The Topological Structure of Asynchronous Computability

Maurice Herlihy and Nir Shavit

Department of Computer Science
Brown University
Providence, Rhode Island 02912

CS-96-03
January 1996

The Topological Structure of Asynchronous Computability

Maurice Herlihy*
Computer Science Department
Brown University
Providence RI 02912

Nir Shavit†
Computer Science Department
Tel-Aviv University
Tel-Aviv 69978, Israel

January 27, 1996

*Supported by NSF grant DMS-9505949.
†Supported by NSF grant CCR-9520298.

Abstract

We give necessary and sufficient combinatorial conditions characterizing the tasks that can be solved by asynchronous processes, of which all but one can fail, that communicate by reading and writing a shared memory. We introduce a new formalism for tasks, based on notions from classical algebraic and combinatorial topology, in which a task's possible input and output values are each associated with high-dimensional geometric structures called simplicial complexes. We characterize computability in terms of the topological properties of these complexes. This characterization has a surprising geometric interpretation: a task is solvable if and only if the complex representing the task's allowable inputs can be mapped to the complex representing the task's allowable outputs by a function satisfying certain simple regularity properties.

Our formalism thus replaces the "operational" notion of a wait-free decision task, expressed in terms of interleaved computations unfolding in time, by a static "combinatorial" description expressed in terms of relations among topological spaces, allowing us to exploit powerful theorems from the classic literature on algebraic and combinatorial topology. This approach yields the first impossibility results for several long-standing open problems in distributed computing, such as the "renaming" problem of Attiya et al., and the "k-set agreement" problem of Chaudhuri.

Preliminary versions of these results appeared in the 1993 and 1994 Symposiums on Theory of Computing [30, 31].

Keywords: Asynchronous Distributed Computation, Algebraic Topology, Homology, Wait-free Algorithms, Decision Tasks.

1 Introduction

Modern multiprocessors, whether they communicate by message-passing or through shared memory, are inherently *asynchronous*: processes can be halted or delayed without warning by cache misses, interrupts, or scheduler pre-emption. A *task* in an *asynchronous system* is a problem where each process starts with a private input value, communicates with the others, and halts with a private output value. A *protocol* is a program that solves the task. In asynchronous systems, it is desirable to design protocols that are *wait-free*: any process that continues to run will halt with an output value in a fixed number of steps, regardless of delays or failures by other processes.

Under what circumstances does a task have a wait-free protocol? In this paper, we give the first completely combinatorial characterization of the circumstances under which tasks have wait-free protocols in shared read/write memory. We show that any task and any protocol can be associated with a pair of high-dimensional geometric structures called *simplicial complexes*. A protocol solves a task if and only if their simplicial complexes can be mapped to one another by a function satisfying certain simple regularity properties. Our main theorem gives necessary and sufficient conditions for such a map to exist.

Although our characterization is quite general, it has concrete applications. In particular, it yields the first impossibility results for several long-standing open problems in distributed computing, including the *renaming* problem of Attiya et. al. [6] with a small number of names, and the *set agreement* problem of Chaudhuri [13]. It is also a building block in other characterizations such as Afek and Stupp's characterization of the effect of register size on the power of multiprocessor synchronization operations [2].

Informally speaking, impossibility is demonstrated as follows. Our theorem implies that there exists a map from the protocol complex to the task complex that preserves certain topological properties. We can establish that no map exists by showing that the complexes are topologically "incompatible," in much the same way that classical algebraic topology uses topological invariants to prove that two spaces cannot be homeomorphic. In particular, we show that the simplicial complex associated with any wait-free protocol using read/write memory has a remarkable topological property: it has no "holes" in any dimension. We exploit this simple property to derive our impossibility results.

In a fundamental paper in 1985, Fischer, Lynch, and Paterson [19] showed that there exists a simple task that cannot be solved in a message-passing system if even one process may fail by halting (or may be infinitely slow). This result showed that the notion of "asynchronous computability" differs in important ways from conventional notions of computability (such as sequential "Turing" computability). It led to the creation of a highly active research area, the full scope of which is surveyed in recent book by Lynch [36].

A first step toward a systematic characterization of asynchronous computability was taken in 1988, when Biran, Moran, and Zaks [8] 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 subsequently received considerable attention, it proved difficult to extend such graph-theoretic approaches to encompass more than a single failure. Even the problem of fully characterizing specific tasks like *renaming* [6] and *set agreement* [13] remained unresolved. Chor and Moscovici [16] later provided a graph-theoretic characterization of tasks solvable in a system where the $n+1$ processes can solve $(n+1)$-process consensus (either deterministically or randomized).

In 1993, three research teams—Borowsky and Gafni [10], Saks and Zaharoglou [39], and the current authors [30], independently derived lower bounds for the *k-set agreement* problem of Chaudhuri. The proof of Borowsky and Gafni [10] is based on a powerful simulation method that allows N-process protocols to be executed by fewer processes in a resilient way. Both Saks and Zaharoglou [39] and the current authors [30] apply notions and techniques from mainstream combinatorial topology. Saks and Zaharoglou introduce an innovative and elegant formal model in which processors' collective knowledge of the unfolding computation is treated as a topological space. They then apply a variant of the Brouwer fixed point theorem [38] to derive impossibility of wait-free set agreement.

While Saks and Zaharoglou exploit notions from point-set topology, our approach [30, 31] is more combinatorial. We introduced a new formalism based on simplicial complexes and homology groups, notions taken from undergraduate-level algebraic topology. Simplicial complexes are a natural generalization of graphs. They provide a notion of dimensionality, absent in earlier graph-theoretic models, that captures in a natural way the effects of multiple failures. A further advantage of this model is that it becomes possible to apply standard results from mainstream mathematics to distributed computation.

The paper is organized as follows. Section 2 provides the details of our new topological framework for asynchronous computation. Section 3 provides examples of how tasks are represented in the framework. Section 4 presents the statement of our main theorem and a collection of example results derived from it, including the impossibility of wait-free *set agreement*. Sections 6 and 5 respectively provide the proofs of the "if" and "only if" parts of our theorem. We conclude the paper with a presentation of a version of our main theorem for "comparison protocols," and use it to prove the impossibility of solving the *renaming* problem with a small number of names.

2 Model

We begin with an informal synopsis of our model. $N = n + 1$ sequential threads of control, called *processes*, communicate by reading and writing shared variables. (For lower bounds, we assume that these variables have unbounded size.) In a *decision task*, each of the $N = n + 1$ processes starts with a private *input* value, communicates by reading and writing variables in shared memory, and halts with a private *output* value. For example, in the well-known *binary consensus* task, the processes have binary inputs, and must agree on some process's input [19]. A *protocol* is a program that solves a decision task. A protocol is *wait-free* if it guarantees that every non-faulty process will finish in a bounded number of steps, no matter how many processes fail.

In this paper, we investigate protocols in which processes communicate by reading and writing variables in shared memory. The literature encompasses a variety of shared-memory models; fortunately they are all equivalent. Given single-bit, single-reader, and single-writer variables, one can construct multi-bit, multi-reader variables (see Lynch's survey [36]). From these variables, in turn, one can implement an *atomic snapshot memory*: an array where each P_i *updates* array element i, and any process can instantaneously *scan* (atomically read) the entire array (see [1, 3] for details.).

In the remainder of this section, we restate this model in more formal terms, and we introduce a number of mathematical concepts we will need later on.

2.1 Decision Tasks

An *input vector* is a sequence of $n + 1$ values, called *components*, each of which is either an input value or the distinguished value $\perp$. At least one component must be distinct from $\perp$. An *output vector* is defined similarly. If $\vec{V}$ is a vector, let $\vec{V}[i]$ denote its i-th component. A vector $\vec{U}$ *matches* $\vec{V}$ if for $0 \leq i \leq n$, $\vec{U}[i] = \perp$ if and only if $\vec{V}[i] = \perp$. A vector $\vec{U}$ is a *prefix* of $\vec{V}$ if $\vec{V}[i] = \perp$ implies that $\vec{U}[i] = \perp$ and for all i such that $\vec{U}[i] \neq \perp$, $\vec{U}[i] = \vec{V}[i]$.

A *task specification* is a map Δ, defined on some non-empty subset of the input vectors, carrying each input vector to a non-empty set of matching output vectors. Informally, an input vector $\vec{I}$ represents a possible initial state of the task, and $\Delta(\vec{I})$ represents the set of legitimate final states for that set of inputs. If the i-th component of a vector is an input or output value, then that value is the input or output to process P_i. If that component is $\perp$, then P_i does not participate (it has no input or output values). Consequently, a *participating* process in an input vector is one whose value is distinct from $\perp$. As seen, our task specifications are non-deterministic: there may be many allowable outputs for a given set of inputs. To be consistent with standard notions of decision tasks, we require that all task specifications meet the following *extensibility* requirement.

Definition 2.1 For any input vectors $\vec{I}$ and $\vec{J}$ where $\vec{I}$ is a prefix of $\vec{J}$, each vector in $\Delta(\vec{I})$ is a prefix of some vector in $\Delta(\vec{J})$.

In other words, the valid outputs in any prefix of a task execution must be consistent with every continuation of this execution. We note that our main theorem and its implications carry through also for tasks that do not meet this requirement as the read/write shared memory model we build on is extensible by definition. However, extensibility is crucial for results such as [2, 12, 26, 29] that use this framework to derive lower bounds in more powerful computational models.

The class of tasks our model describes includes all linearizable *one-time objects*, that is linearizable objects [32] that permit at most one operation per process. Our model incorporates an explicit notion of participating processes because it is convenient to distinguish between tasks such as the following.

Unique-id Each participating process P_i chooses an output $y_i \in \{0, .., n\}$ such that for any pair $P_i \neq P_j$, $y_i \neq y_j$.

Fetch-And-Increment Each participating process P_i chooses an output $y_i \in \{0, .., n\}$ such that (1) for any pair $P_i \neq P_j$, $y_i \neq y_j$, and (2) each y_i is less than or equal to the number of participating processes.

Informally, in an asynchronous fail-stop environment, *Unique-Id* allows identifiers to be assigned statically, while *Fetch-And-Increment* effectively requires that they be assigned dynamically in increasing order. The first is trivially solved in the presence of an arbitrary number of failures, while the second has no solution in read/write memory even if only one process can fail. Note that the model of Biran, Moran, and Zaks [8] does not include a notion of participating processes, although the same effect can be achieved by giving each process an additional "participate" bit as an input.

To simplify the presentation, we henceforth restrict our attention to decision tasks where the number of possible input vectors is finite. Our results hold also for the infinite case, but the proofs and algorithms become longer and more complex. Most decision problems studied in the literature have a finite set of input vectors.

2.2 Objects, Processes, and Protocols

Formally, we model objects, processes, and protocols using a simplified form of the I/O automaton formalism of Lynch and Tuttle [37]. An *I/O automaton* is a non-deterministic automaton with a finite or infinite set of *states*, a set of *input events*, a set of *output events*, and a transition relation given by a set of *steps*, each defining a state transition following a given event. An *execution* of an I/O automaton is an alternating sequence of states and enabled events, starting from an initial state. An automaton *history* is the subsequence of events occurring in an execution. Automata can be composed by identifying input and output events in the natural way (details can be found in [37]).

A *process* P is an automaton with output events $\text{CALL}(P, v, X)$, and $\text{FINISH}(P, v)$; input events $\text{RETURN}(P, v, X)$ and $\text{START}(P, v)$, where P is a process, v is a value, and X an object. An *object* X is an automaton with input events $\text{CALL}(P, v, X)$ and output event $\text{RETURN}(P, v, X)$. An *operation* is a *matching* pair of CALL and RETURN events, that is, having the same name and process id.

A *read/write memory* M is an automaton with input events $\text{READ}(P, a)$ and $\text{WRITE}(P, a, v)$ and output event $\text{RETURN}(P, v)$. A history is *sequential* if it is a sequence of operations. (Notice that a sequential execution permits process steps to be interleaved, but at the granularity of complete operations.) If we restrict our attention to sequential histories, then the behavior of the *read/write memory* is straightforward: any *read* operation at address a returns the value of the last preceding *write* to that address, or $\perp$ if no such write exists. Each history H induces a partial "real-time" order $\prec_H$ on its operations: $op_0 \prec_H op_1$ if the response for op_0 precedes the invocation for op_1. Operations unrelated by $\prec_H$ are said to be *concurrent*. If H is sequential, $\prec_H$ is a total order. A concurrent protocol or object is *linearizable* if its histories can be mapped to sequential ones by making each operation appear to take effect instantaneously at some point between its invocation and its response (see Herlihy and Wing [32] for details). An *atomic* read/write memory automaton is one that is linearizable to the sequential automaton specified above.

A *protocol* $\{P_0, \ldots, P_n; M\}$ is the automaton composed by identifying in the obvious way the events for processes $P_0, \ldots, P_n$ and the memory M. To capture the notion that a process represents a single thread of control, a protocol execution is *well-formed* if every process history (the projection of the history onto the actions of P_i) has a unique START event (generated externally to the protocol), which precedes any CALL or RETURN events, it alternates matching CALL and RETURN events, and has at most one FINISH event. We restrict our attention to well-formed executions.

Operations p and q of object X *commute* if, for all sequential histories H and G, $H \cdot p \cdot q \cdot G$ is a valid history of X if and only if $H \cdot q \cdot p \cdot G$ is a valid history of X (where "$\cdot$" is the concatenation operator). For example, memory READ operations commute with one another, as do memory WRITE operations, a property which will be shown as fundamental in determining the computational power of read/write memory.

For brevity, we express protocols using pseudo-code, although it is straightforward to translate this notation into automaton definitions. Also, as noted above, our protocols use an atomic snapshot memory [1, 3], which can be implemented from the atomic read/write memory described here.

2.3 Solvability

We are interested in solvability of tasks by processes that are computationally equivalent to Turing machines [34] and communicate via atomic read/write memory.

A process is *active* at some point in a history if it does not yet have a FINISH event. An active process is *faulty* at that point if it has no output events later in that history. A history H is t*-faulty* if up to t processes become faulty.

A protocol *solves* a decision task in an execution if the following condition holds. Let $\{P_i | i \in U\}$ be the processes that have START events, and let $\{u_i | i \in U\}$ be their arguments. Let $\{P_j | j \in V\}$, $V \subseteq U$, be the processes that execute FINISH events, and let $\{v_j | j \in V\}$ be their output values. Let $\vec{I}$ be the input vector with u_i in component i, and $\perp$ elsewhere, and let $\vec{O}$ be the corresponding output vector for the v_j. We require that (1) no process takes an infinite number of steps without a FINISH event, and (2) $\vec{O}$ is a prefix of some vector in $\Delta(\vec{I})$. (Informally, the second condition implies that if a protocol solves a task in an execution, its partial outputs in any prefix of the execution are consistent with the allowable outputs of the possibly larger set of inputs to the execution as a whole.) A protocol *solves* a decision task if it solves it in every execution where up to n processes can fail.

It is convenient to assume that any wait-free protocol is expressed in the following *normal form*. The processes share an array V whose $n+1$ elements are all initially $\perp$. Each process has a *local state*, consisting of its input value and the history of values it has so far read from the shared variables. Computation proceeds in a sequence of asynchronous *rounds*, from 0 to some fixed r. In round 0, each process writes its input value to its local variable. In round $r > 0$, each P_i executes a sequence of *steps*: (1) it *updates* $V[i]$ to its current local state, and (2) it atomically *scans* the elements of V, appending them to its local state. After r rounds, P_i computes its output value by applying a task-specific *decision map* δ to its final local state. Figure 2.3 shows a generic protocol in normal form. Because there are only a finite number of input vectors, any kind of wait-free read/write protocol can be expressed in normal form.[1] The memory locations $V[i]$ need only be of bounded size, though for our lower bound proofs we allow them to be unbounded.

```
% Code for process i
update(V[i] := input_value)
for round in 1 .. r do
  local_state := scan(V)
  update(V[i] := local_state)
return δ(local_state)
```

Figure 1: A Wait-Free Protocol in Normal Form

2.4 Simplicial Complexes

We start with a number of standard technical definitions [40].

A *vertex* $\vec{v}$ is a point in a high-dimensional Euclidian space. An *n-dimensional simplex* (or n-simplex) is a set of $n+1$ affinely-independent[2]

[1] If the number of input vectors were unbounded, then the protocol would need an explicit termination test.

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

vertexes. For example, a 0-simplex is a vertex, a 1-simplex a line segment, a 2-simplex a solid triangle, and a 3-simplex a solid tetrahedron. We often use superscripts to indicate the dimension of a simplex, and we sometimes list the vertexes in parentheses: $S^n = (\vec{s}_0, \ldots, \vec{s}_n)$. The convex hull of a simplex S^n is called its *polyhedron*, and is denoted by $|S^n|$. S^m is a (proper) *face* of S^n, written $S^m \subseteq S^n$, if the former's set of vertexes is a (proper) subset of the latter's.

A *simplicial complex* (or complex) is a set of simplexes closed under intersection and containment. The *dimension* of a complex is the highest dimension of any of its simplexes, and is also usually indicated by a subscript. An n-dimensional complex (or n-complex) is *pure* if every simplex is a face of some n-simplex. All complexes considered in this paper are pure. The union of the convex hulls of the simplexes in $\mathcal{C}^n$ is called its *polyhedron*, denoted by $|\mathcal{C}^n|$.

If S^n is an n-simplex, let $\mathcal{S}^n$ denote the complex consisting of all faces of S^n (including S^n itself), and $\dot{\mathcal{S}}^{n-1}$ the complex consisting of all *proper* faces of S^n. We say a complex $\mathcal{C}^n$ is an *n-disk* if $|\mathcal{C}^n|$ is homeomorphic[3] to $|\mathcal{S}^n|$, and it is an *$(n-1)$-sphere* if $|\mathcal{C}^n|$ is homeomorphic to $|\dot{\mathcal{S}}^{n-1}|$.

A *simplicial map* $\mu : \mathcal{A}^n \rightarrow \mathcal{B}^n$ carries vertexes of $\mathcal{A}^n$ to vertexes of $\mathcal{B}^n$ so that every simplex of $\mathcal{A}^n$ maps to a simplex of $\mathcal{B}^n$. Any simplicial map defines a piece-wise linear map $|\mu| : |\mathcal{A}^n| \rightarrow |\mathcal{B}^n|$. A simplicial map ϕ is *non-collapsing* if it preserves dimension: $\dim(\phi(S)) = \dim(S)$. Two complexes $\mathcal{A}^n$ and $\mathcal{B}^n$ are *isomorphic* if there exist simplicial maps $\phi : \mathcal{A}^n \rightarrow \mathcal{B}^n$ and $\psi : \mathcal{B}^n \rightarrow \mathcal{A}^n$, called *isomorphisms*, such that $\phi \cdot \psi$ are $\psi \cdot \phi$ are identity maps. Henceforth, unless stated otherwise, all maps between complexes are assumed to be simplicial.

