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

December 19, 1999

*Supported by NSF grant DMS-9505949.
†Supported by NSF grant CCR-9520298 and Israeli Academy of Sciences Grant Number 0361-88.

Abstract

We give necessary and sufficient combinatorial conditions characterizing the class of decision tasks that can be solved in a wait-free manner by asynchronous processes 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. This allows us to exploit powerful theorems from the classic literature on algebraic and combinatorial topology. The 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 [28, 29].

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. [7] 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, that is, cannot be continuously mapped each to the other. 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 [35].

A first step toward a systematic characterization of asynchronous computability was taken in 1988 by Biran, Moran, and Zaks [9], who 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* [7] 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 [11], Saks and Zaharoglou [38], and the current authors [28], independently derived lower bounds for the *k-set agreement* problem of Chaudhuri. The proof of Borowsky and Gafni [11] 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 [38] and the current authors [28] apply notions and techniques from mainstream combinatorial topology. Saks and Zaharoglou construct an elegant model that casts processors' collective knowledge of the unfolding computation as a topological space, and apply a variant of the Brouwer fixed point theorem [37] to derive impossibility of wait-free set agreement. This technique appears to be specific to set agreement.

In contrast, our work [28, 29] focused on general properties of the model of computation rather than on properties of specific tasks. We introduced a new formalism based on simplicial complexes and homology, notions taken from undergraduate-level algebraic topology. The new formalism replaces the popular "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 simplicial complexes. 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 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 5 and 4 respectively provide the proofs of the "if" and "only if" parts of our theorem. We conclude the paper with a proof of the impossibility of solving the *renaming* problem with a small number of names.

2 Model

We begin with an informal synopsis of our model, in which $N = n + 1$ sequential threads of control, called processes, cooperate to solve a decision task. In a decision task, each process starts with a private input value, 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 as their common output [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.

This paper considers 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 in the sense that any one can be implemented in a wait-free manner from any other. From the simplest single-bit, single-reader, and single-writer variables, one can construct multi-bit, multi-reader variables (see Lynch's survey [35]). From these variables, in turn, one can implement an *atomic snapshot memory*: an array where each P_i updates (writes) array element i, and any process can instantaneously scan (atomically read) the entire array [1, 3]. It is thus convenient to assume that processes communicate via atomic snapshot memory.

In the remainder of this section, we restate the model in more formal terms, and introduce the mathematical concepts underlying our main theorem and its proof.

2.1 Decision Tasks

We now define decision tasks more precisely. We start with the notion of a vector, which describes the input/output behavior of finite executions. Let D_I and D_O be data types, possibly the same, called the *input* and *output* data types.

Definition 2.1 *An* $(n+1)$*-process* input vector $\vec{I}$ *(respectively* output vector $\vec{O}$*) is an* $(n+1)$*-component vector, each component of which is either a value of type* D_I *(respectively* D_O*), or the distinguished value* $\perp$*. At least one component of* $\vec{I}$ *(respectively* $\vec{O}$*) must be different from* $\perp$*.*

We denote the i-th component of input vector $\vec{I}$ by $\vec{I}[i]$, and similarly for output vectors. An input vector $\vec{I}$ represents a possible assignment of input values to processes: if $\vec{I}[i]$ is an input value v, then v is P_i's input at the start of the execution, while if $\vec{I}[i]$ is $\perp$, then P_i does not participate in that execution: it has no input and takes no steps. Similarly, an output vector $\vec{O}$ represents a possible choice of output values by processes: if $\vec{O}[i]$ is an output value v, then v is P_i's output chosen during that execution, while if $\vec{I}[i]$ is $\perp$, then P_i does not choose an output in that execution. Vectors thus describe the input/output behavior of finite executions in which some subset of processes participate.

Definition 2.2 *A* participating *index (process) in an input vector is one whose value is distinct from* $\perp$*.*

Definition 2.3 *Vector* $\vec{U}$ matches $\vec{V}$ *if, for* $0 \leq i \leq n$*,* $\vec{U}[i] = \perp$ *if and only if* $\vec{V}[i] = \perp$*.*

Matching vectors have the same set of participating indexes. We are often concerned with executions that are prefixes of a given execution.

Definition 2.4 *Vector* $\vec{U}$ *is a* prefix *of* $\vec{V}$ *if, for* $0 \leq i \leq n$*, either* $\vec{U}[i] = \vec{V}[i]$ *or* $\vec{U}[i] = \perp$*.*

If a prefix has an entry distinct from $\perp$, then it agrees with the corresponding entry in the original.

Definition 2.5 *A set* V *of vectors is* prefix-closed *if for all* $\vec{V} \in V$*, every prefix* $\vec{U}$ *of* $\vec{V}$ *is in* V*.*

We use prefix-closed sets of vectors to characterize the legitimate sets of input and output value assignments. If the set of input vectors is prefix-closed, then any legitimate assignment of input values remains legitimate if fewer processes participate. If the set of output vectors is prefix-closed, then any legitimate choice of output values remains legitimate if fewer processes decide.

Definition 2.6 *Let I and O be prefix-closed sets of input and output vectors. A* task specification *is a relation $\Delta \subseteq I \times O$, carrying each input vector to a non-empty subset of* matching *output vectors.*

Let $\Delta(\vec{I})$ denote the set of vectors $\vec{O}$ in O such that $(\vec{I}, \vec{O}) \in \Delta$. For a given input vector $\vec{I}$, the set $\Delta(\vec{I})$ is just the set of legitimate output vectors corresponding to the inputs $\vec{I}$. Because task specifications are typically nondeterministic, $\Delta(\vec{I})$ typically contains multiple vectors.

We are now ready to give a precise definition of decision tasks.

Definition 2.7 *A* decision task $\langle I, O, \Delta\rangle$ *is a tuple consisting of a set I of input vectors, a set O of output vectors, and a task specification Δ relating these two sets.*

$\vec{I}$	$\Delta(\vec{I})$
$(0,\perp,\perp)$	$(0,\perp,\perp),(1,\perp,\perp),(2,\perp,\perp)$
$(\perp,0,\perp)$	$(\perp,0,\perp),(\perp,1,\perp),(\perp,2,\perp)$
$(\perp,\perp,0)$	$(\perp,\perp,0),(\perp,\perp,1),(\perp,\perp,2)$
$(0,0,\perp)$	$(0,1,\perp),(1,0,\perp),(0,2,\perp),(2,0,\perp),(2,1,\perp),(1,2,\perp)$
$(0,\perp,0)$	$(0,\perp,1),(1,\perp,0),(0,\perp,2),(2,\perp,0),(2,\perp,1),(1,\perp,2)$
$(\perp,0,0)$	$(\perp,0,1),(\perp,1,0),(\perp,0,2),(\perp,2,0),(\perp,2,1),(\perp,1,2)$
$(0,0,0)$	$(0,1,2),(0,2,1),(1,0,2),(1,2,0),(2,0,1),(2,1,0)$

Figure 1: The Unique-Id task

$\vec{I}$	$\Delta(\vec{I})$
$(0,\perp,\perp)$	$(0,\perp,\perp)$
$(\perp,0,\perp)$	$(\perp,0,\perp)$
$(\perp,\perp,0)$	$(\perp,\perp,0)$
$(0,0,\perp)$	$(0,1,\perp),(1,0,\perp)$
$(0,\perp,0)$	$(0,\perp,1),(1,\perp,0)$
$(\perp,0,0)$	$(\perp,0,1),(\perp,1,0)$
$(0,0,0)$	$(0,1,2),(0,2,1),(1,0,2),(1,2,0),(2,0,1),(2,1,0)$

Figure 2: The Fetch-And-Inc task

This class of decision tasks includes all linearizable *one-time objects*, that is linearizable objects [30] that permit at most one operation per process. The model captures the intuitive notion of "order of events in time" through the use of participating processes. For example, although the following tasks have the same sets of input and output vectors, they have a very different structure.

Unique-id Each participating process $P_i \in \{0, .., n\}$ has an input $x_i = 0$ and 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 $P_i \in \{0, .., n\}$ has an input $x_i = 0$ and chooses a unique output $y_i \in \{0, .., n\}$ such that (1) for some participating index i, $y_i = 0$, and (2) for $1 \leq k \leq n$, if $y_i = k$ then for some $j \neq i$, $y_j = k - 1$.

The tables in Figure 2 show the task specifications for *Unique-id* and *Fetch-And-Increment*. Notice that *Unique-Id* allows identifiers to be assigned statically, while *Fetch-And-Increment* effectively requires that they be assigned dynamically in increasing order. We will see that the first task has a trivial wait-free solution, while the second has no solution in read/write memory if one or more processes can fail.

