here are taken from Herlihy and Rajsbaum [13].

Theorem 4. *Let S^ℓ be a simplex each of whose vertexes is labeled with a distinct input value, and $\mathcal{S}^\ell$ the complex of its faces. Let Π be a protocol, $\mathcal{P}$ its protocol complex, and δ its decision map. If there exists an acyclic carrier Σ from $\mathcal{S}^\ell$ to $\mathcal{P}$ such that*

$$vals(\delta(\Sigma(S))) = vals(S) \tag{2}$$

for all simplexes S in $\mathcal{S}^\ell$, then Π cannot solve k-set agreement for $k \leq \ell$.

Proof. Let $\pi : C(\mathcal{O}) \to C(\mathcal{S}^\ell)$ be the chain map induced by the simplicial map sending $\langle P_i, v_j \rangle$ to the vertex of $\mathcal{S}^\ell$ with value v_j. The acyclic carrier theorem guarantees a chain map $\sigma : C(\mathcal{S}^\ell) \to C(\mathcal{P})$ carried by Σ. By slight abuse of notation, let δ be the chain map induced by the (simplicial) decision map δ. We have:

$$C(\mathcal{S}^\ell) \xrightarrow{\sigma} C(\mathcal{P}) \xrightarrow{\delta} C(\mathcal{O}) \xrightarrow{\pi} C(\mathcal{S}^\ell)$$

Let $\phi : C(\mathcal{S}^\ell) \to C(\mathcal{S}^\ell)$ be the composition of σ, δ, and π. Let Φ be the acyclic carrier from $\mathcal{S}^\ell$ to itself, $\Phi(S^i) = \mathcal{S}^i$, and ι the identity chain map on $\mathcal{S}^\ell$. Thus ι is carried by Φ. Equation 2 implies that Φ is an acyclic carrier also for ϕ.

Because $\dim(S^i) = i = \dim(\Phi(S^i))$, Lemma 3 implies that the two maps are equal. Therefore $\iota(S^\ell) = \phi(S^\ell) = S^\ell$.

Assume for contradiction that $k \leq \ell$. In each execution, however, no more than ℓ values are chosen, implying that π reduces the dimension of every ℓ-simplex. The chain map π thus sends every ℓ-simplex to the 0 chain, so $\pi \circ \delta \circ \sigma(S^\ell) = \phi(S^\ell) = 0$, a contradiction. □

We now show how to apply Theorem 4. Consider a model in which processes communicate by reading and writing shared variables. For any wait-free protocol, Herlihy and Shavit [14] showed that $\mathcal{P}(S^m)$ is acyclic for every input simplex S^m, $0 \leq m \leq n$. Let S^n be an input simplex where each vertex has a distinct input. The map Σ_{WF} that assigns to each face S^m of S^n the protocol subcomplex $\mathcal{P}(S^m)$ is an acyclic carrier. The only input values read or written in any execution in $\mathcal{P}(S^m)$ are the values from S^m, so each process must decide some value from $vals(S^m)$, and therefore Σ_{WF} satisfies the conditions of Theorem 4.

Corollary 5. *There is no wait-free read/write n-consensus protocol [4, 14, 20].*

For t-resilient protocols, a similar argument shows:

Corollary 6. *There is no t-resilient read/write t-consensus protocol.*

Although read/write memories are important from a theoretical point of view, modern multiprocessor architectures provide a variety of more powerful synchronization primitives, including test-and-set, fetch-and-add, compare-and-swap, and load-linked/store-conditional operations. How do these primitives affect the ability to solve k-set agreement? One way to classify these synchronization primitives is by *consensus number* [11, 16]: how many processes can solve consensus using such primitives. For example, test-and-set has consensus number 2, meaning that protocols using read, write, and test-and-set operations can solve consensus for two processes, but not three.

There is a direct connection between an object's consensus number and the homology of its protocol complexes. Consider a system in which processes share both read/write variables and objects that allow any c processes to reach consensus. Using shared objects that solve consensus among c processes, Herlihy and Rajsbaum [12] showed that it is possible to solve k-set agreement for $k = \lceil (n+1)/c \rceil$ (via an easy protocol, left to the reader), but for no lower value of k. In other words, for any protocol complex $\mathcal{P}$ in this model, $\mathcal{P}(S^n)$ has no holes of dimension less than $\lceil (n+1)/c \rceil - 1$ (i.e., these low-order homology groups are trivial). This result is illustrated by the protocol complexes of Figures 3 and 4: here, $n = 3$ and $c = 2$. Each of these complexes has non-trivial homology in dimension $\lceil 4/2 \rceil - 1 = 1$, but trivial homology in dimension $\lceil 4/2 \rceil - 2 = 0$ (being connected). More generally, at one extreme, when $c = 1$, $\mathcal{P}(S^n)$ is acyclic [14]. For higher consensus numbers, 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 $c = n + 1$, the protocol complex may become disconnected.