A *coloring* of a complex $\mathcal{C}^n$ is a non-collapsing map $\chi : \mathcal{C}^n \rightarrow S^n$, where S^n is an n-simplex. A *chromatic complex* is a complex together with a coloring. If $(\mathcal{C}_0, \chi_0)$ and $(\mathcal{C}_1, \chi_1)$ are chromatic complexes, then a map $\phi : \mathcal{C}_0 \rightarrow \mathcal{C}_1$ is *chromatic* if for every vertex $\vec{v} \in \mathcal{C}_0$, $\chi_0(\vec{v}) = \chi_1(\phi(\vec{v}))$. Most, but not all the maps we consider are chromatic.

A *subdivision* of a complex $\mathcal{A}^n$ is a complex $\mathcal{B}^n$ such that (1) the vertexes of $\mathcal{B}^n$ are points of $|\mathcal{A}^n|$, (2) if S^n is a simplex of $\mathcal{B}^n$ there is some simplex $T^n \in \mathcal{A}^n$ such that $S^n \subseteq |T^n|$, and (3) the piece-wise linear map $|\mathcal{B}^n| \rightarrow |\mathcal{A}^n|$ mapping each vertex of $\mathcal{B}^n$ to the corresponding point of $|\mathcal{A}^n|$ is a homeomorphism. (See Figure 19.) If S^m is a simplex of $\mathcal{B}^n$, the *carrier* of S^n, denoted *carrier*(S^n) is the unique smallest $T^\ell \in \mathcal{A}^n$ such that $S^m \subset |T^\ell|$.

[3]Spaces X and Y are *homeomorphic* if there exists a one-to-one continuous map $f : X \rightarrow Y$ with a continuous inverse.

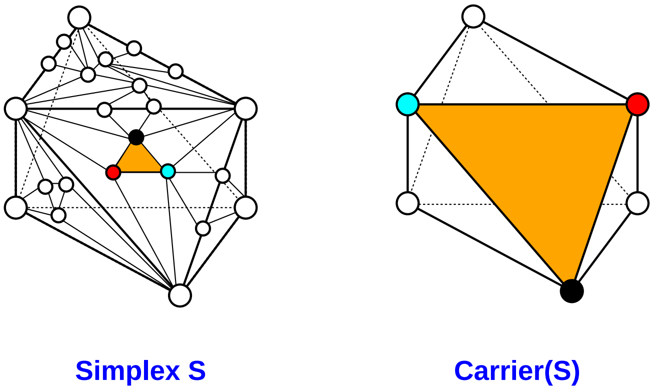

Figure 2: The Carrier of a Simplex

(See Figure 2.)

A *chromatic subdivision* of $(\mathcal{A}_0^n, \chi_0)$ is a chromatic simplex $(\mathcal{A}_1^n, \chi_1)$ such that $\mathcal{A}_1^n$ is a subdivision of $\mathcal{A}_0^n$, and for all S^m in $\mathcal{A}_1^n$, $\chi_1(S^m) \subseteq \chi_0(carrier(S^m))$. A simplicial map $\mu : \mathcal{B}^n \to \mathcal{C}^n$ between subdivisions of $\mathcal{A}^n$ is *carrier preserving* if for all $S^m \in \mathcal{B}^n$, $carrier(S^m) = carrier(\mu(S^m))$. The k-th *skeleton* of a complex, $skel^k(\mathcal{C}^n)$, is the subcomplex consisting of all simplexes of dimension k or less.

2.5 Simplicial Complexes and Tasks

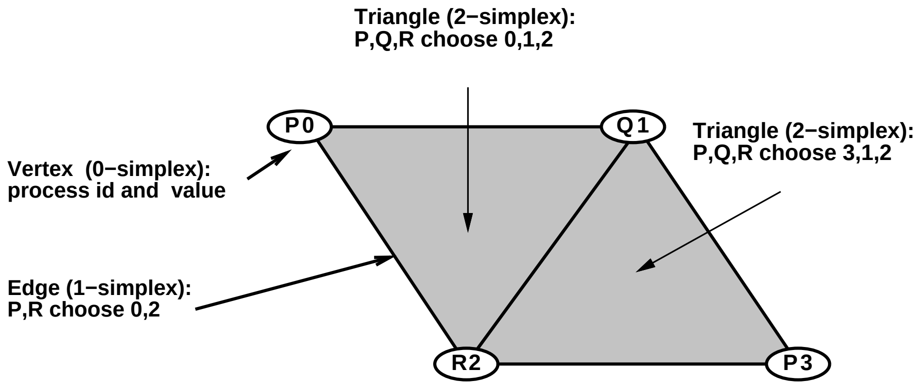

Figure 3: Vertexes and Simplexes

Earlier in this section, we defined the notion of a decision task in terms of input and output vectors. That definition was intended to help the reader understand what a decision task is, but it lacks the mathematical structure necessary to prove interesting results. We now reformulate this definition in terms of simplicial complexes.

To illustrate our constructions, we will use a well-known decision task from the literature:

Renaming The input to each process is a unique *input name* taken from a range $0, \ldots, N$. Each participating process chooses a unique *output name* taken from a strictly smaller range $0, \ldots, K$.

This task was first studied by Attiya et. al [6].

We represent an initial (or final) state of a process as a vertex $\vec{v}$ labeled with a process id and a value: $\langle P_i, v_i \rangle$. The process id labeling $\vec{v}$ is denoted by $id(\vec{v})$, and the value by $val(\vec{v})$. We represent the initial or final states of a set of $n+1$ distinct processes as the n-simplex encompassing those vertexes. We use $ids(S^n)$ to denote the simplex's set of process ids, and $vals(S^n)$ the multiset of values. Figure 3 shows two triangles (2-simplexes) corresponding to two distinct final states for the 3-process renaming task, one in which process P chooses 0, Q chooses 1, and R chooses 2, and another in which P chooses 3, and Q and R choose the same values. Notice that the vertexes of each simplex are colored by the process ids.

This simplicial representation gives a geometric interpretation to the notion of "similar" system states. The vertexes on the common boundary of the two simplexes are local process states that cannot distinguish between the two global states. Unlike graph-theoretic models (e.g., [8]), simplicial complexes capture in a natural way the notion of the *degree* of similarity between two states: it is the dimension of the intersection of the two n-simplexes.

We represent the set of all initial or final states for a decision task as a chromatic complex colored by process ids. For example, Figure 4 shows the output complex for the three-process renaming task using four names. Note that the output complex's polyhedron is homeomorphic (topologically equivalent) to a torus. To see why, notice that the vertexes on edges of the complex are the same, so the edges can be "glued together" in the direction of the arrows.

We are now ready to reformulate our formal definition of a decision task. 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 relation Δ matching each simplex of $\mathcal{I}^n$ to a non-empty set of simplexes of $\mathcal{O}^n$. Δ preserves process ids: for all S^m in

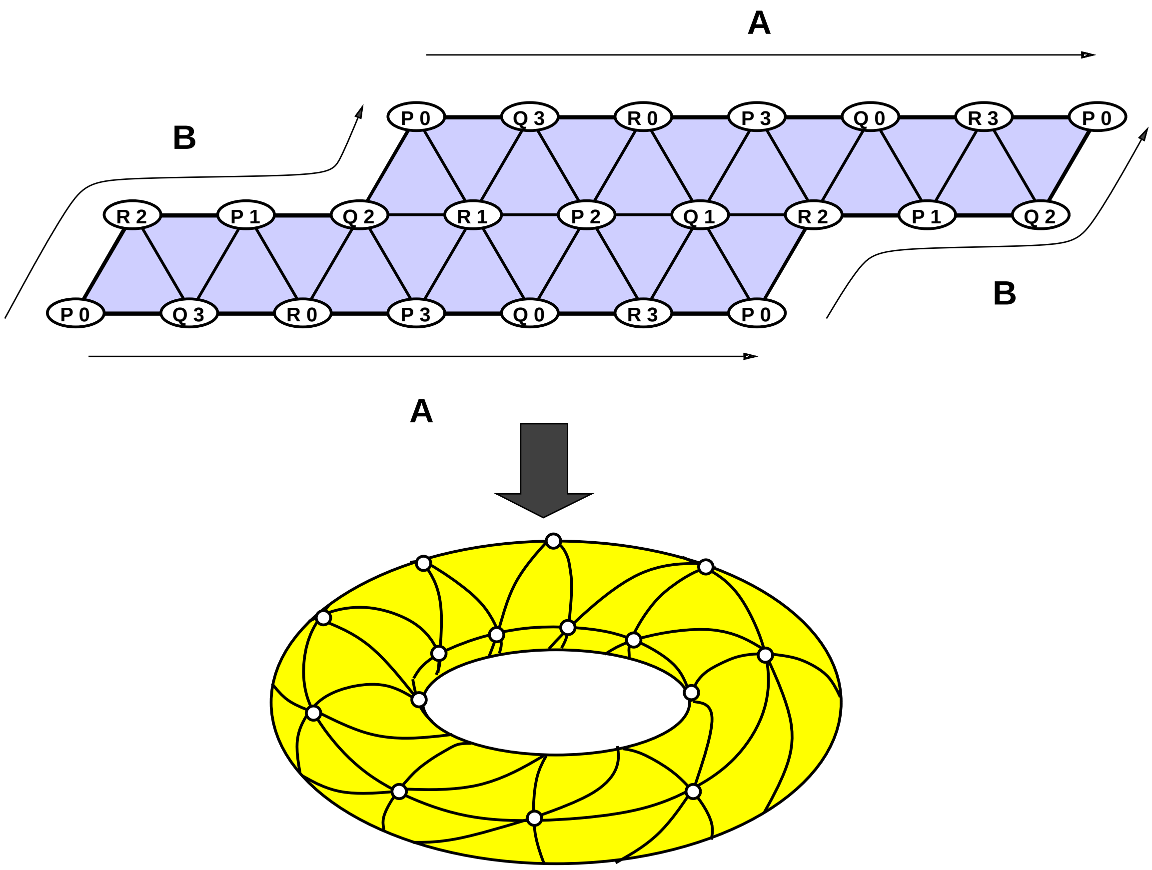

Figure 4: Output Complex for 3-Process Renaming with 4 Names

$\mathcal{I}^n$, and all T^m in $\Delta(S^m)$, $ids(S^m) = ids(T^m)$ (see Figure 5). Simplexes in $\mathcal{I}^n$ and $\mathcal{O}^n$ are respectively called *input* and *output simplexes*.

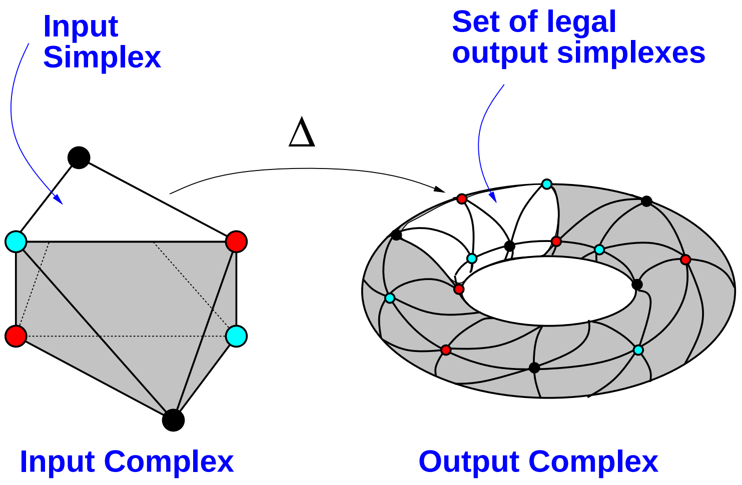

Figure 5: A Decision Task

Informally, each input simplex corresponds to a possible initial state of the protocol. An input m-simplex, where $m \leq n$, specifies the initial states for a subset of $(m+1)$ participating processes. The input complex is the collection of all legal sets of inputs. $\Delta(S^n)$ is the collection of all legal final states when the protocol is started in initial state S^n. 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. $\Delta(S^m)$ for $m < n$ is the set of possible outputs of solo executions by the processes in $ids(S^m)$.

2.6 Tools From Algebraic Topology

The definitions given so far should suffice to understand the statement (and implications) of our main theorem. The remainder of this section consists of definitions needed to understand the proofs of our theorems, and some of the theorem's applications.

The *star* of $S^m \in \mathcal{C}^n$, written $st(S^m, \mathcal{C}^n)$, is the union of all $|T^n|$ such that $S^m \subseteq T^n$ (see Figure 7). The *open star*, written $st^\circ(S^m, \mathcal{C}^n)$, is the interior of the star, and the *link* is the boundary of the star. Although we have defined stars and links to be polyhedra (point-sets), we will sometimes treat them as simplicial complexes, relying on context to clarify the precise meaning. If $S^m = (\vec{s}_0, \ldots, \vec{s}_m)$ and $T^\ell = (\vec{t}_0, \ldots, \vec{t}_\ell)$ are simplexes whose vertexes are affinely independent, their *join*, $S^m \cdot T^\ell$, is the $(m + \ell + 1)$-simplex $(\vec{s}_0, \ldots, \vec{s}_m, \vec{t}_0, \ldots, \vec{t}_\ell)$. If $\mathcal{A}$ and $\mathcal{B}$ are complexes such that every simplex of one is affinely independent of every simplex of the other, then their *join* $\mathcal{A} \cdot \mathcal{B}$ is the set of simplexes of the form $A \cdot B$, where $A \in \mathcal{A}$ and $B \in \mathcal{B}$, together with their faces. A *cone* over $\mathcal{A}$ is $\vec{v} \cdot \mathcal{A}$ for some vertex $\vec{v}$ affinely independent of $\mathcal{A}$.

A complex $\mathcal{A}^n$ is an *n-manifold with boundary* if (1) for every pair of n-simplexes T_0^n, T_1^n in $\mathcal{A}^n$, there exists a sequence of simplexes $S_0^n, \ldots, S_\ell^n$ such that $T_0^n = S_0^n$, $T_1^n = S_\ell^n$, and $S_i^n \cap S_{i+1}^n$ is an $(n-1)$-simplex, and (2) and every $(n-1)$-simplex is contained in either one or two n-simplexes. An $(n-1)$ simplex is *internal* if it is contained in two n-simplexes, and *external* otherwise. The *boundary complex* of $\mathcal{A}^n$, written $\dot{\mathcal{A}}^{n-1}$, is the subcomplex of all external $(n-1)$-simplexes (which is itself an $(n-1)$-manifold). Some, but not all, the complexes we consider are manifolds. Manifolds satisfy the following property [22, Theorem II.2]:

Lemma 2.1 *If $\mathcal{M}$ is an n-manifold with boundary, and T^m an interior simplex, then $lk(T^m, \mathcal{M})$ is an $(n-m-1)$-sphere.*

Many complexes of interest have a simple but important topological property: they have no "holes" in certain dimensions. There are several ways to formalize this notion, but the following is the most convenient for our purposes.

Definition 2.2 A complex $\mathcal{C}$ is *n-connected* ([40, p.51]) if, for $m \leq n$, any continuous map of the m-sphere into $|\mathcal{A}|$ can be extended to a continuous map over the $(m+1)$-disk.

A 0-connected complex is usually called *connected*: there is a path linking every pair of vertices. A 1-connected complex is usually called *simply connected*: any loop (closed path) can be continuously deformed to a point.

Informally, a complex is n-connected if it has no "holes" of dimension n or less: any continuous image of a sphere can be "filled in" (extended to a map of a disk).

We will be interested in proving that certain complexes are n-connected. Unfortunately, connectivity is hard to prove directly, so we must use an indirect approach. The key notion underlying our proof strategy is the sequence of *homology groups* associated with a complex. (Readers completely unfamiliar with the notion may wish to consult any one of a number of standard textbooks [21, 24, 25, 33, 40].) For our purposes, it suffices to note that an n-dimensional complex $\mathcal{C}^n$ is associated with $n+1$ Abelian groups, $H_0(\mathcal{C}^n), \ldots, H_n(\mathcal{C}^n)$, one for each dimension[4] By convention, $H_{-1}(\mathcal{C})$ is the trivial single-element group. If $H_0(\mathcal{C}), \ldots, H_q(\mathcal{C})$ are trivial, then we say $\mathcal{C}$ is *q-acyclic.* If all groups are trivial, we simply say that $\mathcal{C}$ is *acyclic.*

A useful connection between being homology and connectivity is given by the Hurewicz Isomorphism Theorem ([40, 7.5.5]):

Theorem 2.2 (Hurewicz) *A complex is q-connected if and only if it is simply connected and q-acyclic.*

Thus, we can prove a complex is q-connected by showing that it is (1) simply connected, and (2) q-acyclic.

To prove that a complex is simply connected, we use the following special case of the Siefert/Van Kampen Theorem [24, 4.12].

Theorem 2.3 (Siefert/Van Kampen) *If $\mathcal{A}$ and $\mathcal{B}$ are simply connected, and $\mathcal{A} \cap \mathcal{B}$ is connected, then $\mathcal{A} \cup \mathcal{B}$ is simply connected.*

To prove that a complex is q-acyclic, we use the following special case of the Mayer-Vietoris sequence [40, p.186].

Theorem 2.4 (Mayer-Vietoris) *If $\mathcal{A}$ and $\mathcal{B}$ are complexes such that $\mathcal{A} \cap \mathcal{B} \neq \emptyset$, then if $H_q(\mathcal{A})$, $H_q(\mathcal{B})$, and $H_{q-1}(\mathcal{A} \cap \mathcal{B})$ are all trivial, so is $H_q(\mathcal{A} \cup \mathcal{B})$.*

Theorems 2.3 and 2.4 are used inductively to break complexes into component pieces. For the base case, we note some useful examples of n-connected complexes.

Lemma 2.5 *The following complexes are n-connected: (1) the complex $\mathcal{S}^n$ consisting of an n-simplex and its faces, and (2) any cone over an arbitrary complex $\mathcal{C}$.*

These theorems encompass all the topological machinery we need to prove our main theorem. The impossibility proof for renaming presented in Section 7 will require some additional notions, presented later.

[4] Strictly speaking, we use *reduced* homology groups in this paper [38, p.71].

3 Examples

This section is intended to strengthen the reader's intuition by presenting a number of examples of complexes and tasks.

3.1 Consensus

Perhaps the simplest decision task is binary *consensus* [19]. As specified in Figure 6, each process starts with a binary input value and chooses a binary output value. All output values must agree, and each output value must be some process's input value.