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 [36]. 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 some initial state. An *execution fragment* is a subsequence of consecutive states and events occurring in an execution. For simplicity we will use the term execution to mean either execution or execution fragment, the appropriate term being clear from context. All executions considered in this paper are finite. 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 [36]).

An *object* X is an automaton with input events $\text{CALL}(P, v, X, T)$ and output event $\text{RETURN}(P, v, X, T)$ where P is a process id, v is a value, and X an object, and T is a type. A *process* P is an automaton with output events $\text{CALL}(P, v, X, T)$, and $\text{FINISH}(P, v)$ and input events $\text{RETURN}(P, v, X, T)$ and $\text{START}(P, v)$. A RETURN event *matches* an earlier CALL event in a history if the two events have the same type, name, and process id. An *operation* is a matching pair of CALL and RETURN events.

A *read/write memory* object M is an automaton with input events $\text{READ}(P, a)$ and $\text{WRITE}(P, a, v)$[1] and output event $\text{RETURN}(P, v)$. The read/write memory model we use in this paper is the standard *atomic snapshot memory*: an array a where each $\text{WRITE}(P, a, v)$ is an *update* of array element $a[P]$ to the value v, and each $\text{READ}(P, a)$ is an instantaneous *scan* operation returning the contents of the entire array a (see [1, 3] for details and [35] for a survey of why read/write memory models are all equivalent to atomic snapshot memory.)

A history is *sequential* if each call event is immediately followed by a matching response. (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 atomic snapshot memory is straightforward: any *scan* operation of array a returns for each element $a[P]$ the value of the last preceding *update* by process P, or $\perp$ if no such update existed. Each history H induces a partial "real-time" order $\prec_H$ on its operations: $op_0 \prec_H op_1$ if the output event for op_0 precedes the input event 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 for every history H, there exists a sequential history G with the same events as H, where $\prec_H \subseteq \prec_G$. Informally, a history is linearizable if it can be mapped to a sequential history by making each operation appear to take effect instantaneously at some point between its call and its response (see Herlihy and Wing [30] for details). An *atomic snapshot memory* object is one that is linearizable to the sequential object 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. At any point in the execution of a protocol, the state of each process is called its *local state*. The set of local states together with the state of the memory is called the protocol's *global state*. 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.

[1]Strictly speaking, $\text{READ}(P, a)$ is short for $\text{CALL}(P, \text{READ}, M, a)$, and similarly for WRITE.

Definition 2.8 *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 history of X if and only if $H \cdot q \cdot p \cdot G$ is a history of X (where "$\cdot$" is the concatenation operator).*

In this paper we use atomic snapshot memory, where *scan* (READ) operations commute with one another, as do memory *update* (WRITE) operations. This property 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.

2.3 Solvability

We are interested in characterizing when tasks can be solved by processes that are individually equivalent to Turing machines. A process is *active* at a point in an execution 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 execution. An execution is *t-faulty* if up to t processes become faulty.

Definition 2.9 *A protocol* solves *a decision task in an execution if the following condition holds. Let $\{P_i | i \in U\}$ be the set of 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, the outputs of the non-faulty processes 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 for N processes *wait-free solves* a decision task if it solves it in every t-faulty execution where $0 \leq t < N$. We will call such a protocol *wait-free* and henceforth use the term *solves* to mean *wait-free solves.*

It is convenient to assume that any wait-free protocol using read/write memory is expressed in the following *normal form.* The processes share an atomic snapshot memory array a whose $N = n+1$ 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 memory. 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 any subsequent round, each P_i executes a sequence of *steps*: (1) it *updates* $a[i]$ to its current local state, and (2) it atomically *scans* the elements of a, 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 3 shows a generic protocol in normal form.

Because the set I of input vectors is finite, any kind of wait-free atomic snapshot memory protocol can be expressed in normal form.[2] The memory locations $a[i]$ need only be of bounded size, though for our lower bound proofs we allow them to be unbounded. Without loss of generality, we may assume that each time a process performs an update operation it writes a unique value.

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

Figure 3: A Wait-Free Protocol in Normal Form

2.4 Simplicial Complexes

We start with a number of standard technical definitions taken mostly from standard undergraduate textbooks [37, 39].

A *vertex* $\vec{v}$ is a point in a high dimensional Euclidean space. A set $\{\vec{v}_0, \ldots, \vec{v}_n\}$ of vertexes is *affinely independent* if and only if the vectors $\vec{v}_1 - \vec{v}_0, \cdots, \vec{v}_n - \vec{v}_0$ are linearly independent.

Definition 2.10 *Let $\{\vec{v}_0, \ldots, \vec{v}_n\}$ be an affinely independent set of $n+1$ vertexes. We define the (geometric)* n-simplex S *spanned by $\vec{v}_0, \ldots, \vec{v}_n$ to*

[2] If the number of input vectors were unbounded, then the protocol would need an explicit termination test since it could be that there is no bound r on the maximal number of rounds necessary to complete a task.

be the set of all points x such that $x = \sum_{i=0}^{n} t_i \vec{v}_i$ where $\sum_{i=0}^{n} t_i = 1$ and $t_i \geq 0$ for all i.

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. For an example of vertexes[3] and simplexes see Figure 4. For simplicity, we often denote the simplex spanned by a set $\{\vec{v}_0, \ldots, \vec{v}_n\}$ of affinely independent vertexes as $(\vec{v}_0, \ldots, \vec{v}_n)$. The number n is called the *dimension* of the simplex S, and is often denoted by $\dim(S)$. For clarity, we often indicate the dimension of a simplex as a superscript: S^n denotes the simplex spanned by the vertexes in $\{\vec{v}_0, \ldots, \vec{v}_n\}$.

Any simplex T spanned by a subset of $\{\vec{v}_0, \ldots, \vec{v}_n\}$ is called a *face* of S, denoted $T \subseteq S$. The faces of S different from S itself are called the *proper* faces of S. In Figure 4 the 1-simplex spanned by the vertexes $\{P0, Q1\}$ is a proper face of the 2-simplex $\{P0, Q1, R2\}$. The union of the proper faces of S is called the *boundary* of S, and is denoted $Bd(S)$. The interior of S, denoted $Int(S)$, is defined by the set equation $Int(S) = S - Bd(S)$.

Definition 2.11 *Let $S^d = (\vec{s}_0, \ldots \vec{s}_d)$ be a d-simplex. Define $face_i(S^d)$, the i^{th}* face *of S^d, to be the $(d-1)$-simplex $(\vec{s}_0, \ldots, \hat{s}_i, \ldots, \vec{s}_d)$, where the circumflex denotes omission.*

As will soon become clear, we will use vertexes to model local process states, and simplexes to model consistent states of multiple processes involved in solving a decision task or in running a protocol in the atomic snapshot model. To model a collection of such states we need the concept of a geometric simplicial complex, or complex for short.

Definition 2.12 *A geometric simplicial complex $\mathcal{K}$ in a Euclidean space is a collection of geometric simplexes such that*

- *Every face of every simplex of $\mathcal{K}$ is also a simplex of $\mathcal{K}$.*
- *The intersection of any two simplexes of $\mathcal{K}$ is also a simplex of $\mathcal{K}$.*

We consider only finite complexes. The *dimension* of a complex $\mathcal{K}$, denoted $\dim(\mathcal{K})$, is the highest dimension of any of its simplexes, and is sometimes indicated explicitly by a superscript. An n-dimensional complex (or n-complex) is *pure* if every simplex is a face of some n-simplex. Except when noted, all complexes considered in this paper are pure. A simplex S in $\mathcal{K}$ with dimension $\dim(S) = \dim(\mathcal{K})$ is called a *principal* simplex.

[3] In the example we name vertexes with names like $P0$ and $Q1$ using a combination of process id (such as P or Q) and a "value" (such as 0 or 1). The reason will become clear in the sequel.

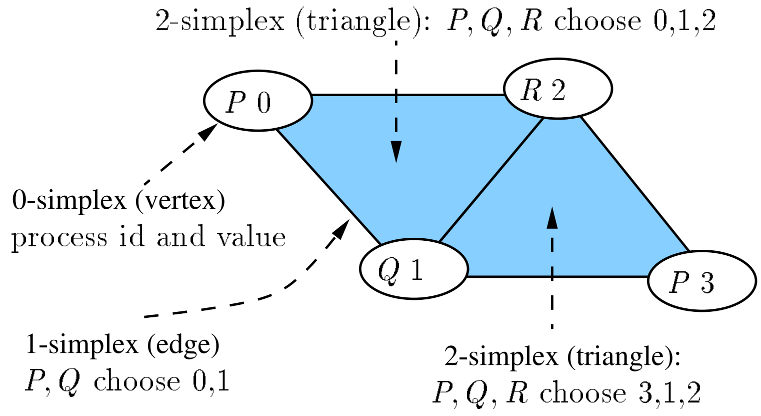

Figure 4: Vertexes and Simplexes

Figure 4 shows an example of a pure 2-dimensional complex consisting of the union of the faces of the 2-simplexes $(P0, Q1, R2)$ and $(P3, Q1, R2)$. Both 2-simplexes are principal simplexes in this example.