Let S^j be an input simplex, where $j = \lceil (n+1)/c \rceil - 1$, and $\mathcal{S}^j$ its complex of faces. These observations about the homology of the protocol complex can be exploited to construct an acyclic carrier Σ_c from $\mathcal{S}^j$ to $\mathcal{P}$. This carrier does not directly satisfy Equation 2, because $\Sigma_c(S)$ may include processes not in $ids(S)$. Nevertheless, it is possible to modify the decision values of the processes in $ids(\Sigma_c(S)) - ids(S)$ so as to satisfy Equation 2 (see [12] for details).

Corollary 7. *There is no wait-free* $(\lceil (n+1)/c \rceil - 1)$*-set agreement protocol if processes share read/write variables and objects with consensus number c [12].*

This result can be further generalized by including objects that allow any m processes to solve j-set agreement, a task we call (m, j)-consensus. There is an acyclic carrier from ℓ-simplexes of $\mathcal{I}$ to $\mathcal{P}$, for $\ell \leq J(n+1)$, where

$$J(u) = j \left\lfloor \frac{u}{m} \right\rfloor + \min\{j, u \bmod m\} - 1. \qquad (3)$$

By a similar argument:

Corollary 8. *There is no wait-free* $(n+1, J(n+1)-1)$*-consensus protocol if processes share a read/write memory and* (m, j)*-consensus objects.*

Finally, we turn our attention to synchronous models. Chaudhuri, Herlihy, Lynch, and Tuttle [8] considered a model in which $n+1$ processes communicate by sending messages over a completely connected network. Computation in this model proceeds in a sequence of rounds. In each round, processes send messages to other processes, then receive messages sent to them in the same round, and then perform some local computation and change state. Communication is reliable, but up to t processes can fail by stopping in the middle of the protocol, perhaps after sending only a subset of their messages. Let $\mathcal{P}_r$ be the protocol complex after r rounds. In the *Bermuda Triangle* construction, Chaudhuri et al. identified an acyclic carrier from an ℓ-simplex S^ℓ to $\mathcal{P}_r$, for $r < \lfloor t/\ell \rfloor$, satisfying Equation 2.

Corollary 9. *There is no* t*-resilient* $(n+1, k)$*-consensus protocol that takes fewer than* $\lfloor t/k \rfloor + 1$ *rounds in the synchronous fail-stop message-passing model [8].*

6 Related Work

In 1985, Fischer, Lynch, and Paterson [9] showed that the consensus task has no 1-resilient solution in a system where asynchronous processes communicate by exchanging messages. In 1988, Biran, Moran, and Zaks [3] gave a graph-theoretic characterization of a class of tasks that could be solved in asynchronous message-passing systems in the presence of a single failure.

Herlihy and Shavit [14] were the first to use simplicial complexes to model decision tasks, and to formulate properties of decision tasks in terms of simplicial homology. They showed that the protocol complex for every wait-free read/write protocol is simply connected with trivial homology (i.e., it has no holes). They also give a complete characterization of tasks that have a wait-free solution in read/write memory [14, 15]. Herlihy and Rajsbaum [12] showed that if read/write variables are augmented by more powerful shared objects than read/write registers, then the protocol complexes may have holes (non-trivial homology), but only in the higher dimensions.

The k-set agreement task was first proposed by Soma Chaudhuri [7] in 1989, along with a conjecture that it could not be solved in asynchronous systems. In 1993, three independent research teams, Borowsky and Gafni [4], Herlihy and Shavit [14], and Saks and Zaharoglou [20] proved this conjecture correct.

Attiya, Bar-Noy, Dolev, Koller, Peleg, and Reischuk [1] showed that in asynchronous systems, the renaming task has a solution if the output name space is sufficiently large, but they were unable to demonstrate the existence of a solution for a range of smaller output name spaces. In 1993, Herlihy and Shavit [14] showed that the task has no solution for these smaller name spaces.

Most of the technical content this paper is adapted from Herlihy and Rajsbaum [13], which simplifies the impossibility results of [14] by eliminating the need for certain continuous arguments. Attiya and Rajsbaum [2] take a different approach, proving a number of results about wait-free read/write memory by extending the simplicial model with a combinatorial notion called a "divided image".

In an intriguing recent development, Gafni and Koutsoupias [10] have shown that it is undecidable whether a wait-free read/write protocol exists for three-process tasks, using an argument based on the impossibility of computing the protocol complex's fundamental group, a topological invariant related to the first homology group.

7 Conclusions

We believe that these techniques and models, borrowed from classical algebraic topology, represent a promising new approach to the theory of distributed computing. Because these notions come from a mature branch of mainstream mathematics, the terminology (and to a lesser degree, the notation) is largely standardized, and the formalisms have been thoroughly debugged. Most importantly, however, this model makes it possible to exploit the extensive literature that has accumulated in the century since Poincaré and others invented modern algebraic topology.