$\vec{I}$	$\Delta(\vec{I})$	$\vec{I}$	$\Delta(\vec{I})$
$(0,\bot)$	$(0,\bot)$	$(1,\bot)$	$(1,\bot)$
$(\bot,0)$	$(\bot,0)$	$(\bot,1)$	$(\bot,1)$
$(0,0)$	$(0,0)$	$(1,1)$	$(1,1)$
$(0,1)$	$(0,0),(1,1)$	$(1,0)$	$(0,0),(1,1)$

Figure 6: The Consensus task

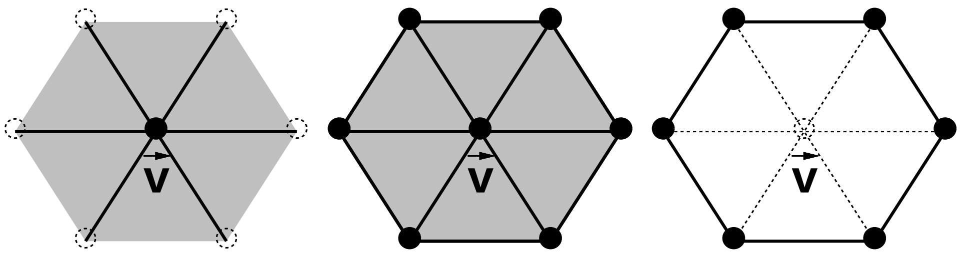

Figure 7: $st^\circ(\vec{v})$, $st(\vec{v})$, and $lk(\vec{v})$.

The input complex for this task is the complex $\mathcal{B}^n$ constructed by assigning independent binary values to $n+1$ processes. We call this complex the *binary n-sphere* (Figure 8). (To see why this is an n-sphere, note that the n-dimensional complex consists of two parts: $\mathcal{E}_0^n$ is the set of n-simplexes containing $\langle P_n, 0\rangle$, and $\mathcal{E}_1^n$ the set containing $\langle P_n, 1\rangle$. Each of these is an n-disk, a cone over the binary $(n-1)$-sphere $\mathcal{B}^{n-1}$. These two n-disks are joined at their boundaries, forming an n-sphere.)

The output complex consists of two disjoint n-simplexes, corresponding to decision values 0 and 1. Figure 9 shows the input and output complexes

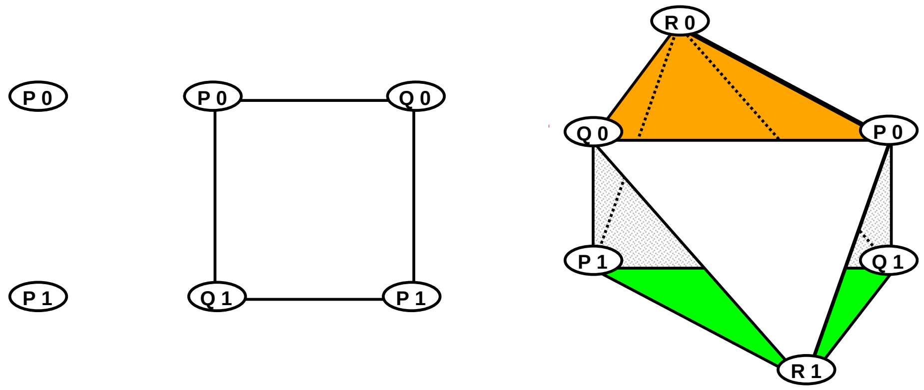

Figure 8: Binary 0, 1, and 2-spheres

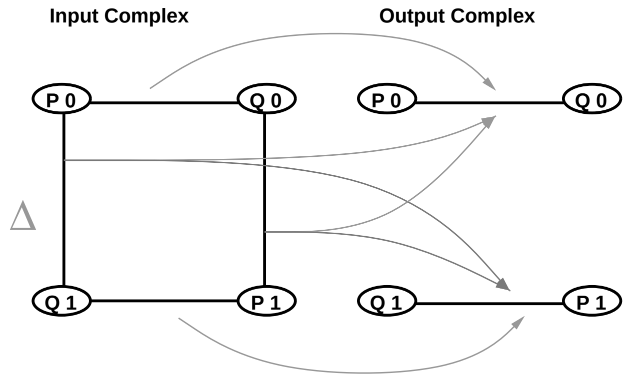

Figure 9: Simplicial Complexes for 2-Process Consensus

for 2-process binary consensus. In general, the input complex is $(n-1)$-acyclic, while the output complex is disconnected. It is known [15, 17, 27, 35], that $(n+1)$-process binary consensus has no wait-free protocol in read/write memory. *Consensus* is the generalization of binary consensus to allow input values from an arbitrary range, not only $\{0, 1\}$.

3.2 Set Agreement

A natural generalization of consensus is the *k-set agreement task* [13].

k-Set Agreement Like consensus, each process starts with an arbitrary input value, and must choose some process's input. Unlike consensus, which requires that all processes agree, set agreement requires that no more than k distinct output values be chosen.

For example, when $n = 2$ and $k = 2$, three processes must choose at most two distinct values. It is not hard to see that the output complex for this problem consists of three binary 2-spheres "linked" in a ring (Figure 3.2). This complex is connected (trivial homology in dimension zero), but has non-trivial homology in dimension one (each sphere is a "hole").

The set agreement problem was first proposed by Soma Chaudhuri [13] in 1989, along with a conjecture that it could not be solved in certain asynchronous systems. This problem remained open until 1993, when three independent research teams, Borowsky and Gafni [10], Herlihy and Shavit [30], and Saks and Zaharoglou [39] proved this conjecture correct.

3.3 Fetch-And-Add

In the *fetch-and-add* task [23], a generalization of the *Fetch-And-Inc* task defined earlier, each of $n+1$ processes atomically adds an integer input (from a fixed range) to a shared register, initially zero, and returns the register's previous contents. Figure 11 shows two input simplexes and their corresponding output complexes. Note that vertexes with the same labels are the same and should be mentally "glued together." They are drawn as distinct only for legibility. It is known [27] that fetch-and-add is not even 1-solvable in read/write memory.

4 The Main Theorem

We are now ready to state our main theorem in its entirety. We give two versions of this theorem, one for arbitrary wait-free read/write protocols,

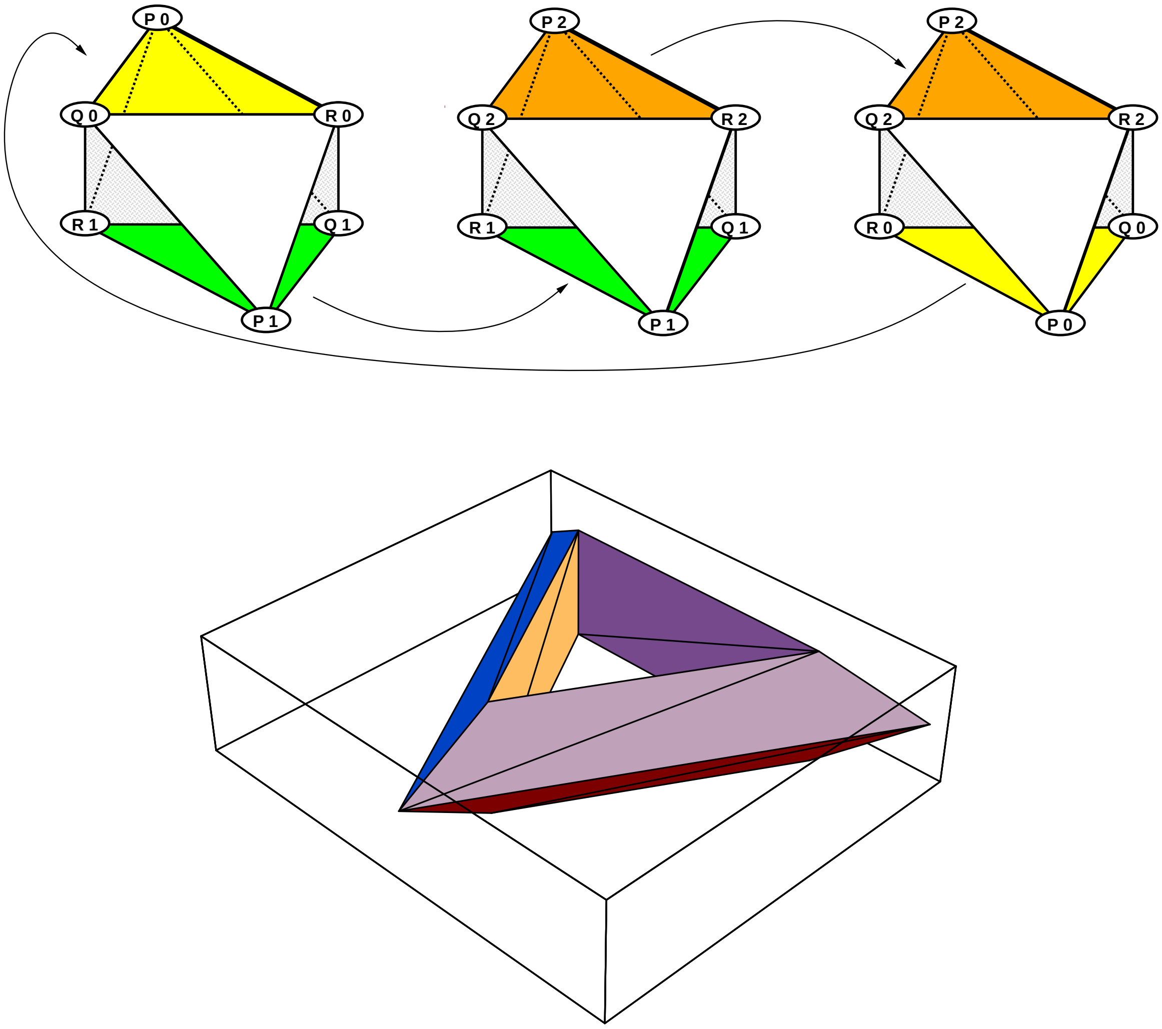

Figure 10: Output Complex for (3,2)-Set Agreement

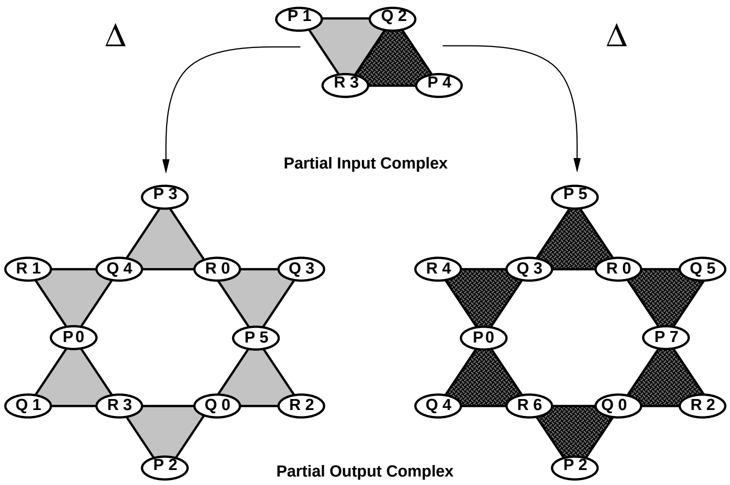

Figure 11: The FETCH-AND-ADD task.

and one for *comparison* protocols, which are allowed to compare process ids for equality and order, but are not allowed to apply any other operations. We start with the general theorem, and defer the presentation of the comparison-based variant to Section 7.

Theorem 4.1 (Asynchronous Computability Theorem) *A decision task* $\langle \mathcal{I}^n, \mathcal{O}^n, \Delta \rangle$ *has a wait-free protocol using read-write memory if and only if there exists a chromatic subdivision* $\sigma(\mathcal{I}^n)$ *and a chromatic simplicial map*

$$\mu : \sigma(\mathcal{I}^n) \to \mathcal{O}^n$$

such that for each simplex S^m *in* $\sigma(\mathcal{I}^n)$, $\mu(S^m) \in \Delta(carrier(S^m))$.

This theorem is illustrated in Figure 12. The proof appears in Sections 5 and 6. In this section, we explore the theorem's consequences.

The simplicial map μ induces a continuous (piece-wise linear) map $|\mu|$ on the complexes' polyhedra such that for each input simplex S^m, $|\mu|(|S^m|) \subset |\Delta(S^m)|$. A task is therefore solvable if and only if the input complex can be continuously "stretched" and "folded" so that each input simplex is carried into its corresponding set of output simplexes. This theorem has intriguing parallels to the classical simplicial approximation theorem [40][5.4.8], which states that any continuous map $|\mathcal{A}^n| \to |\mathcal{B}^n|$ can be approximated by a simplicial map from some subdivision of $\mathcal{A}^n$ to $\mathcal{B}^n$.

We refer to the subdivision $\sigma(S^m)$ as the *span* of S^m. Spans are not necessarily unique, but we will see that they can be chosen to have a highly symmetric structure.

In the remainder of this section, we explore some applications of the theorem.

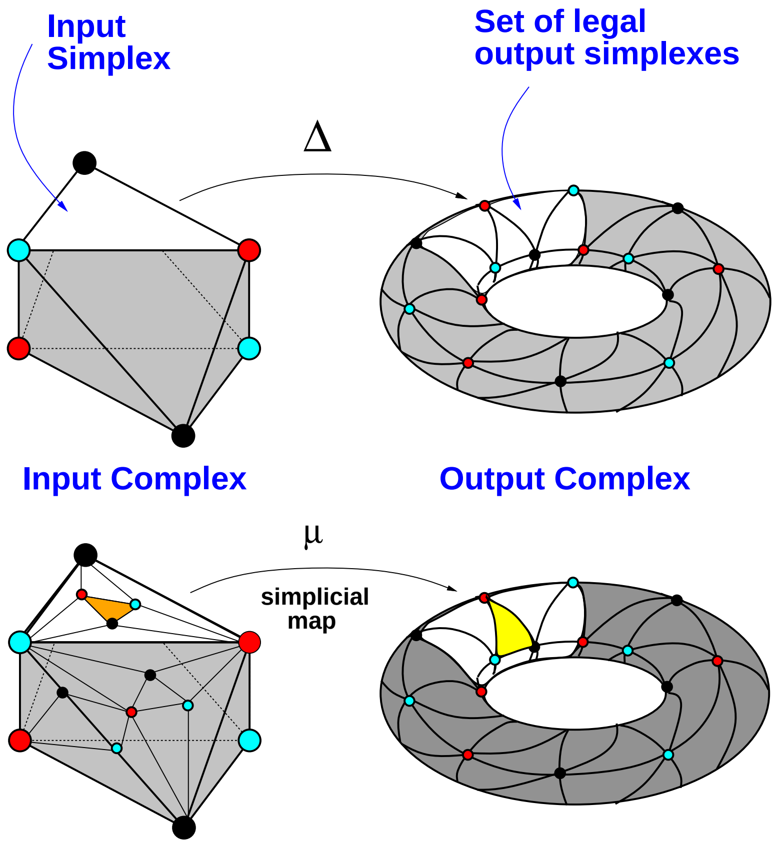

Figure 12: Asynchronous Computability Theorem

4.1 Binary Consensus

We now return to the binary consensus problem mentioned in Section 2. Although it is well known that this problem has no wait-free read-write protocol [15, 17, 35], we believe the reader's intuition will benefit from a brief discussion of how this result follows from the main theorem. To keep our presentation as simple as possible, we focus on the two-process task.

Processes P and Q are given private binary inputs, and they must agree on one of their inputs. In a solo execution, where P runs alone, it observes only its own input, say 0. Since P cannot wait indefinitely for Q to take a step, it must eventually decide 0. The same is true for Q running solo with input 1. However, if P and Q run together, then one of them, say P, must change its tentative decision, while preventing Q from doing the same. At the heart of the published impossibility results for this task is a case analysis showing that the commuting and overwriting properties of read and write operations make this kind of synchronization impossible.

The asynchronous computability theorem captures this impossibility in a geometric way. Figure 9 shows the input and output complexes for the two-process consensus task. Assume by way of contradiction that a protocol exists. The input complex $\mathcal{I}^2$ is connected, and so is the subdivision $\sigma(\mathcal{I}^2)$. Simplicial maps preserve connectivity, so $\mu(\sigma(\mathcal{I}^2))$ is also connected. Let I_{ij} and O_{ij} denote the input and output simplexes where P has value i and Q has value j. Because $\Delta(I_{11}) = O_{11}$, μ carries input vertex $\langle P, 1\rangle$ to output vertex $\langle P, 1\rangle$. Symmetrically, it carries input $\langle Q, 0\rangle$ to output $\langle Q, 0\rangle$. However, these output vertexes lie in distinct connected components of the output complex, so μ cannot be a simplicial map, and by Theorem 4.1 2-process consensus is not solvable. (Generalizing this argument to n processes yields a simple geometric restatement of the impossibility of wait-free consensus in read/write memory [15, 17, 35].)

4.2 Quasi-Consensus

We now use another "toy" problem to illustrate the implications of the theorem. Let us relax the conditions of the consensus task as follows:

Quasi-Consensus Each of P and Q is given a binary input. If both have input v, then both must decide v. If they have mixed inputs, then either they must agree, or Q may decide 0 and P may decide 1 (but not vice-versa).

Figure 14 shows the input and output complexes for the quasi-consensus task.

Is quasi-consensus solvable? It is easily seen that there is no simplicial map directly from the input complex to the output complex. Just as for consensus, the vertexes of input simplex I_{01} must map to output vertexes $\langle P, 0\rangle$ and $\langle Q, 1\rangle$, but there is no single output simplex containing both vertexes. Nevertheless, there is a map satisfying the conditions of the theorem from a *subdivision* of the input complex. If input simplex I_{01} is subdivided as shown in Figure 15, then it can be "folded" around the output complex, allowing input vertexes $\langle P, 0\rangle$ and $\langle Q, 1\rangle$ to be mapped to their counterparts in the output complex.

Figure 16 shows a simple protocol for quasi-consensus. Notice that if P has input 0 and Q has input 1, then this protocol admits three distinct executions: one in which both decide 0, one in which both decide 1, and one in which Q decides 0 and P decides 1. These three executions correspond to the three simplexes in the subdivision of I_{01}, which are carried to O_{00}, O_{10}, and O_{11}. In Section 5, where we prove that the conditions of the theorem are necessary, we will show that each simplex in the subdivision $\sigma(S^n)$ indeed corresponds to some execution of the protocol when started in the initial state S^n.