If $\mathcal{L}$ is a subcollection of simplexes in $\mathcal{K}$ closed under containment and intersection, then $\mathcal{L}$ is a complex in its own right, called a *subcomplex* of $\mathcal{K}$.

The set of simplexes of $\mathcal{K}$ of dimension at most ℓ is a subcomplex of $\mathcal{K}$, called the *ℓ-skeleton* of $\mathcal{K}$, denoted $skel^\ell(\mathcal{K})$. The elements of $skel^0(\mathcal{K})$ are the *vertexes* of $\mathcal{K}$. For example the 0-skeleton of $\mathcal{K}$ in Figure 4 is just $\{P0, P3, Q1, R2\}$. Similarly, the 1-skeleton of $\mathcal{K}$ is the union of the 0-skeleton and the set of 1-simplexes $\{(P0, Q1), (P0, R2)(R2, Q1)(P3, Q1)(P3, R2)\}$. We now define a way of "adding" simplexes, known as *joining.*

Definition 2.13 *Let $S = (\vec{s}_0, \ldots, \vec{s}_p)$ and $T = (\vec{t}_0, \ldots, \vec{t}_q)$ be simplexes whose combined sets of vertexes are affinely independent. Then the* join *of S and T, denoted $S \cdot T$ is the simplex $(\vec{s}_0, \ldots, \vec{s}_p, \vec{t}_0, \ldots, \vec{t}_q)$.*

We may extend the notion of joining to complexes as well.

Definition 2.14 *If $\mathcal{K}$ and $\mathcal{L}$ are simplicial complexes, not necessarily of the same dimension, then their* join, *denoted $\mathcal{K} \cdot \mathcal{L}$, is the collection of simplexes $\mathcal{K} \cup \mathcal{L} \cup \{S \cdot T \mid S \in \mathcal{K}, T \in \mathcal{L}\}$.*

The join of two complexes $\mathcal{K}$ and $\mathcal{L}$ is a complex in its own right [37]. One useful complex derived using the join operator is the *cone* over $\mathcal{K}$, defined as $\vec{v} \cdot \mathcal{K}$ for some vertex $\vec{v}$ affinely independent of $\mathcal{K}$. Let $|\mathcal{K}|$ be the subset $\bigcup_{S \in \mathcal{K}} S$ of a high-dimensional Euclidean space $\mathbf{R}^\ell$, that is, the union of the simplexes of $\mathcal{K}$. Giving each simplex its natural topology as a subspace of $\mathbf{R}^\ell$, we topologize $|\mathcal{K}|$ by declaring a subset A of $|\mathcal{K}|$ to be closed if and only if $A \cap S$ is closed for all $S \in \mathcal{K}$. This space is called the *polyhedron* of $\mathcal{K}$. Conversely, $\mathcal{K}$ is called a *triangulation* of $|\mathcal{K}|$. We define the *diameter* of a simplex S to be the maximum Euclidean distance between any pair of points of $|S|$.

Definition 2.15 *Two topological subspaces A and B are* homeomorphic *if there exists a one-to-one continuous map $f : A \to B$ with a continuous inverse.*

We say that a complex $\mathcal{C}$ is an *n-disk* if $|\mathcal{C}|$ is homeomorphic to $|\mathcal{S}^n|$, and it is an *$(n-1)$-sphere* if $|\mathcal{C}|$ is homeomorphic to $|\dot{\mathcal{S}}^{n-1}|$.

2.5 Abstract Simplexes and Complexes

The geometric representations we have given for simplexes and complexes are not always convenient, and we therefore, introduce the "complementary" notions of *abstract simplexes* and *abstract complexes.*

Definition 2.16 *An abstract simplex S is a finite, non-empty set.*

The *dimension* of S is one less than its cardinality. Each non-empty subset T of S is called a *face* of S. Each element of S is called a *vertex* of S. Geometric and abstract simplexes are closely related: any affinely-independent set of vectors $\{\vec{v}_0, \ldots, \vec{v}_n\}$ span both a geometric and abstract simplex.

Definition 2.17 *An abstract complex $\mathcal{K}$ is a collection of abstract simplexes closed under containment, that is, if S is in $\mathcal{K}$, so is any face of S.*

The notions of *dimension*, *join*, and *subcomplex* are defined for abstract simplexes and complexes in the obvious way.

Definition 2.18 *Let $\mathcal{G}$ be a geometric complex, and let V be the vertex set of $\mathcal{G}$. If $\mathcal{A}$ is the abstract complex that of all subsets S of V such that S spans a simplex in $\mathcal{G}$, then $\mathcal{A}$ is called the* vertex scheme *of $\mathcal{G}$.*

Definition 2.19 *Two abstract complexes $\mathcal{K}$ and $\mathcal{L}$ are* isomorphic *if there is a bijective correspondence ψ between their vertex sets such that a set S of vertexes is in $\mathcal{K}$ if and only if $\psi(S) \in \mathcal{L}$. The bijective correspondence ψ is called an* isomorphism*.*

The proof of the following theorem can be found in most standard textbooks on algebraic topology [37, 39].

Theorem 2.20 *Every abstract complex $\mathcal{A}$ is isomorphic to the vertex scheme of some geometric complex $\mathcal{G}$ in $\mathbf{R}^{2\dim(\mathcal{A})+1}$.*

2.6 Simplicial Maps and Subdivisions

In the rest of this paper, it is convenient to use abstract and geometric simplexes and complexes more or less interchangeably. The remainder of this section, however, focuses on geometric complexes. We first define the notions of vertex maps and simplicial maps.

Definition 2.21 *Let $\mathcal{K}$ and $\mathcal{L}$ be complexes, possibly of different dimensions. A* vertex map *$\mu : skel^0(\mathcal{K}) \to skel^0(\mathcal{L})$ carries vertexes of $\mathcal{K}$ to vertexes of $\mathcal{L}$. The map is a* simplicial map *if it also carries simplexes of $\mathcal{K}$ to simplexes of $\mathcal{L}$. Any simplicial map μ induces a piece-wise linear map $|\mu| : |\mathcal{K}| \to |\mathcal{L}|$ as follows. Every point $\vec{k}$ of $|\mathcal{K}|$ has a unique representation as*

$$\vec{k} = \sum k_i \cdot \vec{k}_i$$

where the $\vec{k}_i$ span a simplex in $\mathcal{K}$, $0 < k_i \leq 1$, and $\sum k_i = 1$. The k_i are called the barycentric coordinates *of $\vec{k}$. Define*

$$|\mu|(\vec{k}) = \sum k_i \cdot \mu(\vec{k}_i)$$

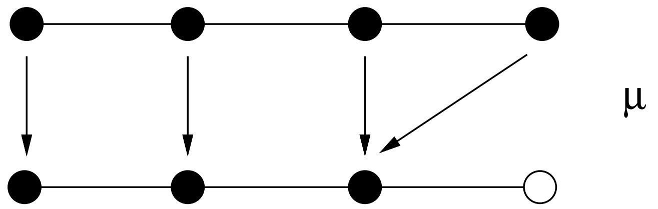

Figure 5: A Simplicial Map

Henceforth, unless stated otherwise, all maps between complexes are assumed to be simplicial. An example of a simplicial map is given in Figure 5.

Definition 2.22 *Let $\mathcal{A} \subset \mathcal{B}$, and $\phi : \mathcal{A} \to \mathcal{C}$. A simplicial map $\psi : \mathcal{B} \to \mathcal{C}$* extends *$\phi$ if they agree on $\mathcal{A}$.*

We note that a simplex and its image under a simplicial map need not have the same dimension. As seen in Figure 5, the simplicial map may "collapse" some simplexes.

Definition 2.23 *A simplicial map* $\phi : \mathcal{B} \to \mathcal{C}$ collapses *a simplex* T^m, $m > 0$, *if* $\dim(\phi(T^m)) < \dim(T^m)$.

A simplicial map $\mu : \mathcal{K} \to \mathcal{L}$ is *non-collapsing* if it preserves dimension, that is, for all $S \in \mathcal{K}$: $\dim(\mu(S)) = \dim(S)$.