We close with a brief summary of some open problems. The problem of classifying the computational power of objects for wait-free computation has attracted the attention of many researchers [5, 6, 16, 17, 19]. We have seen that the protocol complexes for different kinds of objects share certain topological properties: read/write complexes have no holes, set-agreement complexes have no holes below a certain dimension, and so on. It is intriguing to speculate whether some kind of topological classification of protocol complexes might yield a useful computational classification of objects.

We believe the time is ripe to apply these techniques to other tasks, to other models that make different timing or failure assumptions, as well as to long-lived objects that service repeated requests. Finally, most of the results surveyed here are impossibility results; little is known about the complexity of solvable problems in most models.

Acknowledgments

We are grateful to Hagit Attiya, Prasad Chalasani, Alan Fekete, Michaelangelo Grigni, Somesh Jha, Shlomo Moran, Lyle Ramshaw, Amber Settle, Lewis Stiller, Mark Tuttle, and David Wittenberg for their comments.

References

1. H. Attiya, A. Bar-Noy, D. Dolev, D. Peleg, and R. Reischuk. Renaming in an asynchronous environment. *Journal of the ACM*, 37(3):524–548, July 1990.
2. H. Attiya and S. Rajsbaum. A combinatorial topology framework for wait-free computability. Preprint.
3. 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 7th Annual ACM Symposium on Principles of Distributed Computing*, pages 263–275, August 1988.
4. E. Borowsky and E. Gafni. Generalized FLP impossibility result for t-resilient asynchronous computations. In *Proceedings 25th Annual ACM Symposium on Theory of Computing*, pages 206–215, May 1993.
5. E. Borowsky, E. Gafni, and Y. Afek. Consensus power makes (some) sense! In *Proceedings 13th Annual ACM Symposium on Principles of Distributed Computing*, pages 363–373, August 1994.
6. T Chandra, V. Hadzilacos, P. Jayanti, and S. Toueg. Wait-freedom vs. t-resiliency and the robustness of wait-free hierarchies. In *Proceedings 13th Annual ACM Symposium on Principles of Distributed Computing*, pages 334–343, August 1994.
7. S. Chaudhuri. Agreement is harder than consensus: Set consensus problems in totally asynchronous systems. In *Proceedings 9th Annual ACM Symposium On Principles of Distributed Computing*, pages 311–234, August 1990.
8. S. Chaudhuri, M.P. Herlihy, N. Lynch, and M.R. Tuttle. A tight lower bound for k-set agreement. In *Proceedings 34th annual IEEE Symposium on Foundations of Computer Science*, pages 206–215, October 1993.
9. M. Fischer, N.A. Lynch, and M.S. Paterson. Impossibility of distributed commit with one faulty process. *Journal of the ACM*, 32(2):374–382, April 1985.
10. E. Gafni and E. Koutsoupias. 3-processor tasks are undecidable. In *Proceedings 14th Annual ACM Symposium on Principles of Distributed Computing*, August 1995.
11. M.P. Herlihy. Wait-free synchronization. *ACM Transactions on Programming Languages and Systems*, 13(1):123–149, January 1991.
12. M.P. Herlihy and S. Rajsbaum. Set consensus using arbitrary objects. In *Proceedings 13th Annual ACM Symposium on Principles of Distributed Computing*, August 1994.

13. M.P. Herlihy and S. Rajsbaum. Algebraic spans. In *Proceedings 14th Annual ACM Symposium on Principles of Distributed Computing*, August 1995.
14. M.P. Herlihy and N. Shavit. The asynchronous computability theorem for t-resilient tasks. In *Proceedings 25th Annual ACM Symposium on Theory of Computing*, pages 111–120, May 1993.
15. M.P. Herlihy and N. Shavit. A simple constructive computability theorem for wait-free computation. In *Proceedings 26th Annual ACM Symposium on Theory of Computing*, pages 243–252, May 1994.
16. P. Jayanti. On the robustness of Herlihy's hierarchy. In *Proceedings 12th Annual ACM Symposium on Principles of Distributed Computing*, pages 145–158, August 1993.
17. J. Kleinberg and S. Mullainathan. Resource bounds and combinations of consensus objects. In *Proceedings 12th Annual ACM Symposium on Principles of Distributed Computing*, pages 133–145, August 1993.
18. J.R. Munkres. *Elements Of Algebraic Topology.* Addison Wesley, Reading MA, 1984. ISBN 0-201-04586-9.
19. G. Peterson, R. Bazzi, and G. Neiger. A gap theorem for consensus types. In *Proceedings 13th Annual ACM Symposium on Principles of Distributed Computing*, pages 344–354, August 1994.
20. M. Saks and F. Zaharoglou. Wait-free k-set agreement is impossible: The topology of public knowledge. In *Proceedings 25th Annual ACM Symposium on Theory of Computing*, pages 101–110, May 1993.