$\vec{I}$	$\Delta(\vec{I})$	$\vec{I}$	$\Delta(\vec{I})$
$(0, \perp)$	$(0, \perp)$	$(1, \perp)$	$(1, \perp)$
$(\perp, 0)$	$(\perp, 0)$	$(\perp, 1)$	$(\perp, 1)$
$(0, 0)$	$(0, 0)$	$(1, 1)$	$(1, 1)$
$(0, 1)$	$(0, 0), (1, 1), (1, 0)$	$(1, 0)$	$(0, 0), (1, 1), (1, 0)$

Figure 13: The Quasi-Consensus task

4.3 Set Agreement

The same intuition can be extended to n-dimensional task specifications. Recall that in the two-process (one-dimensional) case, the impossibility of consensus follows from the observation that a simplicial map cannot carry a connected component of the subdivided input complex to disconnected components of the output complex. In the $(n+1)$-process (n-dimensional) case, the impossibility of set agreement follows from an analogous observation: a simplicial map cannot carry the boundary of a "solid" disk to the boundary of a "hole". We now show that the k-set agreement task has no wait-free read-write protocol when $k \leq n$. Our proof uses Sperner's Lemma [33, Lemma 5.5]:

Lemma 4.2 (Sperner's Lemma) *Let $\sigma(S^n)$ be a subdivision of simplex S^n. If $F : \sigma(S^n) \to S^n$ is a map sending each vertex of $\sigma(S^n)$ to a vertex in its carrier, then there is at least one n-simplex $T^n = (\vec{t}_0, \ldots, \vec{t}_n)$ in $\sigma(S^n)$ such that the $F(\vec{t}_i)$ are all distinct.*

Lemma 4.3 *The k-set agreement task has no wait-free read-write protocol for $k \leq n$.*

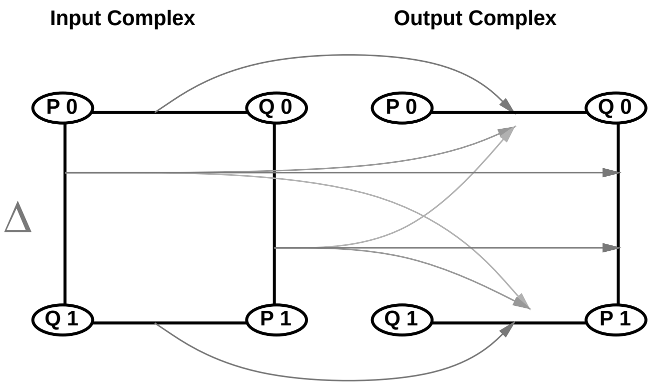

Figure 14: Input and Output Complexes for 2-Process Quasi-Consensus

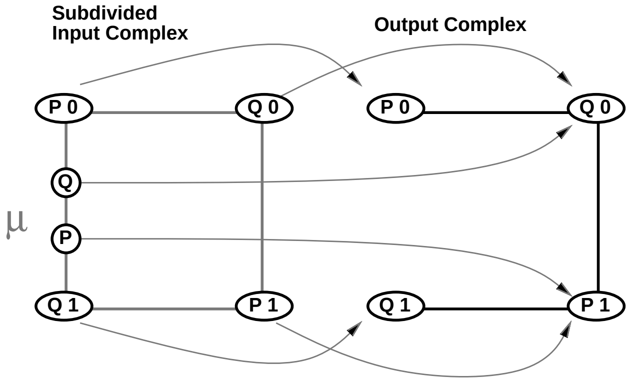

Figure 15: Protocol and Output Complexes for 2-Process Quasi-Consensus

```
initially input[P] = nil              initially input[Q] = nil
input[P] := my_input                  input[Q] := my_input
if my_input = 1 then decide 1         if my_input = 0 then decide 0
if input[Q] != 1 then decide 0        if input[P] != 0 then decide 1
decide 1                              decide 0
```

Figure 16: Quasi-Consensus Protocols for P and Q

Proof: It suffices to prove that there is no protocol for $k = n$. Assume otherwise. Let T^n be an input simplex where each P_i has a distinct input value v_i. For each $M \subseteq vals(T^n)$, where $|M| = m+1$, T^n has a face T^m_M such that $vals(T^m_M) = M$. Moreover, there is an input simplex S^n_M containing T^m_M such that $vals(S^n_M) = M$. (For example, if T^m_M is a single vertex $\langle P, 0\rangle$, then S^n_M is the input simplex where every process has input 0.)

By Theorem 4.1 there is a chromatic subdivsion σ and a color-preserving simplicial map $\mu : \sigma(\mathcal{I}^n) \rightarrow \mathcal{O}^n$, carrying each $\sigma(S^n_M)$ to $\Delta(S^n_M)$. The set agreement task specification states that $vals(\Delta(S^n_M)) = vals(S^n_M) = M$. Because $\sigma(T^m_M) \subseteq \sigma(S^n_M)$, μ also carries each $\sigma(T^m_M)$ to $\Delta(S^n_M)$, so each vertex in $\mu(\sigma(T^m_M))$ is labeled with a value from M.

Define the map $F : \sigma(T^n) \rightarrow T^n$ to be $F(\vec{s}) = \langle P_j, v_j\rangle$, where $val(\mu(\vec{s})) = v_j$. F carries each vertex in $\sigma(T^m_M)$ to a vertex in its carrier T^m_M. By Lemma 4.2 (Sperner's Lemma), F carries some n-simplex S^n in $\sigma(T^n)$ to T^n itself, implying that $\mu(S^n)$ is labeled with $n+1$ distinct values, which is a contradiction because every output simplex is labeled with at most n distinct values. ∎

Informally, this proof shows that the task definition requires mapping the subdivided faces of a simplex T^n around the "hole" in the output complex (created by the missing $n+1$-valued simplexes). Because μ is simplicial, it must map some simplex onto the hole itself, a contradiction.

5 Necessity

In this section, we show that the conditions of our theorem are necessary.

5.1 Protocol Complexes

Recall that at each step in a protocol, a process's *local state* is its input value followed by the sequence of values it has scanned. The *global state* of the protocol at any point is just the set of local states together with the states of the shared memory variables.

We can treat any protocol as an "uninterpreted" protocol simply by treating each processor's final local state as its decision value (i.e., omitting the task-specific decision map δ). This uninterpreted protocol itself defines a *protocol complex* $\mathcal{P}^n$ as follows: each vertex $\vec{v} \in \mathcal{P}^n$ is labeled with a process id and a local state such that there is some execution of the protocol in which process $id(\vec{v})$ finishes the protocol with local state $val(\vec{v})$. A simplex $T^m = (\vec{t}_0, \ldots, \vec{t}_m)$ is in $\mathcal{P}^n$ if there is an execution of the protocol in which each process $id(\vec{t}_i)$ finishes with local state $val(\vec{t}_i)$. Note that such an execution need not be a solo execution by the processes in $ids(T^m)$. For any input simplex S^m, let $\mathcal{P}(S^m)$ be the subcomplex of $\mathcal{P}^n$ generated by solo executions of the processes in $ids(S^m)$ with input values $vals(S^m)$. If $\mathcal{C} \subset \mathcal{I}^n$, then $\mathcal{P}(\mathcal{C})$ is defined analogously.

The protocol complex satisfies some useful functorial properties, which follow immediately from the definitions.

Lemma 5.1 $\mathcal{P}(\bigcap_{i=0}^m S_i) = \bigcap_{i=0}^m \mathcal{P}(S_i)$.

Lemma 5.2 $\mathcal{P}(\bigcup_{i=0}^m S_i) = \bigcup_{i=0}^m \mathcal{P}(S_i)$.

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. A protocol solves a decision task $\langle \mathcal{I}^n, \mathcal{O}, \Delta \rangle$ if and only if there exists a simplicial *decision map*

$$\delta : \mathcal{P}^n \to \mathcal{O}^n,$$

such that for all $S^m \in \mathcal{I}^n$, and all $T^m \in \mathcal{P}(S^m)$, $\delta(T^m) \in \Delta(S^m)$. 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 problem specification. This might seem like a roundabout formulation, 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 expressed in terms of relations among topological spaces. This formulation allows us to exploit classical results from the rich literature on algebraic and combinatorial topology.

5.2 Example: A One-Round Protocol Complex

Figure 5.2 shows the protocol complex for a simple one-round wait-free normal-form protocol. The processes share a three-element array with each entry initialized to $\perp$. Each process P, Q, and R writes its input value (respectively 0, 1, and 2) to its entry, scans the array's values, and halts. This complex has an inductive structure. The vertex at the top "corner" represent a solo execution by P: it writes 0, scans the array, and observes only its own value. The vertexes along the left-hand edge represent solo executions by P and Q: at the two vertexes in the middle of the edge, P and Q each writes its value, scans the array, and observes the other's value. The three vertexes in the interior of the complex represent executions in

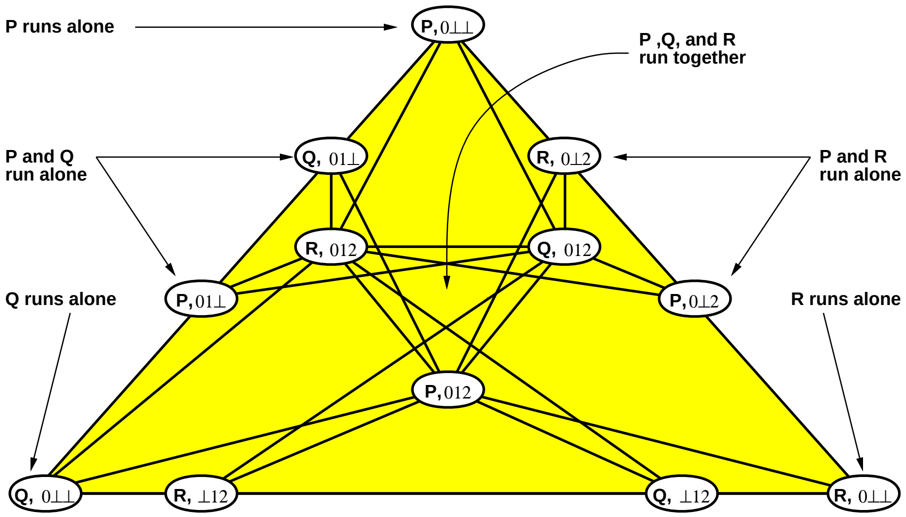

Figure 17: A One-Round Protocol Complex

which all processes' steps are interleaved: each process observes each of the other's values.

Our proof strategy is as follows. For each input simplex S^n, we identify the subdivision $\sigma(S^n)$ with a subcomplex of $\mathcal{P}(S^n)$, and construct the simplicial map μ in terms of the decision map δ. Section 5.4 provides the first step, showing that $\mathcal{P}(S^m)$ is n-connected (it has no holes). In Section 5.5 we show that the link of any k-simplex in $\mathcal{P}(S^m)$ is $(m-k-2)$-connected, and we complete the proof in Section 5.6.

5.3 Basic Lemmas

We begin with some general lemmas about simplicial complexes.

Definition 5.1 Complexes $\mathcal{C}_0, \ldots, \mathcal{C}_n$ *cover* $\mathcal{C}$ if $\mathcal{C} = \cup_{i=0}^{n} \mathcal{C}_i$. For any index set U, define $\mathcal{C}_U = \cap_{i \in U} \mathcal{C}_i$.

Lemma 5.3 *If $\mathcal{C}_0, \cdots, \mathcal{C}_n$ cover $\mathcal{C}$, then for any index sets U and V,*

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

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

In the following lemmas, let $U_0, \ldots, U_m$ be index sets of size u, such that for each distinct U_i and U_j, $\{P_i\} = U_i - U_j$, and let $V_0, \ldots, V_{\ell-1}$ be index sets such that $V_i = U_\ell \cup U_i$. Notice that each $|V_i| = u + 1$, for each distinct V_i and V_j, $\{i\} = V_i - V_j$. Moreover,

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

Lemma 5.4 *If $\mathcal{C}_0, \ldots, \mathcal{C}_n$ cover $\mathcal{C}$, and there exists $k > 0$ such that*

$$\textit{for all } U \textit{such that } u \leq k, \mathcal{C}_U \neq \emptyset, \tag{2}$$

and

$$\textit{for all } U \textit{such that } u < k, \mathcal{C}_U \textit{ is } (k - u - 1)\textit{-acyclic}, \tag{3}$$

then for all ℓ,

$$\bigcup_{i=0}^{\ell} \mathcal{C}_{U_i} \textit{ is } (k - u - 1)\textit{-acyclic.}$$

Proof: It follows immediately from Hypothesis 2 that for $u \leq k$,

$$\bigcup_{i=0}^{\ell} \mathcal{C}_{U_i} \neq \emptyset \tag{4}$$

We proceed by reverse induction on u, starting with $u = k - 1$, and then by induction on ℓ. When $\ell = 0$, the claim follows immediately from Hypothesis 3, so assume the claim for $\ell - 1$. For brevity, let $\mathcal{A} = \mathcal{C}_{U_\ell}$ and $\mathcal{B} = \cup_{i=0}^{\ell-1} \mathcal{C}_{U_i}$. By Equation 1, $\mathcal{A} \cap \mathcal{B} = \cup_{i=0}^{\ell-1} \mathcal{C}_{V_i}$, which is non-empty by Equation 4. We may therefore apply Theorem 2.4 (Mayer-Vietoris) for $q = 0$. By Hypothesis 2, $H_0(\mathcal{A}) = H_0(\mathcal{C}_{U_\ell}) = 0$, and by the induction hypothesis for ℓ, $H_0(\mathcal{B}) = H_0(\cup_{i=0}^{\ell-1} \mathcal{C}_{U_i}) = 0$. $H_{-1}(\mathcal{A} \cap \mathcal{B}) = 0$ by definition, so by Theorem 2.4, $H_0(\mathcal{A} \cup \mathcal{B}) = H_0(\bigcup_{i=0}^{\ell} \mathcal{C}_{U_i}) = 0$.

For the induction step, assume the claim for index sets of size $u + 1$. As before, we proceed by induction on ℓ. When $\ell = 0$, the claim follows immediately from Hypothesis 3, so assume the claim for $\ell - 1$. We apply the Mayer-Vietoris theorem for $\mathcal{A}$ and $\mathcal{B}$. By Equation 3, for $q \leq k - u - 1$, $H_q(\mathcal{A}) = H_q(\mathcal{C}_{U_\ell}) = 0$, and by the induction hypothesis for ℓ,

$$H_q(\mathcal{B}) = H_q(\bigcup_{i=0}^{\ell-1} \mathcal{C}_{U_i}) = 0.$$

By Equation 1 and the induction hypothesis for u:

$$H_{q-1}(\mathcal{A} \cap \mathcal{B}) = H_{q-1}(\mathcal{C}_{U_\ell} \cap (\bigcup_{i=0}^{\ell-1} \mathcal{C}_{U_i})) = H_{q-1}(\bigcup_{i=0}^{\ell-1} \mathcal{C}_{V_i}) = 0 \text{ for } q \leq k - u.$$

The claim then follows from the Mayer-Vietoris theorem. ∎

When $u = 1$ and $\ell = n$, we have the following corollary.

Corollary 5.5 *If $\mathcal{C}$ is covered by $\mathcal{C}_0, \ldots, \mathcal{C}_n$ satisfying the conditions of Lemma 5.4, then $\mathcal{C}$ is $(k-2)$-acyclic.*

We now prove some basic lemmas about simple connectivity. The next lemma is left as an exercise for the reader.

Lemma 5.6 *If $\mathcal{A}$ and $\mathcal{B}$ are connected, and $\mathcal{A} \cap \mathcal{B}$ is non-empty, then $\mathcal{A} \cup \mathcal{B}$ is connected.*

Lemma 5.7 *Let $\mathcal{C}$ be covered by $\mathcal{C}_0, \ldots, \mathcal{C}_n$. Define $\mathcal{C}_{ij} = \mathcal{C}_i \cap \mathcal{C}_j$, and $\mathcal{C}_{ijk} = \mathcal{C}_i \cap \mathcal{C}_j \cap \mathcal{C}_k$. If (1) each $\mathcal{C}_i$ is simply connected, (2) each $\mathcal{C}_{ij}$ is connected, and (3) each $\mathcal{C}_{ijk}$ is non-empty, then $\mathcal{C}$ is simply connected.*

Proof: Hypothesis (3) implies that for $u = 3$,

$$\bigcup_{i=0}^{\ell} \mathcal{C}_{U_i} \neq \emptyset. \tag{5}$$

We now claim that for $u = 2$,

$$\bigcup_{i=0}^{\ell} \mathcal{C}_{U_i} \text{ is connected.} \tag{6}$$

We proceed by induction on ℓ. When $\ell = 0$, connectivity follows immediately from Hypothesis (2), so assume connectivity for $\ell - 1$. By Equation 1 and Equation 5, $\mathcal{C}_{U_\ell} \cap (\cup_{i=0}^{\ell-1} \mathcal{C}_{U_i})$ is non-empty. Hence by Lemma 5.6, $\cup_{i=0}^{\ell} \mathcal{C}_{U_i}$ is connected. Finally, we claim that

$$\bigcup_{i=0}^{\ell} \mathcal{C}_i \text{ is simply connected.} \tag{7}$$

We proceed by induction on ℓ. When $\ell = 0$, simple connectivity follows from Hypothesis (1), so assume simple connectivity for $\ell - 1$. By Equation 1 and Equation 6, $\mathcal{C}_{U_\ell} \cap (\cup_{i=0}^{\ell-1} \mathcal{C}_{U_i})$ is connected, and Equation 7 follows from Theorem 2.3 (Siefert/Van Kampen). The claim follows by setting $u = 1$ and $\ell = n$ in Equation 7. ∎

5.4 Properties of Input Simplexes

We now prove an important global property of protocol complexes: for any input simplex S^m, the corresponding protocol complex $\mathcal{P}(S^m)$ is m-connected.

The set of executions starting in any global state define a *reachable complex* as follows.

Definition 5.2 A simplex R^m of the protocol complex $\mathcal{P}(S^m)$ is *reachable from global state s* if there is some execution starting from s in which each process in $ids(R^m)$ completes the protocol with the local state specified in R^m. The *reachable complex* from state s, written $\mathcal{R}(s)$, is the complex of reachable simplexes from s. Notice that the reachable complex from initial state S^n is just $\mathcal{P}(S^n)$.

If s is a global state in which not all processes have decided, then processes fall into two categories: (1) a *pending* process is about to execute an operation, and (2) a *decided* process has completed its protocol and halted. For a pending process P_i, define $\mathcal{R}_i(s)$ to be the reachable complex after P_i executes its pending operation. As i ranges over the pending processes, the $\mathcal{R}_i(s)$ cover $\mathcal{R}(s)$. A *pending index set* is a set of indexes of pending processes. If U is a pending index set, define $\mathcal{R}_U(s) = \cap_{i \in U} \mathcal{R}_i(s)$. Lemma 5.3 applies. Informally, each simplex in $\mathcal{R}_U(s)$ corresponds to an execution starting in s in which no process can tell which process in U went first.

A global state s is *critical* for a property $\wp$ if $\wp$ does not hold in s, and a step by any pending process will bring the protocol to a state where $\wp$ henceforth holds.

Lemma 5.8 *If $\wp$ is a property that does not hold in some state s and holds in every final state of a protocol, then $\wp$ has a critical state.*

Proof: A process is *non-critical* if its next step will not make $\wp$ henceforth hold. Starting from state s, pick a non-critical pending process and run it until a critical state is reached. Repeat this process until a critical state is reached in which there are no non-critical pending processes, that is, all processes are either decided or about to make $\wp$ henceforth true in their next step. Since a process that has completed the protocol will never make $\wp$ true, and since $\wp$ must become true within a finite number of steps, the protocol must reach such a critical state. ∎

We will now show that $\mathcal{P}(S^m)$ satisfies the conditions of Lemma 5.4. Informally stated, our proof strategy is the following argument by contradiction. Assume that initially the claim does not hold. Since the reachable complex eventually shrinks to a single simplex, it eventually satisfies the desired properties, so we can by Lemma 5.8 run the protocol to a critical state. We then analyze the possible interactions of the pending operations to show that the reachable complex must have satisfied the conditions to begin with, yielding a contradiction.