Definition 2.24 *A* coloring *of an* n*-dimensional complex* $\mathcal{K}$ *is a non-collapsing simplicial map* $\chi : \mathcal{K} \to \mathcal{S}$*, where* S *is an* n*-simplex.*

Intuitively, a coloring corresponds to a labeling of the vertexes of the complex such that no two neighboring vertexes (vertexes connected by a 1-simplex) have the same label. A *chromatic complex* or *colored complex* $(\mathcal{K}, \chi)$ is a complex $\mathcal{K}$ together with a coloring χ of $\mathcal{K}$. An example of a chromatic complex is given in Figure 33, where the colors are the letters $\{P, Q, R\}$. When it is clear from the context, we specify the chromatic complex $(\mathcal{K}, \chi)$ simply as the complex $\mathcal{K}$, omitting explicit mention of the coloring χ.

Definition 2.25 *Let* $(\mathcal{K}, \chi_{\mathcal{K}})$ *and* $(\mathcal{L}, \chi_{\mathcal{L}})$ *be chromatic complexes, and let* $\mu : \mathcal{K} \to \mathcal{L}$ *be a simplicial map. We say that* μ *is* color-preserving *(or* chromatic*) if, for every vertex* $\vec{v} \in \mathcal{K}$, $\chi_{\mathcal{K}}(\vec{v}) = \chi_{\mathcal{L}}(\mu(\vec{v}))$.

In other words, μ is color-preserving if it maps each vertex in $\mathcal{K}$ to a vertex in $\mathcal{L}$ of the same color. Except when otherwise noted, all simplicial maps considered in this paper are color-preserving.

Definition 2.26 *Let* $\mathcal{K}$ *be a complex in* $\mathbf{R}^\ell$. *A complex* $\sigma(\mathcal{K})$ *is a* subdivision *of* $\mathcal{K}$ *if:*

- *Each simplex in* $\sigma(\mathcal{K})$ *is contained in a simplex in* $\mathcal{K}$.
- *Each simplex of* $\mathcal{K}$ *is the union of finitely many simplexes in* $\sigma(\mathcal{K})$.

One type of subdivision of particular interest in our proof is the barycentric subdivision from classical algebraic topology [37]. Let $S^n = \{\vec{s}_0, \ldots, \vec{s}_n\}$ be an n-simplex of some complex $\mathcal{K}$. The point

$$\vec{b} = \sum_{i=0}^{n} (\vec{s}_i/(n+1))$$

is the *barycenter* of S^n. In particular, if S^n is a vertex, then $\vec{b} = S^n$.

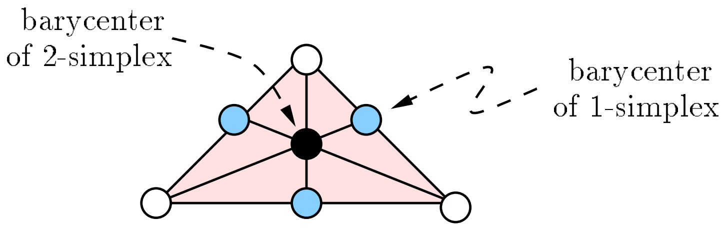

Figure 6: A Barycentric Subdivision of a 2-simplex

Definition 2.27 *The* barycentric subdivision *of a complex $\mathcal{K}$, denoted $\beta(\mathcal{K})$ is defined as follows. Its vertexes are the barycenters of the simplexes of $\mathcal{K}$. For each ordered sequence $S_0, \ldots, S_m$ of simplexes of $\mathcal{K}$ where S_i is a face of S_{i+1} $(i = 0, \ldots, m-1)$, the sequence of corresponding barycenters is the set of vertexes of a simplex of $\beta(\mathcal{K})$. Only simplexes obtained in this manner are in $\beta(\mathcal{K})$.*

Figure 6 shows $\beta(\mathcal{S})$ defined on a 2-complex $\mathcal{S}$.

Definition 2.28 *If S is a simplex of $\sigma(\mathcal{K})$ or a point in $|\mathcal{K}|$, the* carrier *of S, denoted carrier$(S, \mathcal{K})$ is the unique smallest $T \in \mathcal{K}$ such that $S \subset |T|$.*

Figure 7 illustrates the notions of subdivisions and carriers. It shows a complex on the right, and a subdivision on the left, and highlights a simplex S in the subdivision together with its carrier.

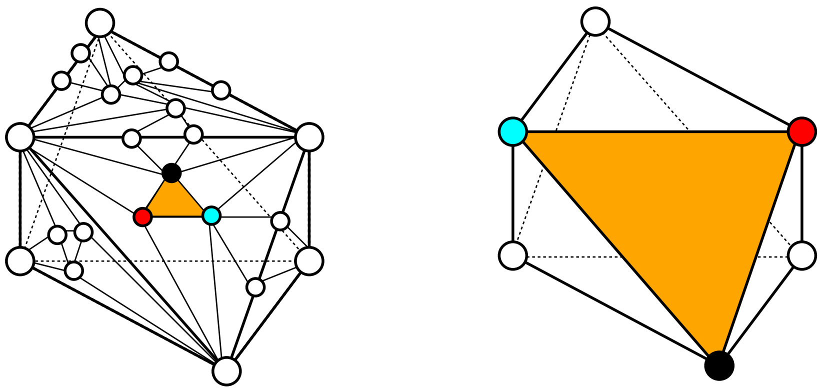

Figure 7: A simplex and its carrier

One drawback of the barycentric subdivision is that a barycentric subdivision of a chromatic complex is typically not chromatic.

Definition 2.29 *A* chromatic subdivision *of* $(\mathcal{K}, \chi_{\mathcal{K}})$ *is a chromatic complex* $(\sigma(\mathcal{K}), \chi_{\sigma(\mathcal{K})})$ *such that* $\sigma(\mathcal{K})$ *is a subdivision of* $\mathcal{K}$*, and for all* S *in* $\sigma(\mathcal{K})$*,* $\chi_{\sigma(\mathcal{K})}(S) \subseteq \chi_{\mathcal{K}}(\mathit{carrier}(S, \mathcal{K}))$*.*

Figure 33 shows a chromatic subdivision of a complex $\mathcal{S}$ defined on a 2-simplex S. We call this specific type of subdivision the *standard chromatic subdivision* and will use it extensively later in the paper. All subdivisions we consider will be chromatic unless, as in the case of the barycentric subdivision, we explicitly state otherwise.

Definition 2.30 *A simplicial map* $\mu : \sigma_1(\mathcal{K}) \to \sigma_2(\mathcal{K})$ *between chromatic subdivisions of* $\mathcal{K}$ *is* carrier-preserving *if for all* $S \in \sigma_1(\mathcal{K})$*,* $\mathit{carrier}(S, \mathcal{K}) \subseteq \mathit{carrier}(\mu(S), \mathcal{K})$*.*

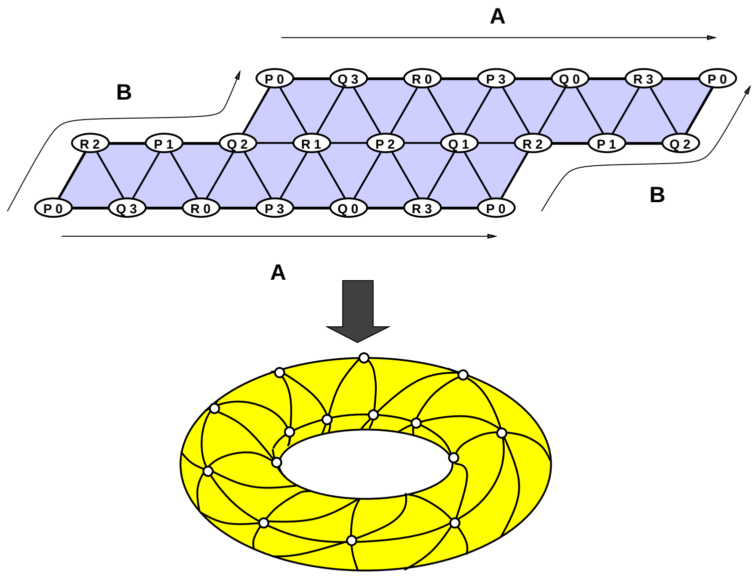

Figure 8: Output Complex for 3-Process Renaming with 4 Names

2.7 Simplicial Complexes and Tasks

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. We present a topological specification that replaces the vector-based task specification of Section 2.1. A formal proof of the correspondence among the two representations is beyond the scope of this paper and can be found in [31].

We will construct the topological specification using abstract simplexes and complexes. The reader should note that it follows from Theorem 2.20 that there exists a representation using geometric simplexes and complexes, for which the vertex scheme is isomorphic to the abstract representation. To illustrate our constructions, we accompany the formal definitions with examples of how they are used to represent the following variant[4] of the well-known Renaming decision task first introduced and studied by Attiya et. al [7].