Lemma 5.9 *For all pending index sets U, $\mathcal{R}_U(s)$ is non-empty.*

Proof: If all pending operations in state s are scans, $\mathcal{R}_U(s)$ includes the simplex corresponding to one process executing solo from s. If there are processes with pending update operations, $\mathcal{R}_U(s)$ includes the simplex corresponding to the execution in which these processes execute the updates in some order and then one updating process executes solo until it decides. Either way, the resulting state is the same regardless of the order in which the pending update operations are executed, since updates to disjoint variables commute and the state does not reflect the results of the pending scans. ■

Lemma 5.10 *For every global state s and pending index set U, $\mathcal{R}_U(s)$ is non-empty and acyclic.*

Proof: By way of contradiction, assume the claim does not hold in some state and pick an $(m+1)$-process protocol Π such that in some critical state s $\mathcal{R}_U(s)$ is not acyclic. Such a critical state s exists because the reachable complex is eventually a single simplex, which is non-empty and acyclic (Lemma 2.5). Choose Π so that m is minimal. If any process is decided in s, then every simplex in every $\mathcal{R}_U(s)$ includes that process's vertex, implying that $\mathcal{R}_U(s)$ is a cone, which is acyclic by Lemma 2.5, a contradiction.

Each simplex in $\mathcal{R}_U(s)$ is reachable by an execution in which all pending operations in U occur before any other operation. Pick a canonical order for the pending operations in U, and let s' be the state reached by executing the pending operations in that order.

If the pending operations all commute, then any execution leading to a simplex in $\mathcal{R}_U(s')$ is equivalent to one in which the pending operations execute in the canonical order, thus $\mathcal{R}_U(s') = \mathcal{R}(s)$, the reachable complex for Π from s. Since s is critical, the first pending operation executed ensures that all reachable complexes are henceforth acyclic. So $\mathcal{R}_U(s)$ is acyclic.

Otherwise, if operations fail to commute, both scans and updates must be pending. Let R be the index set of scanning processes. Clearly, $\mathcal{R}_U(s)$ contains no vertexes of processes in R, since any such process observes (and thus writes into its final local state) whether its scan preceded pending updates. $\mathcal{R}_U(s)$ is thus the reachable complex from s for the $(m+1-|R|)$-process protocol identical to Π except that the processes in R do not participate. Because $|R|>0$ and m is minimal, $\mathcal{R}_U(s)$ is acyclic. ∎

Lemma 5.11 *In any global state, the reachable complex is acyclic.*

Proof: In any global state s, $\mathcal{R}_U(s)$ satisfies the conditions of Lemma 5.4 for $k=n+2$. Equation 2 is satisfied vacuously for $k=n+2$ (since there are no index sets of size $n+2$), and is satisfied for smaller values of k by Lemma 5.9. Equation 3 is satisfied by Lemma 5.10. The claim follows by setting $u=1$. ∎

Corollary 5.12 *For all $S^m \in \mathcal{I}^n$, $\mathcal{P}(S^m)$ is acyclic.*

Lemma 5.13 *For all input simplexes S^m, $\mathcal{P}(S^m)$ is simply connected.*

Proof: It suffices to prove that in every global state, the reachable complex is simply connected. Suppose not. Since the final state is simply connected, there exists a critical state s. By Lemma 5.10, each $H_q(\mathcal{R}_{ij})=0$, meaning that each $\mathcal{R}_{ij}$ is connected. By Lemma 5.9, each $\mathcal{R}_{ijk}$ is non-empty. The claim follows from Lemma 5.4. ∎

Lemma 5.14 *For any input simplex S^m, $\mathcal{P}(S^m)$ is m-connected.*

Proof: $\mathcal{P}(S^m)$ is simply connected (Lemma 5.13) acyclic (Corollary 5.12), so the claim follows immediately from Theorem 2.2 (Hurewicz). ∎

5.5 Properties of Protocol Complexes

The previous section's results suffice to show that if a protocol solves a task, then there exists a subdivision $\sigma(\mathcal{I}^n)$ and a simplicial map $\mu:\ \sigma(\mathcal{I}^n)\ \rightarrow\ \mathcal{P}^n$, but we need to prove some additional properties of $\mathcal{P}^n$ before we can show that this map is chromatic.

Our approach will be to build up σ and ϕ by induction on the skeleton of $\mathcal{I}^n$, each time building the next dimension of the subdivision and map based on the σ and ϕ defined for all lower dimensions. We need to show that the simplicial map ϕ can be extended to higher dimensions in a way that does not cause it to "collapse" simplexes, that is, send higher-dimensional input simplexes to lower-dimensional output simplexes. This property suffices to make the simplicial map chromatic because each vertex in the subdivided input complex can just be colored with the id of its image.

The first step in showing this "non-collapsing" property is to show that in addition to being n-connected (Lemma 5.14), we must show that it is *link-connected*.

Definition 5.3 An n-complex $\mathcal{C}^n$ is *link-connected* if for all simplexes $T^m \in \mathcal{C}^n$, $lk(T^m, \mathcal{C}^n)$ is $(n-m-2)$-connected.

Lemma 5.15 *If $\mathcal{C}^n$ is link-connected, so is $lk(T^m, \mathcal{C}^n)$ for all $T^m \in \mathcal{C}^n$.*

Proof: $lk(T^p, lk(T^m, \mathcal{C}^n)) = lk(T^p \cdot T^m, \mathcal{C}^n)$. ∎

Lemma 5.16 *For every input simplex S^n, and output simplex $T^m \in \mathcal{P}(S^n)$, $lk(T^m, \mathcal{P}(S^n))$ is $(n-m-2)$-acyclic.*

Proof: The proof is nearly the same as the proof of Lemma 5.10, except that we restrict attention to executions leading to output simplexes containing T^m. In any state where T^m is in the reachable complex, a process is *enabled* if executing its pending operation leaves T^m within the reachable complex, and *disabled* otherwise. An execution is *enabled* if it consists entirely of enabled operations.

Following Lemma 5.8, any $\wp$ that does not hold in some state but holds in every final state reachable by an enabled execution has a *critical state* in which $\wp$ does not hold, but any enabled operation will make $\wp$ henceforth true for enabled executions.

Let s be a state reached by an enabled execution, P_i an enabled process in s, and U a set of enabled processes. Define $\mathcal{Q}(s) = lk(T^m, \mathcal{R}(s))$, $\mathcal{Q}_i(s) = lk(T^m, \mathcal{R}_i(s))$, and $\mathcal{Q}_U(s) = lk(T^m, \mathcal{R}_U(s))$, where $\mathcal{R}$, $\mathcal{R}_i$, and $\mathcal{R}_U$ are complexes from the proof of Lemma 5.10. The $\mathcal{Q}_i(s)$ cover $\mathcal{Q}(s)$, and Lemma 5.3 applies.

Let s be a critical state for the property "$\mathcal{Q}(s)$ is $(n-m-2)$-acyclic". As in the proof of Lemma 5.10, if the enabled operations are all scans or all updates, then they commute, and $\mathcal{Q}(s)$ is already $(n-m-2)$-acyclic. So both scans and updates must be pending. Let R be the set of process ids with pending scans in s. R and $ids(T^m)$ are disjoint, because the final state of any process in R depends on when its scan occurred with respect to enabled updates, but that state is fixed for processes in $ids(T^m)$. So $R \subset \overline{ids(T^m)}$, the complement of $ids(T^m)$. For the same reason, $ids(\mathcal{Q}_U(s))$ and R are also disjoint,

The *commit point* of an enabled execution occurs immediately after the last scan by any process in $ids(T^m)$. If P_j is in R, then its next update will not be enabled until after the commit point, because otherwise it would affect the scans of $ids(T^m)$. Following a commit point, the final local states of $ids(T^m)$ are fixed, so all pending operations are henceforth enabled.

We claim that if $|U| \leq n - m$, then $\mathcal{Q}_U(s)$ is non-empty. If $|U| \leq n - m$, then $|R| \leq n - m - 1$, so there exists a P_i in $\overline{ids(T^m)}$ but not in R. We now construct an enabled execution in which no process in R takes any steps, but P_i eventually halts with a decision value. As noted above, any enabled execution from s leaves processes in R disabled until the commit point. After the commit point P_i becomes enabled, and we can run it solo until it halts with an output value.

We claim that $\mathcal{Q}_U(s)$ is $(n - m - u - 1)$-acyclic. $\mathcal{Q}_U(s)$ is $\mathcal{Q}(s)$ for the $(n + 1 - |R|)$-process protocol identical to Π except that the processes in R do not participate. The values of n and u are both reduced by $|R| > 0$ and n is minimal, so $\mathcal{R}_U(s)$ is $(n - m - u - 1)$-acyclic.

Finally, we observe that the $\mathcal{Q}_U(s)$ satisfy the conditions of Lemma 5.4 for $k = n - m$. The claim follows from Corollary 5.5. ∎

Lemma 5.17 *For every input simplex S^n, and output simplex $T^m \in \mathcal{P}(S^n)$, if $n - m > 1$, $lk(T^m, \mathcal{P}(S^n))$ is simply connected.*

Proof: The proof is nearly identical to the proof of Lemma 5.16, except substituting simple connectivity for acyclicity, and Lemma 5.7 for Lemma 5.4. ∎

From Lemmas 5.16 and 5.17 and Theorem 2.2 (Hurewicz), it follows that:

Corollary 5.18 *$\mathcal{P}(S^m)$ is link-connected.*

5.6 Spans

Definition 5.4 Let $\mathcal{A} \subset \mathcal{B}$, and $\phi : \mathcal{A} \to \mathcal{C}$. A simplicial map $\psi : \mathcal{B} \to \mathcal{C}$ *extends* ϕ if they agree on $\mathcal{A}$.

The following lemma appears in Glaser [22, Theorem IV.2].

Lemma 5.19 *Let $\mathcal{A}$, $\mathcal{B}$, and $\mathcal{C}$ be complexes such that $\mathcal{A} \subset \mathcal{B}$, and $f : |\mathcal{B}| \to |\mathcal{C}|$ is a continuous map such that f restricted to $|\mathcal{A}|$ is simplicial. There exists a subdivision τ of $\mathcal{B}$ such that $\tau(\mathcal{A}) = \mathcal{A}$, and a simplicial map $\phi : \tau(\mathcal{B}) \to cC$ extending the restriction of f to $|\mathcal{A}|$.*

Lemma 5.20 *Let $\mathcal{A}$ be an $(m-1)$-sphere and $\phi : \mathcal{A} \to \mathcal{C}$ a simplicial map where $\mathcal{C}$ is $(m-1)$-connected. $\mathcal{A}$ is the boundary complex of an m-disk $\mathcal{B}$ with a simplicial map $\psi : \mathcal{B} \to \mathcal{C}$ extending ϕ.*

Proof: Pick a vertex $\vec{v}$ affinely independent of any vertex in $\mathcal{A}$, and let $\mathcal{A}'$ be the complex $\vec{v} \cdot \mathcal{A}$. Because $\mathcal{A}'$ is a disk, and $\mathcal{C}$ is $(m-1)$-connected, ϕ can be extended to a continuous $f : |\mathcal{A}'| \to |\mathcal{C}|$ simplicial on $\mathcal{A}$. The claim then follows from Lemma 5.19. ∎

Definition 5.5 A simplicial map $\phi : \mathcal{B} \to \mathcal{C}$ *collapses* a simplex T^m, $m > 0$, if $\dim(\phi(T^m)) = 0$.

Lemma 5.21 *Let $\mathcal{A}^{m-1}$ be an $(m-1)$-sphere, and $\phi : \mathcal{A}^{m-1} \to \mathcal{C}^n$ a simplicial map where $\mathcal{C}^n$ is $(m-1)$-connected and link-connected. $\mathcal{A}^{m-1}$ is the boundary complex of an m-disk $\mathcal{B}^m$ with a simplicial map $\psi : \mathcal{B}^m \to \mathcal{C}^n$ extending ϕ such that ψ collapses no internal simplexes of $\mathcal{B}$.*

Proof: Because $\mathcal{C}^n$ is $(m-1)$-connected, Lemma 5.20 implies that there exists an m-disk $\mathcal{B}^m$ with boundary complex $\mathcal{A}^{m-1}$ and a simplicial map $\psi : \mathcal{B}^m \to \mathcal{C}^n$ extending ϕ. Let n be the least dimension for which the claim fails, and let $\psi : \mathcal{B}^m \to \mathcal{C}^k$ be the map that collapses the fewest internal simplexes of maximum dimension. We will argue by contradiction that this number must be zero.

Suppose ψ collapses an internal $T^m \in \mathcal{B}^m$ to $\vec{v}$. Let $\vec{t}$ be the barycenter of T^m, $\mathcal{T}^m$ the complex of faces of T^m, and $\dot{\mathcal{T}}^{m-1}$ the complex of proper faces of T^m. Define $\tau(\mathcal{T}^m)$ to be $\vec{t} \cdot \dot{\mathcal{T}}^{m-1}$, the complex constructed by joining $\vec{t}$ with the proper faces of T^m. This complex is a subdivision of $\mathcal{T}^m$ [40, 3.3.8] that leaves the boundary complex unchanged, so replacing T^m in $\mathcal{B}^m$ with $\tau(\mathcal{T}^m)$ yields a subdivision $\tau(\mathcal{B}^m)$. Pick $\vec{u}$ such that $(\vec{v}, \vec{u})$ is a 1-simplex in $\mathcal{C}^n$. Define $\psi' : \tau(\mathcal{B}^m) \to \mathcal{C}^n$ such that $\psi'(\vec{t}) = \vec{u}$, and elsewhere $\psi' = \psi$. The complex $\tau(\mathcal{B}^m)$ and map ψ' satisfy our conditions, but collapse one fewer m-simplex, a contradiction.

Suppose ψ collapses an internal $T^p \in \mathcal{B}^m$, $p < m$, to $\vec{v}$, but collapses no internal simplexes of higher dimension. Let $\vec{t}$ be the barycenter of T^p, $\mathcal{T}^p$ the complex of faces of T^p, and $\dot{\mathcal{T}}^{p-1}$ the complex of proper faces of T^p. The vertexes of any $R^{m-p-1} \in lk(T^p, \mathcal{B}^m)$ are affinely independent of the vertexes of T^p, so $\vec{t}$ is affinely independent of each R^{m-p-1}. Moreover, each $\vec{t} \cdot R^{m-p-1}$ is a simplex, and $\vec{t} \cdot lk(T^p, \mathcal{B}^m)$ is a complex. Because $\mathcal{B}^m$ is a manifold with boundary, Lemma 2.1 implies that $lk(T^p, \mathcal{B}^m)$ is a $(m-p-1)$-sphere, and hence $\vec{t} \cdot lk(T^p, \mathcal{B}^m)$ is an $(m-p)$-disk. Because ψ does not collapse any $(p+1)$-simplexes, ψ does not send any vertex of $lk(T^p, \mathcal{B}^m)$ to $\vec{v}$, so $\psi : lk(T^p, \mathcal{B}^m) \to lk(\vec{v}, \mathcal{C}^n)$. Notice that $lk(\vec{v}, \mathcal{C}^n)$ is an $(n-1)$-complex that is $(m-2)$-connected because $\mathcal{C}^n$ (for our least dimension n) is link-connected by Lemma 5.15.

Because n is minimal by hypothesis, the conditions of the lemma apply, and there exists an $(m-p)$-disk $\mathcal{E}^{m-p}$ having $lk(T^p, \mathcal{B}^m)$ as boundary complex, and a map $\psi' : \mathcal{E}^{m-p} \to lk(\vec{v}, \mathcal{C}^n)$ that extends ψ and collapses no internal simplex of $\mathcal{E}^{m-p}$. Define $\tau(\mathcal{T}^p)$ to be $\mathcal{E}^{m-p} \cdot \dot{\mathcal{T}}^{p-1}$. This subdivision of $\mathcal{T}^p$ leaves the boundary complex unchanged, so replacing T^p in $\mathcal{B}^m$ with $\tau(\mathcal{T}^p)$ yields a subdivision $\tau(\mathcal{B}^m)$. Define $\psi' : \tau(\mathcal{B}^m) \to \mathcal{C}^n$ such that $\psi'(\vec{t}) = \vec{u}$, and elsewhere $\psi' = \psi$. Define $\psi' : \tau(\mathcal{B}^m) \to lk(\vec{v}, \mathcal{C}^n)$ by

$$\psi'(\vec{x}) = \begin{cases} \psi'(\vec{x}) & \vec{x} \in \tau(\mathcal{T}^p) \\ \psi(\vec{x}) & \text{otherwise} \end{cases}$$

The complex $\tau(\mathcal{B}^m)$ and map ψ' satisfy our conditions, but collapse one fewer p-simplex, a contradiction. ■

Definition 5.6 A *span* for $\mathcal{I}^n$ is a subdivision $\sigma(\mathcal{I}^n)$ and a color-preserving simplicial map $\phi : \sigma(\mathcal{I}^n) \to \mathcal{P}(\mathcal{I}^n)$, such that for every $S^\ell \in \mathcal{I}^n$, $\phi : \sigma(S^\ell) \to \mathcal{P}(S^\ell)$.

Theorem 5.22 *Every wait-free protocol has a span.*

Proof: We will build up σ and ϕ by induction on the skeleton of $\mathcal{I}^n$, as illustrated in Figure 18.

For each $\vec{v} \in \mathcal{I}^n$, define $\phi_0(\vec{v}) = \mathcal{P}(\vec{v})$, the unique vertex in the protocol complex corresponding to a solo execution of process $id(\vec{v})$ with input $val(\vec{v})$. This map collapses no simplexes, and is carrier preserving.

Assume inductively that we have a subdivision σ_{k-1} and a carrier-preserving simplicial map

$$\phi_{k-1} : \sigma_{k-1}(skel^{k-1}(\mathcal{I}^n)) \to \mathcal{P}(\mathcal{I}^n)$$

that collapses no simplexes. Let S^k be a simplex in $skel^k(\mathcal{I}^n)$, $\mathcal{S}^k$ the complex of its faces, and $\dot{\mathcal{S}}^{k-1}$ the complex of its proper faces. Because $\mathcal{P}(S^k)$ is k-connected (Lemma 5.14) and link-connected (Corollary 5.18), Lemma 5.21 implies that $\phi_{k-1} : \sigma_{k-1}(\dot{\mathcal{S}}^{k-1}) \to \mathcal{P}(\dot{\mathcal{S}}^{k-1})$ can be extended to $\phi_k : \sigma_k(\mathcal{S}^k) \to \mathcal{P}(S^k)$ so that on $\dot{\mathcal{S}}^{k-1}$, ϕ_k and ϕ_{k-1} agree, and so do σ_k and σ_{k-1}. The maps and subdivisions defined in this way agree on the intersections of their domains, so together they define a subdivision σ_{k+1} and a carrier-preserving simplicial map

$$\phi_k : \sigma(skel^k(\mathcal{I}^n)) \to \mathcal{P}(skel^k(\mathcal{I}^n))$$

that collapses no simplexes.

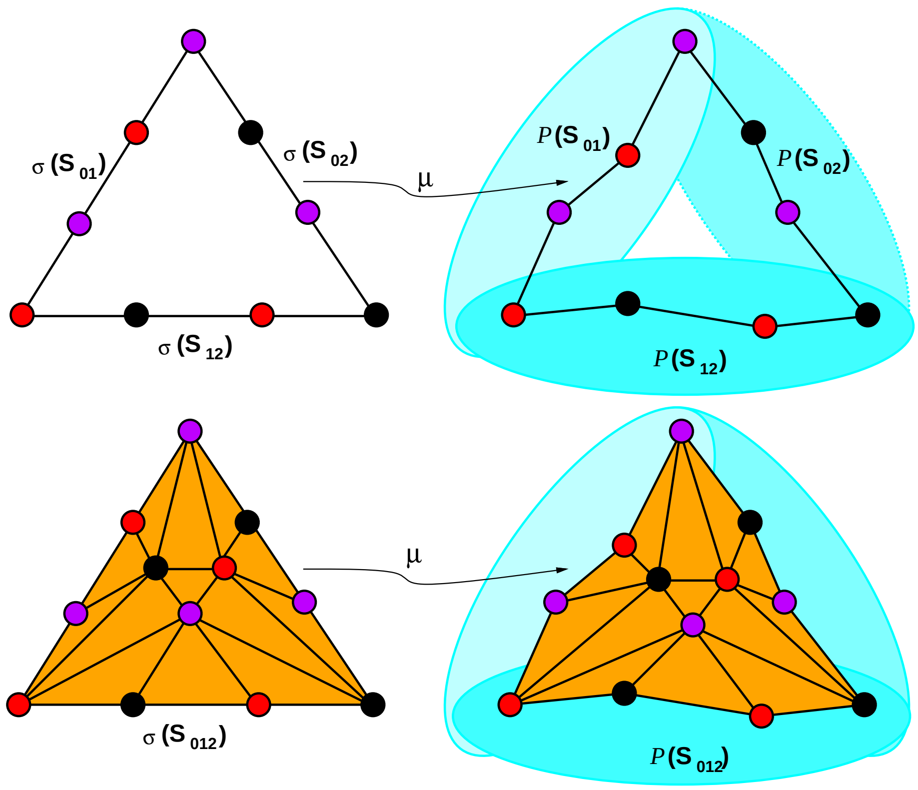

Figure 18: Inductive Span Construction

The desired subdivision σ is σ_n, and the desired map ϕ is ϕ_n. Because $\phi : \sigma(\mathcal{I}^n) \to \mathcal{P}(\mathcal{I}^n)$ collapses no vertexes, each $\vec{v} \in \sigma(\mathcal{I}^n)$ can be colored with $id(\phi(\vec{v}))$, implying that σ is a chromatic subdivision. ■

Theorem 5.23 *If a decision task* $\langle \mathcal{I}^n, \mathcal{O}^n, \Delta \rangle$ *has a wait-free solution using read-write memory, then there exists a chromatic subdivision* $\sigma(\mathcal{I}^n)$ *and a color-preserving simplicial map* $\mu : \sigma(\mathcal{I}^n) \to \mathcal{O}^n$ *such that for each simplex* S^m *in* $\sigma(\mathcal{I}^n)$, $\mu(S^m) \in \Delta(carrier(S^m))$.

Proof: Suppose a protocol exists. By Theorem 5.22, the protocol has a span. Let σ be the chromatic subdivision of $\mathcal{I}^n$ induced by the span, and let $\mu(\vec{v}) = \delta(\phi(\vec{v}))$, the composition of the span map and the decision map. ■

6 Sufficiency

In this section, we show how to use the conditions of the Asynchronous Computability Theorem to construct a protocol.

The basic intuition is that solving a decision task is really a form of *approximate agreement* [4, 18], in which processes start out preferring vertexes "far apart" on the output complex, but after a process of negotiation eventually converge to the vertexes of a single output simplex. Formally:

Definition 6.1 The *simplex agreement task* $\langle \mathcal{I}, \mathcal{O}, \Delta \rangle$ has an arbitrary input complex $\mathcal{I}$, output complex $\mathcal{O} = \sigma(\mathcal{I})$, a chromatic subdivision of $\mathcal{I}$, and for all input simplexes S^m, $\Delta(S^m)$ is the set of m-simplexes in $\sigma(S^m)$.

The conditions of Theorem 4.1 imply that any protocol that solves simplex agreement for an arbitrary chromatic subdivision of $\mathcal{I}$ is a universal protocol.

Our construction has the following structure.

- We introduce the *standard chromatic subdivision* of a complex $\mathcal{I}$, denoted $\chi(\mathcal{I})$, and the *iterated* standard chromatic subdivision $\chi^k(\mathcal{I})$. This subdivision is a color-preserving analogue of the classical barycentric subdivision.

- Simplex agreement on $\chi(\mathcal{I})$ is solved by the "participating set" protocol of Borowsky and Gafni [11]. Simplex agreement on $\chi^k(\mathcal{I})$ is solved by iterating that protocol k times.

- If $\sigma(\mathcal{I})$ is an arbitrary chromatic subdivision of $\mathcal{I}$, then there exists an integer k such that if $K > k$, there is a carrier-preserving simplicial map $\phi : \chi^K(\mathcal{I}) \to \sigma(\mathcal{I})$.

Putting these results together, we have a universal protocol. If the subdivision σ and simplicial map μ are given, then the value of K and the simplicial map ϕ may be computed off-line. The processes first solve simplex agreement on $\chi^K(\mathcal{I})$. A process that chooses vertex $\vec{v} \in \chi^K(\mathcal{I})$ then chooses as its output value $val(\mu(\phi(\vec{v})))$.

6.1 The Standard Chromatic Subdivision

We start with a purely combinatorial definition of the standard chromatic subdivision. (For an analogous combinatorial definition of the standard barycentric subdivision, see [38, Lemma 15.3].) Let $S^n = (\vec{s}_0, \ldots, \vec{s}_n)$, where $id(\vec{s}_i) = P_i$.

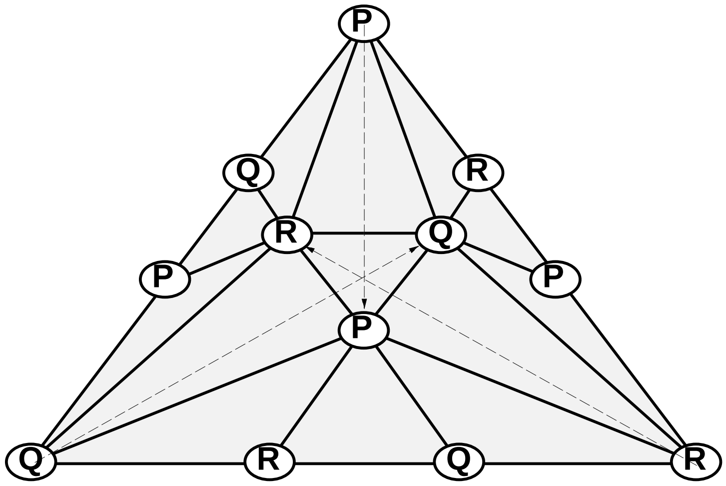

Figure 19: Standard Chromatic Subdivision

Definition 6.2 In the *standard chromatic subdivision* of S^n, denoted $\chi(S^n)$, each n-simplex has the form $(\langle P_0, S_0\rangle, \ldots, \langle P_n, S_n\rangle)$, where S_i is a face of S^n, such that (1) $P_i \in ids(S_i)$, (2) for all S_i and S_j, one is a face of the other, and (3) if $P_j \in ids(S_i)$, then $S_j \subseteq S_i$.

We refer to the $\vec{x}_i = \langle P_i, S^n\rangle$ as the *central vertexes* of the subdivision. The standard chromatic subdivision of S^2 is illustrated in Figure 19. Applying the standard chromatic subdivision repeatedly yields a subdivision $\chi^k(S^n)$. Applying it to every simplex in a complex $\mathcal{C}^n$ yields the complex $\chi^k(\mathcal{C}^n)$. $\chi(S^n)$ is a subdivision by the following homeomorphism

$$\iota: \ |\chi(S^n)| \ \rightarrow \ |S^n|.$$

Assume inductively that there exist homeomorphisms

$$\iota_i: \ |\chi(face_i(S^n))| \ \rightarrow \ |face_i(S^n)|.$$

Let $\vec{b} = \sum_{i=0}^{n}(\vec{s}_i/(n+1))$ be the barycenter of S^n, and δ any value such that $0 < \delta < 1/n$. Define

$$\iota\langle P_i, S^k\rangle = \begin{cases} \iota_i\langle P_i, S^k\rangle & \text{If } S^k \subseteq face_i(S^n). \\ (1+\delta)\vec{b} - \delta\vec{s}_i & \text{If } S^k = S^n. \end{cases}$$

The *mesh* of a complex is the maximum diameter of any simplex.

Lemma 6.1 *$mesh(\chi(S^n)) \leq \frac{n}{n+1} diam(S^n)$.*

Proof: We argue by induction on n. When n is zero, the claim is trivial. Let $\dot{\mathcal{S}}^{n-1}$ be the boundary complex of S^n, $\beta(S^n)$ the barycentric subdivision, and $\vec{b}$ and $\vec{b}_i$ the respective barycenters of S^n and $face_i(S^n)$. Assume inductively that the claim holds for simplexes in $\chi(\dot{\mathcal{S}}^{n-1})$. Each remaining central vertex $\vec{x}_i$ has the form $\vec{x}_i = (1+\delta)\vec{b} - \delta\vec{s}_i$, which lies on the line joining $\vec{b}$ to $\vec{b}_i$. If $\vec{x} \in \chi(face_i(S^n))$, then the edge $(\vec{x}, \vec{x}_i)$ lies inside the triangle $(\vec{x}, \vec{b}, \vec{b}_i)$, which lies inside a simplex in $\beta(S^n)$. Since $mesh(\beta(S^n)) \leq (n/(n+1))diam(S^n)$ [38, Theorem 15.4], $|\vec{x} - \vec{x}_i| \leq (n/(n+1))diam(S^n)$. Finally, $|\vec{x}_i - \vec{x}_j| = \delta|\vec{s}_i - \vec{s}_j|$, and the claim follows because $\delta < 1/n$. ■

Lemma 6.1 implies that by taking sufficiently large k, $mesh(\chi^k(\mathcal{I}))$ can be made arbitrarily small.

```
Initially: f[i] = n+2; view_f[j] = null for j in {1..n+1};
           S = empty;

procedure participating-set(i: process id; f: shared array);
  repeat
   f[i] := f[i]-1;
   for j := 1 to n+1 do view_f[j] := f[j] od;
   S := {j | view_f[j] <= f[i]};
  until |S| >= f[i];
  return S;
end participating-set;
```

Figure 20: The Participating Set Protocol.

6.2 Simplex Agreement

Lemma 6.2 *There exists a wait-free solution to simplex agreement with input complex $\mathcal{I}$ and output complex $\chi(\mathcal{I})$, the standard chromatic subdivision.*

Proof: Each process P_i must choose a face of S_i of S^n such that (1) $P_i \in ids(S_i)$, (2) for all S_i and S_j, one is a subset of the other, and (3) if $P_j \in ids(S_i)$, then $S_j \subseteq S_i$. This is exactly the *participating set* problem of Borowsky and Gafni [11], developed as part of their "immediate snapshot" algorithm. Their elegant wait-free solution appears in Figure 20. Borowsky [9] gives a proof of this algorithm. ■

The following lemma is immediate.

```
f[1..k][0..n], S[1..k][0..n], input[0..n]: shared array;
Initially for all r in {1..k} f[r][i] = n+2;
          S[r][i] = empty;

procedure simplex-agree(i: process_id;
                        my_vertex: vertex value;
                        k: refinement);

  input[i] := my_vertex;
  for r := 1 to k do
   S[r][i] := participating-set(i,f[r]);
   if r = 1
     then vertex[j,1] := <i,{input[k] | k in S[j,1]}>
     else vertex[j,r] := <i,{vertex[k,r-1] | k in S[j,r]}>
  return(μ(φ(vertex(i,k))));
end simplex-agree;
```

Figure 21: The Iterated Participating Set Protocol.

Lemma 6.3 *The iterated version of the participating set algorithm (Figure 21) is a wait-free solution to simplex agreement with input complex $\mathcal{I}$ and output complex $\chi^k(\mathcal{I})$, the iterated standard chromatic subdivision for any $k > 0$.*

6.3 Approximating Chromatic Subdivisions

We start with some conventional definitions.

Definition 6.3 An *open cover* of a compact metric space $\mathcal{C}$ is a finite collection $U_0, \ldots, U_k$ of open sets such that $\mathcal{C} \subset \cup_{i=0}^{k} U_i$. A *Lebesgue number* for the covering is a $\lambda > 0$ such that any closed set of diameter less than or equal to λ lies entirely in one of the U_i.

Lemma 6.4 ([38, p.89]) *Every open cover of a compact metric space has a Lebesgue number.*

Definition 6.4 Given two chromatic subdivisions $\sigma(S^n)$ and $\tau(S^n)$, a simplicial map $\phi : \sigma(S^n) \rightarrow \tau(S^n)$ is *conservative* if it is color and carrier preserving.

Our main combinatorial result is to show that if $\sigma(S^n)$ is an arbitrary chromatic subdivision of S^n, then there exists a K such that for all $k \geq K$, there is a conservative simplicial map:

$$\phi : \chi^k(S^n) \to \sigma(S^n).$$

As a first step, we show that given a subdivision of a simplex, the result of "perturbing" a vertex within its carrier by a sufficiently small distance is still a subdivision.

Definition 6.5 Let $\sigma(S^n)$ be a subdivision of S^n. An *ϵ-perturbation* of $\sigma(S^n)$ is a complex $\sigma'(S^n)$ with a conservative map $\iota : \sigma(S^n) \to \sigma'(S^n)$, bijective on vertexes, such that for all $\vec{v}$, $|\vec{v} - \iota(\vec{v})| < \epsilon$.

To avoid cumbersome notation and diction, we will usually say "$\tau(S^n)$ can be perturbed ..." in place of explicit references to τ' and ι.

Any simplex S^k determines a unique k-dimensional hyperplane, denoted $hyper(S^k)$.

Lemma 6.5 *Let $\vec{s}$ and $\vec{t}_0, \ldots, \vec{t}_{n-1}$ be points in $|S^n|$. If $\vec{s}$ is affinely independent of the $\vec{t}_0, \ldots, \vec{t}_{n-1}$, then so is every point within some $\epsilon > 0$ of $\vec{s}$.*

Proof: Suppose not. If every neighborhood of $\vec{s} \in carrier(\vec{s}, S^n)$ contains a point affinely dependent on the $\vec{t}_i$, then by examining successively smaller neighborhoods, one can construct a Cauchy sequence $\vec{s}_i$, converging to $\vec{s}$, so that each $\vec{s}_i$ is affinely dependent on the $\vec{t}_i$. The $\vec{s}_i$ lie in the hyperplane $hyper(\vec{t}_0, \ldots, \vec{t}_{n-1})$. Since this hyperplane is closed, it contains the limit point $\vec{s}$, hence $\vec{s}$ itself is affinely dependent on the $\vec{t}_i$, a contradiction. ∎

Lemma 6.6 *If $\sigma(S^n)$ is a subdivision of S^n, then there exists $\epsilon > 0$, such that any ϵ-perturbation of $\sigma(S^n)$ is also a subdivision of S^n.*

Proof: Because $\sigma(S^n)$ encompasses a finite number of vertexes, it is enough to show that any vertex $\vec{s}$ of $\sigma(S^n)$ can be perturbed by $\epsilon > 0$ within its carrier. Let L_i^{n-1} be a simplex in $lk(\vec{s}, \sigma(S^n))$. Because $\vec{s} \cdot L_i^{n-1}$ is a simplex of $st(\vec{s}, \sigma(S^n))$, $\vec{s}$ is affinely independent of the vertexes of L_i^{n-1}. By Lemma 6.5, every point within some $\epsilon_i > 0$ of $\vec{s}$ is also affinely independent of L_i^{n-1}. Since the link contains only a finite number of simplexes, we can let $\epsilon = \min(\epsilon_i)$. For any $\vec{r}$ within ϵ of $\vec{s}$, $|\vec{r} \cdot lk(\vec{s}, \sigma(S^n))| = |star(\vec{s}, \sigma(S^n))|$ so the result of perturbing $\vec{s}$ within ϵ remains a subdivision of S^n. ∎

Henceforth, all perturbations are assumed to be subdivisions. Note that $mesh(\sigma'(S^n)) \leq mesh(\sigma(S^n)) + 2\epsilon$.

Definition 6.6 $\sigma(S^n)$ and $\tau(S^n)$ are *chromatically independent* if for every $\vec{t} \in \tau(S^n)$ and $S^k \in \sigma(S^n)$ such that $id(\vec{t}) \notin ids(S^k)$, $\vec{t}$ and S^k are affinely independent.

Lemma 6.7 *$\tau(S^n)$ has an ϵ-perturbation chromatically independent of any $\sigma(S^n)$.*

Proof: Given $\vec{t} \in \tau(S^n)$ and $S^k \in \sigma(S^n)$ such that $id(\vec{t}) \notin ids(S^k)$, we show that $\vec{t}$ can be perturbed to be affinely independent of S^k. Let $carrier(\vec{t}, S^n)$ have dimension c, and let C be the hyperplane determined by $carrier(\vec{t}, S^n)$. If the hyperplane H determined by S^k contains $carrier(\vec{t}, S^n)$, then $ids(carrier(\vec{t}, S^n)) \subseteq ids(S^k)$, which is impossible because $id(\vec{t}) \notin ids(S^k)$. The intersection of H and C is a hyperplane of dimension less than c, so any ϵ-neighborhood of $\vec{t}$ in $carrier(\vec{t}, S^n)$ contains points not on H. Moreover, the same property holds for any finite collection of hyperplanes. ∎

Lemma 6.8 *If $\sigma(S^n)$ and $\tau(S^n)$ are chromatically independent, then for any $S^k \in \sigma(S^n)$ and $T^\ell \in \tau(S^n)$, if $ids(S^k)$ and $ids(T^\ell)$ are disjoint, then so are $|S^k|$ and $|T^\ell|$.*

Proof: Because $ids(S^k)$ and $ids(T^\ell)$ are disjoint, $k + \ell < n$, and because $\sigma(S^n)$ and $\tau(S^n)$ are chromatically independent, $|S^k|$ and $|T^\ell|$ lie on non-intersecting hyperplanes. ∎

Lemma 6.9 *If $\sigma(S^n)$ and $\tau(S^n)$ are chromatically independent, then for any $T \in \tau(S^n)$, the set*

$$\{st^\circ(\vec{s}, \sigma(S^n)) | id(\vec{s}) \in ids(T)\}$$

is an open cover of $|T|$.