Renaming The input to each process is a unique *input name* in the range $\{0, \ldots, M-1\}$ Each participating process chooses a unique *output name* taken from a strictly smaller range $0, \ldots, K$.

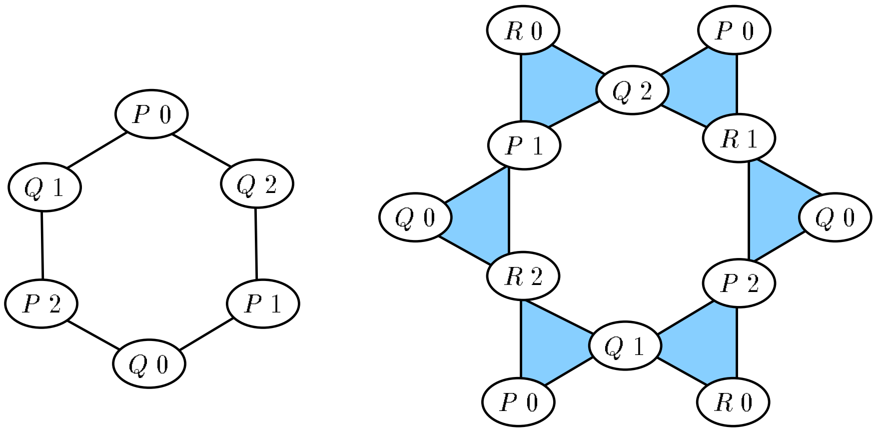

Figure 9: Some output complexes for the renaming task

Definition 2.31 *Let $\vec{I} \in I$ be an input vector. The* input simplex *corresponding to $\vec{I}$, denoted $\mathcal{S}(\vec{I}\,)$, is the abstract colored simplex whose vertexes $\langle P_i, v_i \rangle$ correspond to the participating entries in $\vec{I}$, for which $\vec{I}[i] = v_i \neq \perp$.*

The *output simplex* corresponding to $\vec{O}$, denoted $\mathcal{S}(\vec{O})$, is defined accordingly. The process id labeling of a vertex $\vec{v}$ is denoted by $id(\vec{v})$, and the

[4]The full renaming problem is treated in the sequel. It assumes an additional symmetry property.

value by $val(\vec{v})$. We use $ids(S)$ to denote the set of process ids of vertexes in a simplex S, and $vals(S)$ to denote the set of values. Figure 4 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., [9]), 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.

Since the sets of input vectors we consider are prefix-closed, we can collect input and output vectors into abstract chromatic complexes (i.e. sets of simplexes closed under containment).

Definition 2.32 *The* input complex *corresponding to I, denoted $\mathcal{I}$, is the collection of input simplexes $\mathcal{S}(\vec{I})$ corresponding to the input vectors of I. The* output complex *corresponding to O, denoted $\mathcal{O}$, is the collection of output simplexes $\mathcal{S}(\vec{O})$ corresponding to the output vectors of O.*

For example, Figures 8 and 9 show the output complexes for renaming with two processes $\{P, Q\}$ using three names $\{0, 1, 2\}$, three processes $\{P, Q, R\}$ with three names $\{0, 1, 2\}$, and three processes using the four names $\{0, 1, 2, 3\}$. The four-name output complex's polyhedron is homeomorphic (topologically equivalent) to a torus. To see why, notice in Figure 8 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 now construct a topological equivalent of the task specification map $\Delta \subseteq I \times O$.

Definition 2.33 *The* topological task specification *corresponding to the task specification Δ, denoted $\Delta \subseteq \mathcal{I} \times \mathcal{O}$, is defined to contain all pairs $(\mathcal{S}(\vec{I}), \mathcal{S}(\vec{O}))$ where $(\vec{I}, \vec{O})$ are in the task specification Δ.*

Note that the topological task specification Δ is not a simplicial map.

We can now put these definitions together, as shown schematically in Figure 10.

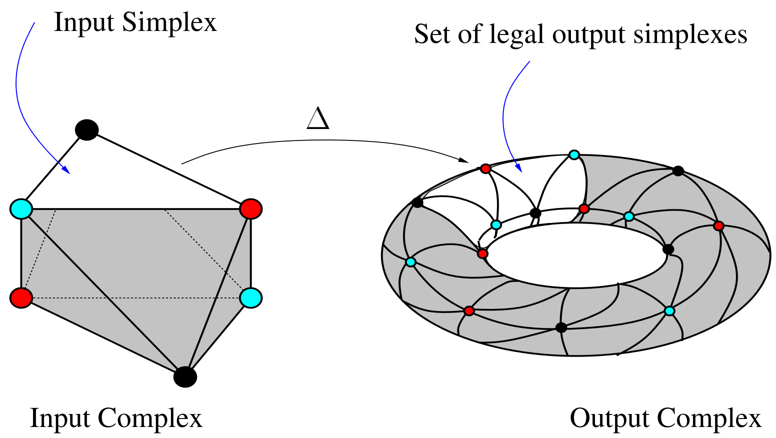

Figure 10: A Decision Task

Definition 2.34 *Given an $(n+1)$-process decision task $\langle I, O, \Delta\rangle$, the corresponding* topological representation *of the task, denoted $\langle \mathcal{I}, \mathcal{O}, \Delta\rangle$, consists of an input complex $\mathcal{I}$ corresponding to I, and output complex $\mathcal{O}$ corresponding to O, and a topological task specification Δ corresponding to the task specification Δ.*

Usually, we simply refer to a topological task specification as a "task specification".

Definition 2.35 *Let U be a set of processes. A* solo *execution by U is one in which 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 $ids(S^m)$.*

2.8 Links, Manifolds, and Connectivity

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 a simplex $S \in \mathcal{C}$, written $st(S, \mathcal{C})$, is the union of all $|T|$ such that $S \subseteq T$ (see Figure 11). Although stars are defined as polyhedrons, we sometimes treat them as simplicial complexes, relying on context to clarify the precise meaning. The *open star*, written $st^\circ(S, \mathcal{C})$, is the interior of the star. The *link*, written $lk(S, \mathcal{C})$, is the complex consisting of all simplexes in $st(S, \mathcal{C})$ that contain no vertex of S. These concepts are illustrated in Figure 11. The notion of a link has a simple interpretation. In the renaming complex shown in Figure 8, consider the node labeled P 2. This node indicates that 2 is a correct output for P. The link of this node is a one-dimensional complex (a hexagon) in which each 1-simplex represents a possible combination of legal outputs for the remaining processes Q and R. In general, in any chromatic complex $\mathcal{C}$, $lk(S, \mathcal{C})$ is a colored complex with the following interpretation: if we fix the values assigned to processes in $ids(S)$, then $lk(S, \mathcal{C})$ represents all possible legal ways to assign values to the remaining processes.

A complex $\mathcal{A}$ is an *n-manifold with boundary* if

1. for every pair of n-simplexes T_0^n, T_1^n in $\mathcal{A}$, 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. every $(n-1)$-simplex is contained in either one or two n-simplexes.

The *boundary complex* of a manifold with boundary is the subcomplex of $(n-1)$-simplexes contained in exactly one n-simplex. If the boundary complex is empty, we refer to the complex simply as a *manifold.*

Some, but not all, the complexes we consider are manifolds. Manifolds satisfy the following property [21, Theorem II.2]:

Lemma 2.36 *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.*

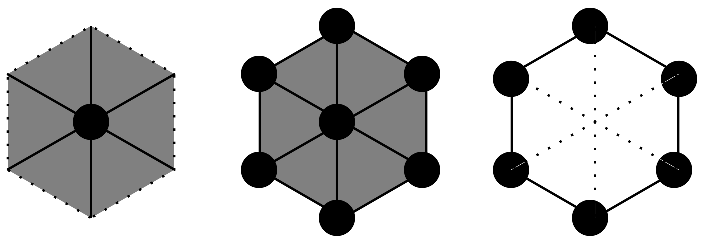

Figure 11: $st^\circ(\vec{v})$, $st(\vec{v})$, and $lk(\vec{v})$.

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.

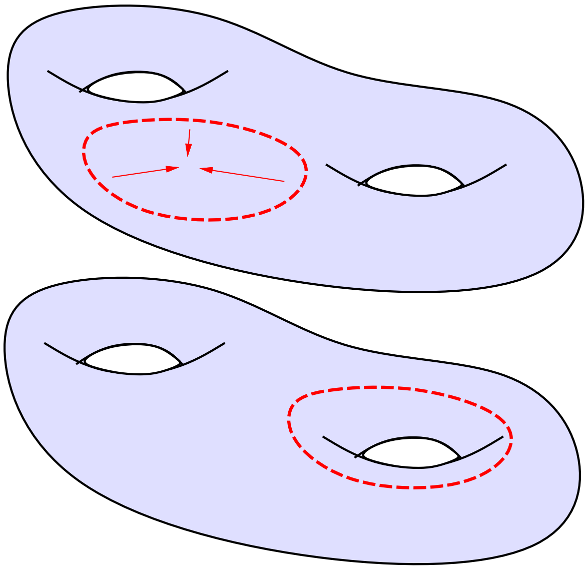

Figure 12: Contractible and Non-Contractible Loops

Definition 2.37 *For $k > 0$, a non-empty complex $\mathcal{C}$ is* k-connected *[39, p.51] if, for $m \leq k$, any continuous map of the m-sphere into $|\mathcal{C}|$ can be extended to a continuous map over the $(m+1)$-disk. It is convenient to define a complex to be* (-1)-connected *if it is non-empty, and any complex to be k-connected for $k < -1$.*

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.

To illustrate how this formal definition captures the informal notion of a "hole" in a complex, Figure 12 shows a complex (rendered as a surface for simplicity) together with images of a 1-sphere (circle) under continuous maps f and g. The image under f can be contracted to a point, while the image under g circumnavigates a "hole" and cannot be contracted. We will be concerned with proving that certain complexes are k-connected. Although the definition of k-connectivity is topological in nature, we can reason about connectivity in a purely combinatorial way, using the following elementary theorem, proved in the appendix.

Theorem 2.38 *If $\mathcal{K}$ and $\mathcal{L}$ are complexes such that $\mathcal{K}$ and $\mathcal{L}$ are k-connected, and $\mathcal{K} \cap \mathcal{L}$ is $(k-1)$-connected, then $\mathcal{K} \cup \mathcal{L}$ is k-connected.*

Before continuing, we note some examples of useful k-connected complexes.

Lemma 2.39 *The following complexes are k-connected: (1) the complex $\mathcal{S}^k$ consisting of a k-simplex and its faces, and (2) a cone over an arbitrary $(k-1)$-dimensional complex.*

This completes the topological concepts necessary to prove our main theorem.

3 The Main Theorem

We are now ready to state our main theorem.

Theorem 3.1 (Asynchronous Computability Theorem) *A decision task $\langle \mathcal{I}, \mathcal{O}, \Delta \rangle$ has a wait-free protocol using read-write memory if and only if there exists a chromatic subdivision σ of $\mathcal{I}$ and a color-preserving simplicial map*

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

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

This theorem establishes that task solvability can be characterized in terms of purely topological properties of the task specification, without explicit mention of protocols and executions. A task is solvable if and only if one can subdivide the input complex and map that subdivided complex to the outputs in a way that agrees with Δ. In one direction, we will see that any protocol induces a subdivision, and the structure of that subdivision reflects in a natural way the structure of the protocol.

The theorem is depicted schematically in Figure 14. The figure's top half shows how the task specification Δ takes input simplexes to allowed output simplexes. The bottom half shows how μ maps the subdivided input complex to the output complex in a way consistent with Δ.

In Figure 13, we give a simple example illustrating how the subdivision mentioned in the theorem reflects the unfolding of a protocol execution. Consider the following 2-process task $(\mathcal{I}, \mathcal{O}, \Delta)$, solved by a single-round normal-form protocol. P and Q have respective inputs p and q. They share a two-element array, both elements initialized to $\perp$. P writes its value to the first array element, and then scans the array, while Q writes its value to the second element, and then scans the array. Define a process's *view* to be the sequence of values it read at the end of the protocol. A one-round, two-process normal-form protocol has only three possible executions: (1) If P

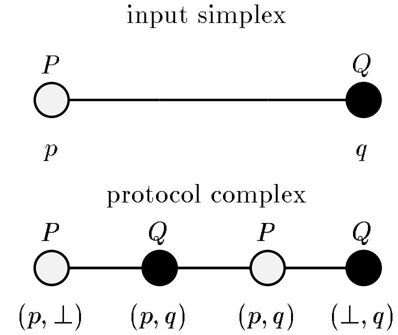

Figure 13: Simplicial representation of a one-round execution

reads before Q writes, then P's view is $(p, \perp)$, and Q's is (p, q). (2) If P and Q each reads after the other writes, then both have view (p, q). (3) Q reads before P writes, then P's view is (p, q), and Q's is $(\perp, q)$. These executions define a *protocol complex* $\mathcal{P}$ as follows. There are three 1-simplexes, one for each execution. Each vertex is labeled with a process id and that process's view in that execution. If P reads (p, q), then it cannot "tell" from its view whether P executed before Q or concurrently with Q. The complex $\mathcal{P}$ captures this ambiguity in a geometric way by placing $\langle P, pq\rangle$ in the intersection of the two 1-simplexes representing these two executions. If we identify vertexes $\langle P, p\rangle$ and $\langle Q, q\rangle$ of $\mathcal{I}$ with vertexes $\langle P, p\perp\rangle$ and $\langle Q, \perp q\rangle$ of $\mathcal{P}$, we can see that $\mathcal{P}$ is a subdivision of $\mathcal{I}$, dividing the single edge of $\mathcal{I}$ into three edges.

When a process finishes executing the protocol, it chooses an output value based only on its local state. Formally, this choice is captured by a *decision map* δ carrying vertexes of $\mathcal{P}$ to vertexes of $\mathcal{O}$. Recall that a fixed execution corresponds to a single simplex of $\mathcal{P}$. At the end of this execution, the processes must choose vertexes that lie on a common simplex of $\mathcal{O}$, implying that the vertex map $\delta : \mathcal{P} \to \mathcal{O}$ is a simplicial map. Moreover, because the protocol solves the task, for every simplex S in $\mathcal{P}$, $\delta(S) \in \Delta(\mathit{carrier}(S, \mathcal{I}))$, satisfying the theorem's conditions. In summary, the one-round normal-form protocol defines a protocol complex $\mathcal{P}$ which is the desired subdivision of $\mathcal{I}$, and the decision map defines the desired map to the output complex.

Although this construction of a subdivision and map for a two-process one-round protocol is only an example, the proof of the "only if" part of the theorem is based on the same notions: the protocol itself defines a *protocol complex* which encompasses a subdivision of the input complex having the desired properties.

The proof of the asynchronous computability theorem appears in Section 4 and 5. In the remainder of this section we give examples of applications of the asynchronous computability theorem.

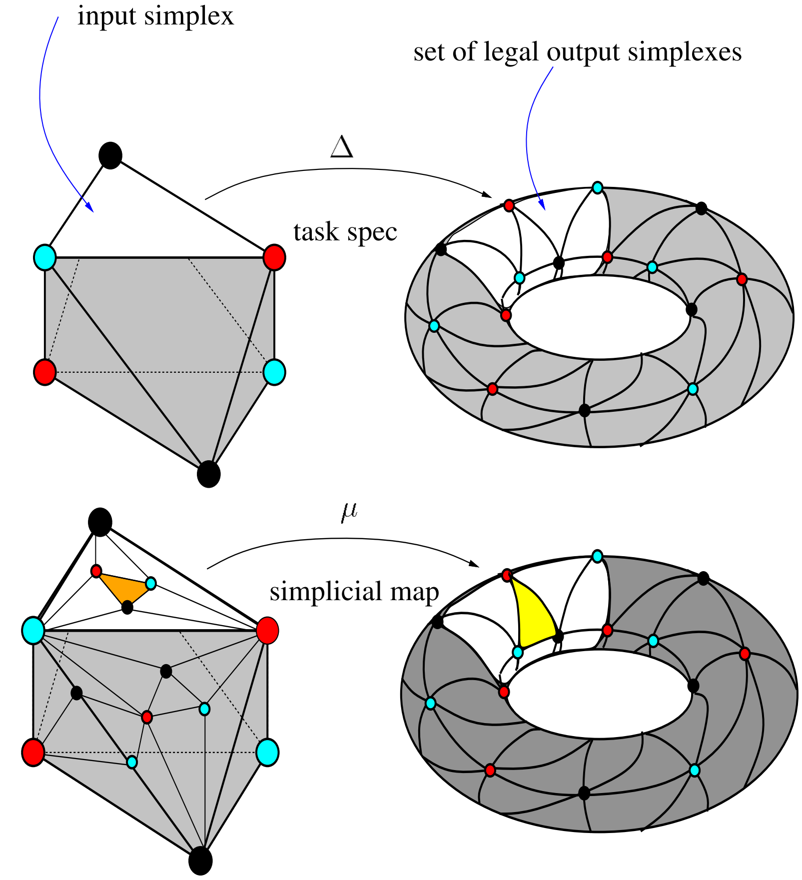

Figure 14: Asynchronous Computability Theorem

3.1 Binary Consensus

Perhaps the simplest decision task is binary *consensus* [19]. As specified in Figure 15, each process starts with a binary input, and eventually chooses a binary output. All output values must agree, and each output must be some process's input.