Proof: Any point not in the open cover lies in $|S|, S \in \sigma(S^n)$, such that $ids(S)$ and $ids(T)$ are disjoint. By Lemma 6.8, $|S|$ and $|T|$ are also disjoint. ∎

Definition 6.7 Consider complexes $\mathcal{B} \subset \mathcal{C}$, and let $\sigma(\mathcal{B})$ be a subdivision. The *relative subdivision* $\sigma(\mathcal{B}, \mathcal{C})$ is defined as follows. For all $B \in \sigma(\mathcal{B})$, if $carrier(B, \mathcal{B}) \cdot C \in \mathcal{C}$, then $B \cdot C \in \sigma(\mathcal{B}, \mathcal{C})$.

Definition 6.8 Define the *extended star* of a simplex T in $\mathcal{C}$ to be

$$st^*(T, \mathcal{C}) = \bigcup_{\vec{t} \in T} st(\vec{t}, \mathcal{C}).$$

The proof of the next lemma is left as an exercise for the reader.

Lemma 6.10 *If σ is a subdivision of S^n, then $diam(st^*(T^n, \sigma(S^n))) \leq 3 \cdot mesh(\sigma(S^n))$.*

Definition 6.9 Let $\mathcal{T}$ be a complex whose polyhedron is contained in $|S^n|$. A vertex $\vec{t} \in \mathcal{T}$ is *covered* in $\mathcal{T}$ by $\vec{s}$ if there exists $\vec{s} \in \sigma(S^n)$ such that $carrier(\vec{t}, S^n) = carrier(\vec{s}, S^n)$, $id(\vec{s}) = id(\vec{t})$, and $st(\vec{t}, \mathcal{T}) \subset st^\circ(\vec{s}, \sigma(S^n))$.

We now inductively construct a sequence of complexes $\mathcal{T}_n, \ldots, \mathcal{T}_0$ (note the decreasing index) with matching subdivisions $\tau_n, \ldots, \tau_0$, such that

1. $\mathcal{T}_n = S^n$.
2. $\mathcal{T}_k$ has dimension at most k.
3. $\mathcal{T}_k \subset \tau_{k+1}(\mathcal{T}_{k+1})$.
4. Every simplex $T^k \in \mathcal{T}_k$ has at least one vertex covered in $\mathcal{T}_k$.

By Lemma 6.9, the open stars of the vertexes of $\sigma(S^n)$ form an open cover of $|S^n|$ with Lebesgue number λ_n (Lemma 6.4), By Lemma 6.1, we can pick K_n large enough that $mesh(\chi^{K_n}(S^n)) < \lambda_n/9$. By Lemma 6.7, $\chi^{K_n}(S^n)$ has an ϵ-perturbation τ_n chromatically independent of $\sigma(S^n)$. Perturbation adds at most $2\lambda_n/9$ to the diameter of any simplex in the subdivision, so $mesh(\tau_n(S^n)) \leq \lambda_n/3$. By Lemma 6.10, every $T^n \in \tau_n(S^n)$,

$$diam(st^*(T^n, \tau_n(S^n))) \leq 3 \cdot mesh(\tau_n(S^n)) < \lambda_n,$$

so $st^*(T^n, \tau_n(S^n)) \subset st^\circ(\vec{s}, \sigma(S^n))$ for some $\vec{s} \in \sigma(S^n)$. Since $ids(T^n)$ includes all $n+1$ processes, T^n has at least one vertex $\vec{t}$ covered in $\tau_n(S^n)$ by some $\vec{s} \in \sigma(S^n)$. For all such $\vec{t} \in T^n$ and $\vec{s} \in \tau_n(S^n)$ define $\phi(\vec{t}) = \vec{s}$. Let $\mathcal{T}_{n-1}$ be the largest subcomplex of $\tau_n(S^n)$ containing all uncovered vertexes. As noted, $\mathcal{T}_{n-1}$ has dimension at most $n-1$.

Inductively assume we have constructed τ_{k+1} and $\mathcal{T}_k$. Because τ_{k+1} is chromatically independent of σ, Lemma 6.9 implies that

$$\left\{ st^\circ(\vec{s}, \sigma(S^n)) | carrier(\vec{s}, S^n) = carrier(\vec{t}, S^n) \wedge id(\vec{s}) \in ids(T^k) \right\}$$

is an open cover of each T^k. Because $\mathcal{T}_k$ is finite, we can take λ_k to be the minimum of the Lebesgue numbers for all such coverings.

By Lemma 6.1, we can pick K_k large enough that $mesh(\chi^{K_k}(\mathcal{T}_k)) < \lambda_k/9$. By Lemma 6.7, $\chi^{K_k}(\mathcal{T}_k)$ has an ϵ-perturbation $\tau_k(S^n)$ chromatically independent of $\sigma(S^n)$. Perturbation adds at most $2\lambda_k/9$ to the diameter of any simplex, so $mesh(\tau_k(\mathcal{T}_k)) \leq \lambda_k/3$. For each $T^k \in \tau_k(\mathcal{T}_k)$,

$$diam(st^*(T^k, \tau_k(\mathcal{T}_k))) \leq 3 \cdot mesh(\tau_k(\mathcal{T}_k)) < \lambda_k,$$

so $st^*(T^k, \tau_k(\mathcal{T}_k)) \subset st^\circ(\vec{s}, \sigma(S^n))$, for some $\vec{s}$ such that $id(\vec{s}\,) \in ids(T^k)$. Every T^k thus contains at least one vertex $\vec{t}$ covered in $\tau_k(\mathcal{T}_k)$ by some $\vec{s} \in \sigma(S^n)$. For all such $\vec{t}$ and $\vec{s}$, define $\phi(\vec{t}\,) = \vec{s}$, and let $\mathcal{T}_{k-1}$ be the largest subcomplex of $\tau_k(\mathcal{T}_k)$ containing all uncovered vertexes.

Inductively define

$$\tau_k(S^n) = \tau_k(\mathcal{T}_i, \tau_{k+1}(S^n)),$$

and let $\tau(S^n) = \tau_0(S^n)$.

Lemma 6.11 ([40, 2.1.25]) *A set of vertexes $\vec{v}_0, \ldots, \vec{v}_m$ belong to a common m-simplex of $\mathcal{C}$ if and only if*

$$\bigcap_{i=0}^{m} st^\circ(\vec{v}_i, \mathcal{C}) \neq \emptyset.$$

Lemma 6.12 *The vertex-to-vertex map $\phi : \tau(S^n) \to \sigma(S^n)$ is a conservative simplicial map.*

Proof: It follows directly from the construction that ϕ is color and carrier-preserving. It remains to check that ϕ is simplicial. For any $T^n = (\vec{t}_0, \ldots, \vec{t}_n)$ in $\tau(S^n)$, the construction ensures that one can assign each $\vec{t}_i \in T^n$ a *rank*: the largest k such that $\vec{t}$ is covered in $\tau_k(\mathcal{T}_k)$. If $\vec{t}_i$ has rank less than or equal to $\vec{t}_j$, then $\vec{t}_i \in st^\circ(\phi(\vec{t}_j), \tau(S^n))$. Any vertex of minimal rank is in the intersection of all $st^\circ(\phi(\vec{t}_i), \tau(S^n))$, so ϕ is a simplicial map by Lemma 6.11. ■

To show that $\tau(S^n)$ is the image of an iterated standard chromatic subdivision, we need the following technical lemma.

Lemma 6.13 *Given colored complexes $\mathcal{A}, \mathcal{B}$, and $\mathcal{C}$, $\mathcal{B} \subset \mathcal{C}$, and a conservative map $\phi : \mathcal{A} \to \mathcal{C}$, there exists a conservative map $\psi : \chi^\ell(\mathcal{A}) \to \chi^\ell(\mathcal{B}, \mathcal{C})$, that agrees with ϕ on vertexes of $\mathcal{A}$.*

Proof: We claim that there exists a conservative map $\eta_1 : \chi(\mathcal{A}) \rightarrow \chi(\mathcal{C})$ that agrees with ϕ on vertexes of $\mathcal{A}$. If $\vec{v}$ is a central vertex of $\chi(A^n)$, define $\eta_1(\vec{v})$ to be the central vertex of $\chi(\phi(A^n))$ labeled with the same color. Iterating this construction yields a conservative map $\eta_\ell : \chi^\ell(\mathcal{A}) \rightarrow \chi^\ell(\mathcal{C})$.

We next define a conservative map $\theta : \chi^\ell(\mathcal{C}) \rightarrow \chi^\ell(\mathcal{B}, \mathcal{C})$ by

$$\theta(\vec{v}) = \begin{cases} \vec{v} & \text{if } \vec{v} \in \mathcal{B} \\ \vec{u} \in carrier(\vec{v}, \mathcal{C}) \text{ such that } id(\vec{u}) = id(\vec{v}) & \text{otherwise.} \end{cases}$$

The map ψ is the composition of η_ℓ and θ. ■

Lemma 6.14 *There is a conservative simplicial map*

$$\phi : \chi^{\sum_{i=0}^{n} K_i}(S^n) \rightarrow \tau(S^n).$$

Proof: We inductively build up a family

$$\phi_k : \chi^{\sum_{i=k}^{n} K_i}(S^n) \rightarrow \tau(S^n).$$

For the base case, $\psi_n : \chi^{K_n}(S^n) \rightarrow \tau_n(S^n)$ is the identity map on $\chi^{K_n}(S^n)$, induced by the perturbation of $\chi^{K_n}(S^n)$ used to define τ_n.

For the induction step, Lemma 6.13 states that there exists a conservative map $\theta : \chi^{\sum_{i=k}^{n} K_i}(S^n) \rightarrow \chi^{K_k}(\tau_{k+1}(S^n))$. The map ψ_k is the perturbation of θ induced by the perturbation used to define τ_k. ■

Theorem 6.15 *A decision task $\langle \mathcal{I}, \mathcal{O}, \Delta \rangle$ has a wait-free protocol using read-write memory if there exists a chromatic subdivision $\sigma(\mathcal{I})$ and a chromatic simplicial map*

$$\mu : \sigma(\mathcal{I}) \rightarrow \mathcal{O}$$

such that for each simplex S^m in $\sigma(\mathcal{I})$, $\mu(S) \in \Delta(carrier(S))$.

Proof: Lemma 6.14 implies that for each $S_i^n \in \mathcal{I}$, we can find a K_i such that for all $k > K_i$, there exists a conservative map $\phi : \chi^k(S^n) \rightarrow \sigma(S^n)$. Let $K = \max_i(K_i)$, which is well-defined because $\mathcal{I}$ is finite. On input simplex S^n, the protocol has the following steps:

1. Using the iterated participating set protocol, the processes agree on a simplex in $\chi^K(S^n)$.

2. A process that chooses vertex $\vec{x} \in \chi^K(S^n)$ then chooses as its output $val(\mu(\phi(\vec{x})))$.

■

This completes the proof of the Asynchronous Computability Theorem.

7 Renaming with a Small Number of Names

In this section we use the Asynchronous Computability Theorem to prove the impossibility of solving the renaming task with a small number of names.

7.1 Comparison Protocols

A protocol is *comparison-based* if the only operations a process can perform on process ids is to test for equality and order; that is, given two process ids P and Q, a process can test for $P = Q$, $P \leq Q$, and $P \geq Q$, but cannot examine the structure of the identifiers in any more detail (e.g., it cannot test whether P is prime).

Let $\mathcal{A}$ and $\mathcal{B}$ be complexes where each vertex is labeled with a process id, and possibly with a value. $\mathcal{B}$ is a *recoloring* of $\mathcal{A}$ if there exists a bijective simplicial map (*not* color-preserving)

$$\rho : \mathcal{A} \to \mathcal{B}$$

that is (1) order-preserving on process ids: if $id(\vec{u}) < id(\vec{v})$ then $id(\rho(\vec{u})) < id(\rho(\vec{v}))$, and (2) value-preserving: if $val(\vec{v})$ is defined then $val(\vec{v}) = val(\rho(\vec{v}))$.

Theorem 7.1 (ACT for Comparison-Based Protocols) *A decision task $\langle \mathcal{I}^n, \mathcal{O}^n, \Delta \rangle$ has a wait-free solution using read-write memory if and only if there exists a chromatic subdivision $\sigma(\mathcal{I}^n)$ and a chromatic simplicial map*

$$\mu : \sigma(\mathcal{I}^n) \to \mathcal{O}^n$$

such that (1) for each simplex S^m in $\sigma(\mathcal{I}^n)$, $\mu(S^m) \in \Delta(carrier(S^m))$, (2) any recoloring $\rho : S_0^m \to S_1^m$ induces a recoloring $\rho' : \mu(\sigma(S_0^m)) \to \mu(\sigma(S_1^m))$ such that for every face S^ℓ of S_0^m, $\rho'(\mu(\sigma(S^\ell))) = \mu(\sigma(\rho(S^\ell)))$.

The second condition captures the notion that the behavior of comparison-based protocols does not change if processes are renamed in an order-preserving way. The proof of this theorem is almost identical to the proof of Theorem 4.1, except that it is necessary to check at each step of the span construction that the equivalence-under-recoloring property continues to hold.

7.2 The Renaming Problem

In the renaming task of Attiya *et al.* [6], processes are given unique input names from a large name space, and must choose unique output names taken from a smaller name space. More precisely, we are given a set of $N^* > 2n+1$ processes. In any execution, however, at most $n+1$ of them participate. (Formally, the relation Δ is defined only on simplexes of dimension at most n.) These *participating* processes are required to choose unique names in the range $0, \ldots, K$. In the message-passing model, Attiya et al. [6] showed that renaming has a wait-free solution for when $K \geq 2n+1$, and none when $K \leq n+2$. Bar-Noy and Dolev [7] extended their upper bound solution to the shared read-write memory model. The protocols of Attiya et al. and of Bar-Noy and Dolev are both comparison-based. Whether a protocol exists for $n+2 < K \leq 2n$ names remained open until 1993, when Herlihy and Shavit [30] showed that no such protocol exists. Henceforth, by *renaming*, we mean the renaming task where $K \leq 2n$.

Lemma 7.2 *If a renaming protocol exists, then a comparison-based protocol exists.*

Proof: Assume we are given a wait-free renaming protocol Π. We use the comparison-based renaming protocol of Bar-Noy and Dolev as a *filter* protocol Φ. The participating processes first execute Φ to choose an *intermediate name*, and then they run Π, where each process uses its intermediate name as its process id. Note that intermediate names are unique, and lie in the range $0, \ldots, 2n$. The composite protocol is clearly comparison-based, because the original process ids are used only the comparison-based Φ, not in Π. ∎

We use the converse of this lemma: if no wait-free comparison-based protocol exists, then no wait-free protocol exists. We exploit the additional symmetry properties of Theorem 7.1 to show that no comparison-based protocol exists.

It is also convenient to simplify the task by reducing the size of the output complex. Consider the following *reduced renaming* task.

Reduced Renaming At most $n+1$ out of N^* processes participate in each execution. Each participating process chooses a binary value, and in every execution where all $n+1$ processes choose a value, at least one chooses 0, and at least one chooses 1.

Lemma 7.3 *If a comparison-based renaming protocol exists, then so does a comparison-based reduced renaming protocol.*

Proof: Any comparison-based renaming protocol can be transformed into a comparison-based reduced renaming protocol simply by taking the parity of the names chosen. ∎

We now demonstrate the impossibility of comparison-based reduced renaming.

Here is an informal summary of our proof strategy for three processes. Assume we have a comparison-based protocol for three-process reduced renaming. Let S^2 be an input simplex labeled with process ids P, Q, and R. The matching output complex $\mathcal{O}^2$ appears in Figure 22. This complex is constructed by taking a binary 2-sphere over P, Q, and R, and removing the all-zero and all-one simplexes. The result can be viewed as a cylinder or an annulus: a ring with a hole in the middle. We will show that the simplicial map μ "wraps" the boundary of $\sigma(S^2)$ around the hole in the output complex a non-zero number of times. This observation yields a contradiction, because a simplicial map is carrying the boundary of a "solid" region to the boundary of a "hole".

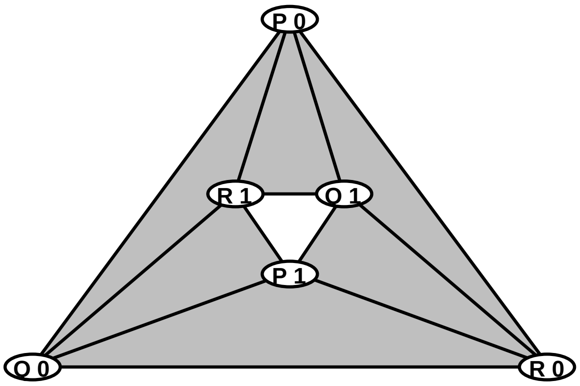

Figure 22: Reduced Renaming Output Complex $\mathcal{O}^2$

7.3 Orientation, Cycles, and Boundaries

This section presents some additional definitions needed for our proof. Our discussion closely follows that of Munkres [38, Section 1.13], which the reader is encouraged to consult for more details.

Let $S^n = (\vec{s}_0, \ldots, \vec{s}_n)$ be an n-simplex. An *orientation* for S^n is an equivalence class of orderings on $\vec{s}_0, \ldots, \vec{s}_n$, consisting of one particular ordering and all even permutations of it. For example, an orientation of a 1-simplex $(\vec{u}, \vec{v})$ is just a direction, either from $\vec{u}$ to $\vec{v}$, or vice-versa. An orientation of a 2-simplex $(\vec{u}, \vec{v}, \vec{w})$ can either be clockwise, as in $(\vec{u}, \vec{v}, \vec{w})$, or counterclockwise $(\vec{u}, \vec{w}, \vec{v})$. Any orientation of a simplex induces orientations on its faces. An n-manifold with boundary is *oriented* if each n-simplex is oriented in a way that each internal $(n-1)$-simplex inherits opposite orientations from its two containing n-simplexes. Any n-sphere can be oriented in this way. Given a simplex whose vertexes are colored by process ids, the *standard orientation* orders them by increasing process id.