The input complex for this task is the complex $\mathcal{B}^n$ constructed by as-

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

Figure 15: The Consensus task

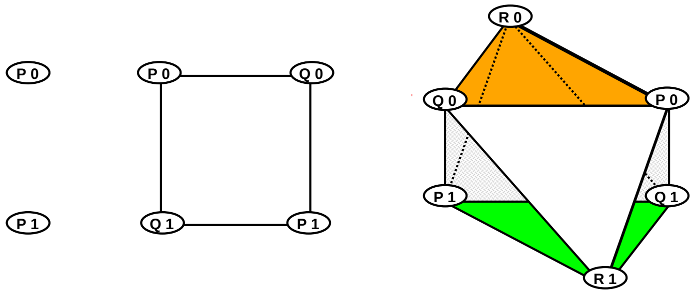

Figure 16: Binary 0, 1, and 2-spheres

signing independent binary values to $n+1$ processes. We call this complex the *binary n-sphere* (Figure 16).[5]

The output complex consists of two disjoint n-simplexes, corresponding to decision values 0 and 1. Figure 17 shows the input and output complexes for 2-process binary consensus. In general, the input complex is $(n-1)$-connected, while the output complex is disconnected. *Consensus* is the generalization of binary consensus to allow input values from an arbitrary range, not only $\{0, 1\}$.

It is well-known that binary consensus has no wait-free read-write protocol [15, 24, 34]. Nevertheless, it is instructive to see how this result follows from the asynchronous computability theorem. To keep our presentation simple, 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 must choose a value even if Q never takes a step, P must eventually choose 0. The same is true if Q runs solo

[5] Informally, to see why this complex is homeomorphic to an n-sphere, note that it consists of two subcomplexes: $\mathcal{E}_0^n$ 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.

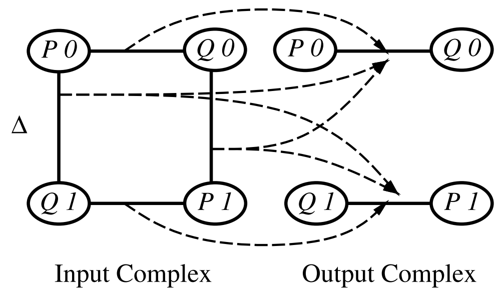

Figure 17: Simplicial Complexes for 2-Process Consensus

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

Figure 18: The Quasi-Consensus task

with input 1. If, However, 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 of a "bad" execution 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 rather than operational way. Figure 17 shows the input and output complexes for the two-process consensus task. Assume by way of contradiction that a protocol exists. The input complex $\mathcal{I}$ is connected, and so is the subdivision $\sigma(\mathcal{I})$. Simplicial maps preserve connectivity, so $\mu(\sigma(\mathcal{I}))$ 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 3.1, 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, 34].)

3.2 Quasi-Consensus

We introduce the following "toy" problem to illustrate further 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 agree, or Q may decide 0 and P may decide 1 (but not vice-versa).

Figure 19 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 20, 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 21 shows a simple protocol for quasi-consensus. Recall our earlier explanation that the subdivisions of an input simplex correspond to executions of the protocol. 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}.

This approach can be extended to tasks involving more than two processes. 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 the k-set agreement task follows from an similar observation: a simplicial map cannot carry the boundary of a "solid" disk to the boundary of a "hole".

3.3 Set Agreement

The *k-set agreement* task [13] is a natural generalization of consensus.

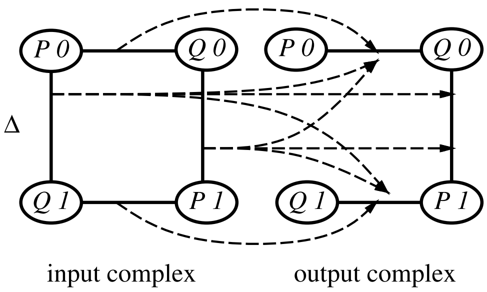

Figure 19: Input and Output Complexes for 2-Process Quasi-Consensus

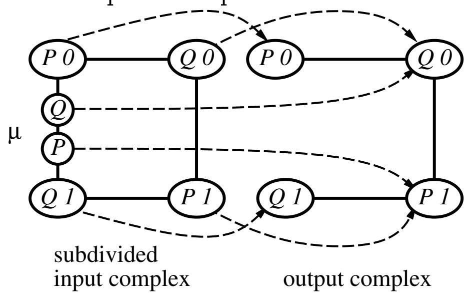

Figure 20: Subdivided Input and Output Complexes for 2-Process Quasi-Consensus

k-Set Agreement Like consensus, each process starts with an input value from some domain, and must choose some some process's input as its output. Unlike consensus, all processes together may choose no more than k distinct output values.

Consensus is just 1-set agreement: all processes together may choose no more than $k = 1$ distinct values. When $n = 2$ and $k = 2$, the three processes must return at most two distinct values. As illustrated in Figure 22, the output complex for three-process two-set agreement consists of three binary 2-spheres linked in a ring. This complex is not 1-connected — note the "hole" created by the missing simplexes colored with all three values.

The set agreement problem was first proposed by Chaudhuri [13] in 1989, along with a conjecture that it could not be solved when $k \leq n$. This problem remained open until 1993, when three independent research teams, Borowsky and Gafni [11], Herlihy and Shavit [28], and Saks and Zaharoglou [38] proved this conjecture correct.