A *d-chain* is a formal sum of oriented d-simplexes: $\sum_{i=0}^{\ell} \lambda_i \cdot S_i^d$, where λ_i is an integer. When writing chains, we typically omit d-simplexes with zero coefficients, unless they are all zero, when we simply write 0. We write $1 \cdot S^d$ as S^d, $-1 \cdot S^d$ as $-S^d$, and $S^d + \ldots + S^d$ (k times) as $k \cdot S^d$. It is convenient to identify $-S^d$ with S^d 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 $\mathcal{K}$.

Let $S^d = (v_0, \ldots v_d)$ be an oriented d-simplex. Define $face_i(S^d)$, the i^{th} *face* of S^d, to be the $(d-1)$-simplex $(v_0, \ldots, \hat{v}_i, \ldots, v_d)$, where the circumflex denotes omission. The *boundary* operator $\partial_q : C_q(\mathcal{K}) \to C_{q-1}(\mathcal{K})$, $q > 0$, is defined on simplexes:

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

and extends additively to chains: $\partial_q(\alpha_0 + \alpha_1) = \partial_q \alpha_0 + \partial_q \alpha_1$. The boundary operator has the important property that applying it twice causes chains to vanish:

$$\partial_{q-1} \partial_q \alpha = 0. \tag{8}$$

(Henceforth, we sometimes omit subscripts from boundary operators.) A q-chain α is a *boundary* if $\alpha = \partial \beta$ for some $(q+1)$-chain β, and it is a *cycle* if $\partial \alpha = 0$. Equation 8 implies that that every boundary is a cycle. Two d-cycles α and β are *homologous*, written $\alpha \sim \beta$, if $\alpha - \beta$ is a boundary.

The *chain complex* $C(\mathcal{K})$ is the sequence of groups and homomorphisms $\{C_q(\mathcal{K}), \partial\}$. Let $C(\mathcal{K}) = \{C_q(\mathcal{K}), \partial\}$ and $C(\mathcal{L}) = \{C_q(\mathcal{L}), \partial'\}$ 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 preserve cycles and boundaries: $\partial' \circ \phi_q = \phi_{q-1} \circ \partial$. Subdivision and simplicial maps induce chain maps in the obvious way.

We use the following facts ([38, Th.8.3]) about chain groups of spheres. Let the complex $\mathcal{B}^{n-1}$ be an $(n-1)$-sphere, and let B be the cycle constructed by orienting the simplexes of $\mathcal{B}^{n-1}$.

Lemma 7.4 *For $q < n-1$, every q-cycle of $\mathcal{B}^{n-1}$ is a boundary, and every $(n-1)$-cycle is homologous to $k \cdot B$, for some integer k.*

We refer to B as a *generator* for $\mathcal{B}^{n-1}$.

Let $\mathcal{A} \subset \mathcal{K}$ be complexes. A *deformation retraction* of $\mathcal{K}$ onto $\mathcal{A}$ is a continuous map $F : |\mathcal{K}| \times I \to |\mathcal{K}|$ such that $F(x, 0) = x$ for $x \in |\mathcal{K}|$, $F(x, 1) \in \mathcal{A}$ for $x \in |\mathcal{K}|$, and $F(a, t) = a$ for $a \in |\mathcal{A}|$. If such an F exists, we say that $\mathcal{A}$ is a *deformation retract* of $\mathcal{K}$. Every cycle of $\mathcal{K}$ is homologous to a cycle of $\mathcal{A}$.

Lemma 7.5 *If the $(n-1)$-sphere $\mathcal{B}$ is a deformation retract of $\mathcal{K}$, then every q-cycle of $\mathcal{K}$ is a boundary, and every $(n-1)$-cycle of $\mathcal{K}$ is homologous to $k \cdot B$, for some integer k.*

We also refer to B as a *generator* of $\mathcal{K}$. If B and B' are both generators of $\mathcal{K}$, then $B \sim \pm B'$.

This paragraph is an aside for readers already somewhat familiar with standard algebraic topology. The simplicial homology groups $H_q(\mathcal{K})$ for a complex $\mathcal{K}$ are typically defined to be the quotient groups of the q-dimensional group of cycles by the q-dimensional group of boundaries. Theorem 7.4 reformulates the well-known result that the homology groups of the $(n-1)$-sphere $\mathcal{B}^{n-1}$ are trivial below $(n-1)$, and infinite cyclic in dimension n, generated by B. Lemma 7.5 is a special case of the well-known classical theorem [38, P.108] that any complex has the same homology as its deformation retracts.

7.4 The Impossibility of Renaming

Let S^n be an input simplex. The subdivision σ induces a chain map

$$s : C(S^n) \to C(\sigma(S^n)),$$

and the simplicial map μ induces a chain map

$$m : C(\sigma(S^n)) \to C(\mathcal{O}^n).$$

Let the chain map $a : C(S^n) \to C(\mathcal{O})$ denote the composition of s and m.

Let $\rho_i : \mathit{face}_n(S^n) \to \mathit{face}_i(S^n)$ be a recoloring, and $\rho''_i : \mu(\sigma(\mathit{face}_n(S^n))) \to \mu(\sigma(\mathit{face}_i(S^n)))$ the matching recoloring guaranteed by Theorem 7.1. These recolorings induce chain maps

$$r_i : \; C(\mathit{face}_n(S^n)) \;\to\; C(\mathit{face}_i(S^n))$$

and

$$r'_i : C(\mu(\sigma(\mathit{face}_n(S^n)))) \;\to\; C(\mu(\sigma(\mathit{face}_i(S^n))))$$

that commute for all chains C,

$$r_i(a(C)) = a(r_i'(C)). \tag{9}$$

It is also convenient to define the chain map $z : C(S^n) \to C(\mathcal{O}^n)$ induced by the vertex map that sends each $\vec{s}_i$ to $\langle P_i, 0\rangle$. This map also commutes with r and r_i':

$$r_i(z(\alpha)) = z(r_i'(\alpha)). \tag{10}$$

We now consider the effects of recoloring on the output complex homology. Let $0^n \in \mathcal{B}^n$ be the all-zero n-simplex with standard orientation, and $\dot{0}^{n-1}$ its boundary complex. Because the $(n-1)$-sphere $\dot{0}^{n-1}$ is a deformation retract of $\mathcal{O}^n$, $\partial 0^n$ is a generator for $\mathcal{O}^n$. Note that $\partial 0^n = z(\partial S^n)$.

Let $\mathcal{O}_i^{n-1}$ denote the subcomplex of $\mathcal{O}^n$ colored with process ids $P_0, \ldots, \widehat{P}_i, \ldots, P_n$. $\mathcal{O}_i^{n-1}$ is a binary $(n-1)$-sphere. Let B_i denote the chain constructed by orienting each $(n-1)$-simplex of $\mathcal{O}_i^{n-1}$ so that the all-zero simplex has standard orientation (Figure 23). Because each $\mathcal{O}_i^{n-1}$ is a deformation retract of $\mathcal{O}^n$, each B_i is a generator for $\mathcal{O}^n$. Note that $r_i'(B_n) = B_i$. Algebraically, the generators B_i are related as follows:

Lemma 7.6 $B_i \sim (-1)^i \partial 0^n$.

Proof: Since B_i and $\partial 0^n$ are both generators for $\mathcal{O}^n$, $B_i \sim \pm\partial 0^n$. The simplex $face_i(0^n)$ is common to both B_i and $\partial 0^n$. It has the standard orientation in B_i, and $(-1)^i$ times the standard orientation in $\partial 0^n$. ∎

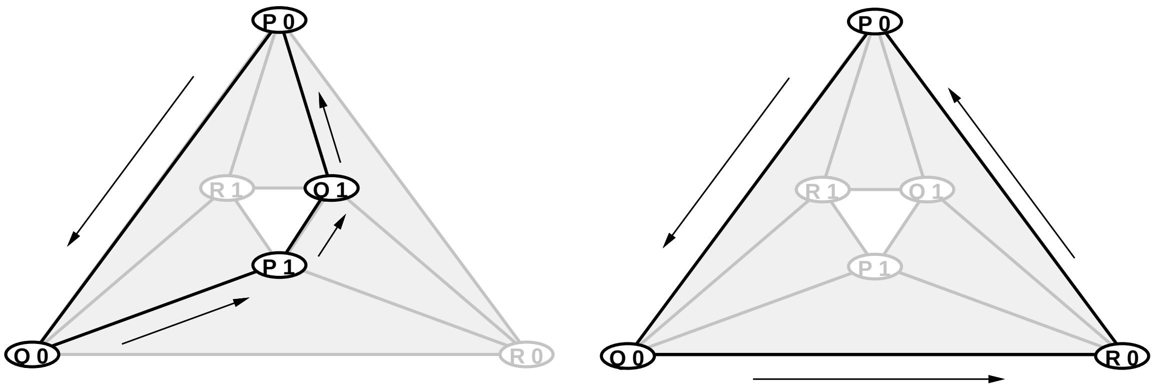

Figure 23: Cycles B_2 and $\partial 0^2$

Lemma 7.7 *For $0 \leq i < n$, there is a chain map*

$$d : C_i(S^n) \to C_{i+1}(\mathcal{O}^n)$$

such that

$$a(S^m) - z(S^m) - d(\partial S^m)$$

is an m-cycle of $\mathcal{O}^n$. Moreover, if $r_i' : C(S^m_{m+1}) \to C(S^m_i)$ is a recoloring,

$$r_i'(d(S^{m-1})) = d(r_i(S^{m-1}))$$

for $0 \leq m < n$.

Proof: We argue by induction on m. When $m = 1$, $ids(S^1) = \{i, j\}$. $a(S^1)$ and $z(S^1)$ are 1-chains with a common boundary $\langle P_i, 0\rangle - \langle P_j, 0\rangle$, so $a(S^1) - z(S^1)$ is a cycle, and $d(\langle P_i, 0\rangle) = 0$.

Assume the claim for m, $1 \leq m < n - 1$. By Lemma 7.5, every m-cycle is a boundary (for $m < n - 1$), so there exists an $(m+1)$-chain $d(S^m)$ such that

$$\partial d(S^m) = a(S^m) - z(S^m) - d(\partial S^m)$$

Assume inductively that $r_i'(d(S^{m-1}_m)) = d(r_i(S^{m-1}))$.

$$\begin{aligned}
\partial r_i'(d(S^m_{m+1})) &= r_i'\left(a(S^m_{m+1}) - z(S^m_{m+1}) - d(\partial S^m_{m+1})\right) \\
&= a(r_i(S^m_{m+1})) - z(r_i(S^m_{m+1})) - d(\partial(r_i(S^m_{m+1}))) \\
&= \partial d(r_i(S^m_{m+1}))
\end{aligned}$$

Taking the alternating sum over the faces of S^{m+1} yields

$$\begin{aligned}
\partial d(\partial S^{m+1}) &= a(\partial S^{m+1}) - z(\partial S^{m+1}) - d(\partial\partial S^{m+1}) \\
&= a(\partial S^{m+1}) - z(\partial S^{m+1}).
\end{aligned}$$

Rearranging terms yields

$$0 = \partial\left(a(S^{m+1}) - z(S^{m+1}) - d(\partial S^{m+1})\right),$$

implying that

$$C = a(S^{m+1}) - z(S^{m+1}) - d(\partial S^{m+1})$$

is an $(m+1)$-cycle.

■

Theorem 7.8 *There is no wait-free reduced renaming protocol.*

Proof: By Lemma 7.7,

$$a(S_n^{n-1}) - z(S_n^{n-1}) - d(\partial(S_n^{n-1}))$$

is an $(n-1)$-cycle of $\mathcal{O}^n$. Lemma 7.5 implies that this cycle is homologous to $k \cdot \partial B_n$, for some integer k. Recall that $r_i''(B_n) = B_i$.

$$r_i''\left(a(S_n^{n-1}) - z(S_n^{n-1}) - d(\partial(S_n^{n-1}))\right) = k \cdot r'(B_n) = k \cdot B_i.$$

Recall that $r_i(S_n^{n-1}) = \mathit{face}_i(S_n)$. Taking the alternating sum over the $(n-1)$-dimensional faces of S^n yields:

$$a(\partial S^n) - z(\partial S^n) - d(\partial\partial S^n) = \sum_{i-0}^{n}(-1)^i B_i$$

Because $\partial\partial S^n = 0$, $z(\partial S^n) = 0^n$, and $B_i \sim (-1)^i \partial 0^n$, (Lemma 7.6),

$$a(\partial S^n) \sim (1 + (n+1)k) \cdot \partial 0^n.$$

Since there is no value of k for which $(1+(n+1)k)$ is zero, the cycle $a(\partial S^n)$ is not a boundary, yielding the desired contradiction. ∎

Corollary 7.9 *There is no wait-free read/write renaming protocol with* $2n$ *or fewer names.*

8 Discussion

Since the preliminary version of this paper was published, our simplicial model has yielded a variety of additional results. The analysis of the topological properties of protocol complexes for protocols using more powerful primitives appears in a recent paper by Herlihy and Rajsbaum [29], and first tight topology-based lower bounds on certain tasks in the synchronous fail-stop model were presented by Chaudhuri, Herlihy, Lynch, and Tuttle [14]. Attiya and Rajsbaum cast our topological model in a related "combinatorial" representation [5]. Borowsky and Gafni [9, 11] have based a key part of the proof of their simulation method our proof and construction of spans. More recently, Herlihy and Rajsbaum used homology theory to derive further impossibility results for set agreement [29], and to unify a variety of known impossibility results in terms of the algebraic theory of chain maps and chain complexes [26].

The graph theoretic characterization of [8] does more than provide impossibility results, it also provides an effective procedure for deciding if a task has a 1-resilient message passing protocol. It is natural to ask whether a similar decision procedure exists for our topological representation of wait-free read/write protocols. In a recent announcement [20], Gafni and Koutsoupias assert that the classical contractibility problem of algebraic topology can be used to show that it is undecidable whether 3-process tasks have wait-free read-write protocols. Based on this idea, Herlihy and Rajsbaum [28] prove the stronger claim that for N processes it is in fact undecidable whether a task has a t-resilient protocol for all $1 < t < N$.

There are many directions in which the topological approach could be extended. Our current characterizations apply only to the class of decision tasks, short-lived synchronization problems where each process chooses a single output and halts. Most distributed and concurrent programs, however, make use of long-lived data objects, which are repeatedly accessed by processes. Extending our characterizations to such ongoing computations is an interesting research direction.

Our topological approach can give insight into complexity as well as computability. The unwritten thesis underlying our main computability theorem is that solvable decision problems are those for which one can run an "approximate agreement" algorithm (in the style of [4]) on the underlying subdivided simplicial complex to converge to a single decision simplex. In the example shown in Figure 14, one level of subdivision suffices to allow a simplicial map from input to output. In the corresponding algorithm, one process had to perform two reads to allow convergence to a final output. The example suggests that there is an inherent relation between the level of subdivision necessary to allow a simplicial map, and the complexity of the matching "approximate agreement" style algorithm (e.g. [4, 18]) defined by it.

We believe the topological approach has a great deal of promise for the theory of distributed and concurrent computation, and that it merits further investigation.

Acknowledgments

We wish to thank the many people whose comments have helped to improve this paper over the past four years: Yehuda Afek, Hagit Attiya, Elizabeth Borowsky, Alan Fekete, Eli Gafni, Nancy Lynch and her students, Shlomo Moran, Lyle Ramshaw, Mark Tuttle, and most of all to Sergio Rajsbaum.

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] Y. Afek and G. Stupp. Synchronization power depends on the register size (preliminary version). In *IEEE Symposium on Foundations of Computer Science (FOCS)*, 1993.

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

[4] 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.

[5] H. Attiya and S. Rajsbaum. A combinatorial topology framework for wait-free computability. Preprint, 1995.

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

[7] A. Bar-Noy and D. Dolev. Shared memory vs. message passing in an asynchronous distributed environment. In *Proceedings of the 8th Annual ACM Symposium on Principles of Distributed Computing*, pages 371–382, August 1989.

[8] 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.

[9] E. Borowski. Capturing the power of reseiliency and set consensus in distributed systems. Technical report, University of California Los Angeles, Los Angeles, California, 1995.

[10] 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.

[11] E. Borowsky and E. Gafni. Immediate atomic snapshots and fast renaming. In *Proceedings of the 12th Annual ACM Symposium on Principles of Distributed Computing*, August 1993.

[12] E. Borowsky and E. Gafni. The implications of the borowski-gafni simulation on the set consensus hierarchy. Technical Report TR-930021, University of California Los Angeles, Los Angeles, California, 1993.

[13] 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.

[14] 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.

[15] B. Chor, A. Israeli, and M. Li. On processor coordination using asynchronous hardware. In *Proceedings of the 6th Annual ACM Symposium on Principles of Distributed Computing*, pages 86–97, 1987.

[16] B. Chor and L. Moscovici. Solvability in asynchronous environments. In *IEEE Symposium on Foundations of Computer Science (FOCS)*, 1989.

[17] D. Dolev, C. Dwork, and L Stockmeyer. On the minimal synchronism needed for distributed consensus. *Journal of the ACM*, 34(1):77–97, January 1987.

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

[19] 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.

[20] E. Gafni and E. Koutsoupias. 3-processor tasks are undecidable. PODC 95, Brief announcement.

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

[22] L.C. Glaser. *Geometrical Combinatorial Topology*, volume 1. Van Nostra D Reinhold, New York, 1970.

[23] A. Gottlieb, R. Grishman, C.P. Kruskal, K.P. McAuliffe, L. Rudolph, and M. Snir. The NYU Ultracomputer – designing an MIMD parallel computer. *IEEE Transactions on Computers*, C-32(2):175–189, February 1984.

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

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

[26] M. Herlihy and S. Rajsbaum. Algebraic spans. In *Proceedings of the 14th Annual ACM Symposium on Principles of Distributed Computing*, pages 90–99. ACM, August 1995.

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

[28] M.P. Herlihy and S. Rajsbaum. On the decidability of distributed decision problems. In preparation.

[29] M.P. Herlihy and S. Rajsbaum. Set consensus using arbitrary objects. In *Proceedings of the 13th Annual ACM Symposium on Principles of Distributed Computing*, August 1994.

[30] 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.

[31] 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.

[32] M.P. Herlihy and J.M. Wing. Linearizability: A correctness condition for concurrent objects. *ACM Transactions On Programming Languages and Systems*, 12(3):463–492, July 1990.

[33] S. Lefschetz. *Introduction To Topology.* Princeton University Press, Princeton, New Jersey, 1949.

[34] H. R. Lewis and C. H. Papadimitriou. *Elements of the Theory of Computation*, chapter 4, pages 168–175. Prentice Hall, 1981.

[35] M.C. Loui and H.H. Abu-Amara. *Memory Requirements For Agreement Among Unreliable Asynchronous Processes*, volume 4, pages 163–183. JAI Press, 1987.

[36] N. A. Lynch. *Distributed Algorithms.* Morgan Kaufmann, New York, 1996.

[37] 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.

[38] J.R. Munkres. *Elements Of Algebraic Topology.* Addison Wesley, Reading MA, 1984. ISBN 0-201-04586-9.

[39] 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.

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