We now show that the k-set agreement task of Chaudhuri has no wait-free read-write protocol when $k \leq n$. The proof we present is short and rather

```
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 21: Quasi-Consensus Protocols for P and Q

simple since most of the complexity is hidden in the use of the asynchronous computability theorem. Our proof uses *Sperner's Lemma* [33, Lemma 5.5], a standard tool from algebraic topology.

Lemma 3.2 (Sperner's Lemma) *Let $\sigma(T)$ be a subdivision of an n-simplex T. If $F : \sigma(T) \to T$ is a map sending each vertex of $\sigma(T)$ to a vertex in its carrier, then there is at least one n-simplex $S = (\vec{s}_0, \ldots, \vec{s}_n)$ in $\sigma(T)$ such that the $F(\vec{s}_i)$ are distinct.*

We begin with an informal sketch of the three-process case, where $k = 2$, and input and output values taken from $\{0, 1, 2\}$. Figure 22 shows the output complex.

Figure 24 shows a subcomplex of the input complex consisting of a simplex T with vertexes $\langle P, 0\rangle, \langle Q, 1\rangle$ and $\langle R, 2\rangle$ ($P0$, $P1$, and $R2$ for short), and a collection of 2-simplexes that intersect it along its proper faces. In simplex S_0, all processes have input value 0, and similarly for S_1 and S_2. In simplex S_{01}, all processes have input value 0 or 1, and similarly for S_{12} and S_{02}.

Now assume by way of contradiction that a protocol exists. Recall that the task specification requires each process to decide some process's input. By the asynchronous computability theorem, there exist subdivision σ and color-preserving simplicial map μ consistent with the task specification.

For S_0, the task specification requires each process to decide 0, so for every vertex of $\vec{s}$, $val(\mu(\vec{s})) = 0$. As a result, μ sends the vertex $P0$ of T to an output vertex also labeled with 0. Similarly, $\mu(Q1)$ and $\mu(R2)$ are respectively labeled with 1 and 2.

For S_{01}, the task specification requires each process to decide 0 or 1, so for every vertex of $\vec{s}$, $val(\mu(\vec{s})) \in \{0, 1\}$. As a result, μ sends every vertex in the subdivided edge $\sigma(P0, Q1)$ to an output vertex labeled with 0 or 1.

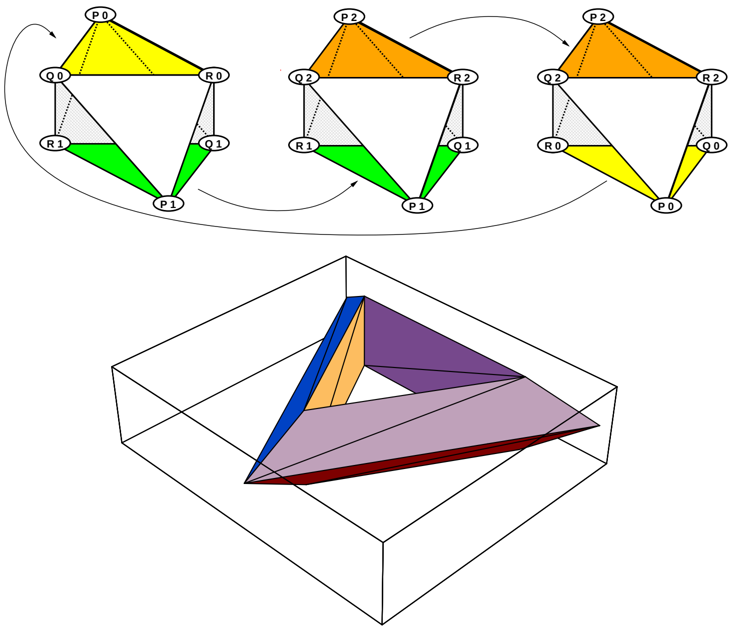

Figure 22: Output Complex for (3,2)-Set Agreement

Similarly, the vertexes of $\mu(\sigma(Q1, R2))$ and $\mu(\sigma(P0, R2))$ are respectively labeled with values from $\{1, 2\}$ and $\{0, 2\}$.

As a result, μ carries each vertex in each subdivided proper face of T to a value in its carrier's set of inputs, as depicted in Figure 23. The map μ thus satisfies the preconditions of Sperner's Lemma, and therefore it carries some 2-simplex in the subdivision $\sigma(T)$ to an output simplex labeled with all three values. The output complex, however, contains no such three-colored simplex, because there is no execution in which three distinct output values are chosen. (Even less formally, it maps to the "hole" in the output complex.)

Here is the full proof.

Theorem 3.3 *The k-set agreement task has no wait-free read-write protocol for $k \leq n$.*

Proof: It suffices to prove that there is no solution for $k = n$. Assume by way of contradiction that there is such a protocol. From the k-set agreement task specification, there is an input n-simplex T^n in $\mathcal{I}^n$ with $n+1$ distinct inputs ($|vals(T^n)| = n+1$). For every proper face $T^m \subset T^n$, there exists an input simplex $S^n \subset \mathcal{I}^n$ such that $T^m \subset S^n$ and $vals(T^m) = vals(S^n)$. For

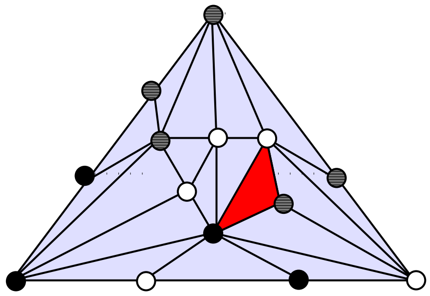

Figure 23: Sperner's Lemma: at least one simplex has all colors

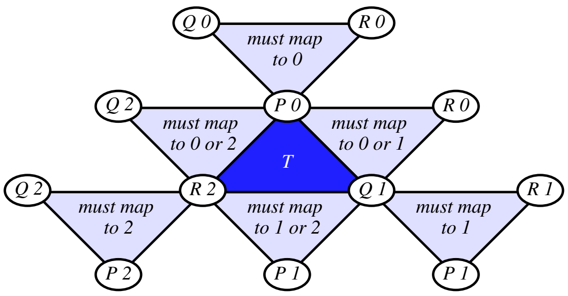

Figure 24: Part of the set agreement input complex.

example, if T^m is a single vertex $\langle P, 1\rangle$, then S^n is the input simplex with $vals(S^n) = \{1\}$. By Theorem 3.1 there exists a color-preserving simplicial map $\mu : \sigma(\mathcal{I}^n) \to \mathcal{O}^n$, and by definition, $\mu(\sigma(T^m))$ must be consistent with $\Delta(S^n)$ for any $S^n \subseteq \mathcal{I}^n$ containing T^m. By the task specification $vals(\Delta(S^n) \subseteq vals(S^n)$ and it follows that the simplicial map μ carries every vertex v of $\sigma(T^m)$ to a vertex in its carrier, hence by Lemma 3.2, there exists a simplex in $\sigma(T^n)$ whose vertexes are mapped to $n+1$ distinct inputs, that is, to a simplex in $\mathcal{O}$ with $n + 1$ distinct values, a contradiction. ∎

4 Necessity

In this section, we show that the conditions of our theorem are necessary: any decision task $\langle \mathcal{I}, \mathcal{O}, \Delta\rangle$ has a wait-free protocol using read/write memory *only if* there exists a chromatic subdivision $\sigma(\mathcal{I})$ and a color-preserving simplicial map

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

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

In our informal discussion of Figure 13, we represented the three possible executions of a one-round normal form protocol as a simplicial complex which we called the "protocol complex." We observed that this protocol complex induces a subdivision of an input simplex, and that the decision map δ from the protocol complex to the output complex is a simplicial map satisfying the conditions of the theorem.

Our lower-bound proof is just a formalization of the same argument. We first define the protocol complex for a normal-form protocol with multiple processes and multiple rounds. We then show that the protocol complex encompasses a subdivided image of the input complex, and that the decision map induces the desired simplicial map.

First, a note about notational conventions. Some of the results presented here concern properties of arbitrary complexes, while others concern properties of complexes that arise in the context of asynchronous computation. To highlight this distinction, we use symbols such as p and q for dimensions of arbitrary simplexes, while n, as usual, is one less than the number of processes, and m typically ranges between 0 and n.

4.1 Protocol Complexes

At each step in a protocol, the local state of a process consists of its input value together with the sequences of values it scanned. The protocol's global state is just the set of local states, together with the state of the shared atomic snapshot memory $a[0..n]$. It is useful to treat any protocol as an "uninterpreted" protocol in which each process's decision value is just its final local state (bypassing the decision map δ).

We model protocols just like decision tasks. The inputs and outputs for any execution of a protocol $\mathcal{P}$ are given by sets of $(n+1)$-process input and output vectors, respectively denoted by I and O. As noted in Section 2, because the protocols solve decision tasks, the set I of possible input vectors is prefix-closed. For any protocol, the set O of possible output vectors from all executions of the protocol must also be prefix-closed, for the following reason. Let $\vec{O}$ be an output vector produced by an execution of the protocol, and $\vec{P}$ any prefix. For each i such that $\vec{O}[i] \neq \perp$ and $\vec{P}[i] = \perp$, we can create a new execution in which process i fails just before the FINISH event, that is, just before deciding. Clearly, this execution is a possible execution of $\mathcal{P}$ with output vector $\vec{P}$.

Because the sets of input and output vectors I and O associated with a protocol $\mathcal{P}$ are prefix-closed, there exist corresponding input and output complexes, respectively denoted $\mathcal{I}$ and $\mathcal{P}(\mathcal{I})$.

Definition 4.1 *The complex $\mathcal{P}(\mathcal{I})$ is called a* protocol complex *over $\mathcal{I}$. Similarly, for a subcomplex $\mathcal{C}$ of the input complex $\mathcal{I}$, $\mathcal{P}(\mathcal{C})$ denotes the set of possible outputs when the protocol is given inputs corresponding to $\mathcal{C}$.*

An important special case occurs when $\mathcal{C}$ is S^m, where $0 \leq m \leq n$. The complex $\mathcal{P}(S^m)$ is the set of output simplexes when the processes in $ids(S^m)$ start with their corresponding values in $vals(S^m)$.

The protocol complex satisfies some useful functorial properties, which follow immediately from the definitions. Let $\mathcal{C}_1, \ldots, \mathcal{C}_k$ be subcomplexes of $\mathcal{I}$.

Lemma 4.2 $\mathcal{P}(\bigcap_{i=0}^{k} \mathcal{C}_i) = \bigcap_{i=0}^{k} \mathcal{P}(\mathcal{C}_i)$.

Lemma 4.3 $\mathcal{P}(\bigcup_{i=0}^{k} \mathcal{C}_i) = \bigcup_{i=0}^{k} \mathcal{P}(\mathcal{C}_i)$.

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. We reformulate our main theorem to say that a protocol $\mathcal{P}$ solves a decision task $\langle \mathcal{I}, \mathcal{O}, \Delta \rangle$ if and only if there exists a simplicial, color-preserving *decision map*

$$\delta : \mathcal{P}(\mathcal{I}) \rightarrow \mathcal{O},$$

such that for every simplex $S^m \in \mathcal{I}$, and every simplex $T^m \in \mathcal{P}(S^m)$, where $0 \leq m \leq n$, $\delta(T^m) \in \Delta(S^m)$. 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. Though this might seem like a roundabout formulation, 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.

4.2 Our Proof Strategy

Figure 25 shows the protocol complex for a simple one-round wait-free normal-form protocol. The processes share a three-element atomic snapshot memory array with each entry initialized to 0. Each process P, Q, and R writes 1 to its entry in the array, scans the array's values, and halts. This complex has a simple inductive structure. The vertex at the top "corner" represents a solo execution by P: it writes 1, scans the array, and observes only its own value. The vertexes along the left-hand edge represent solo executions by P and Q together, as in Figure 13. The three vertexes

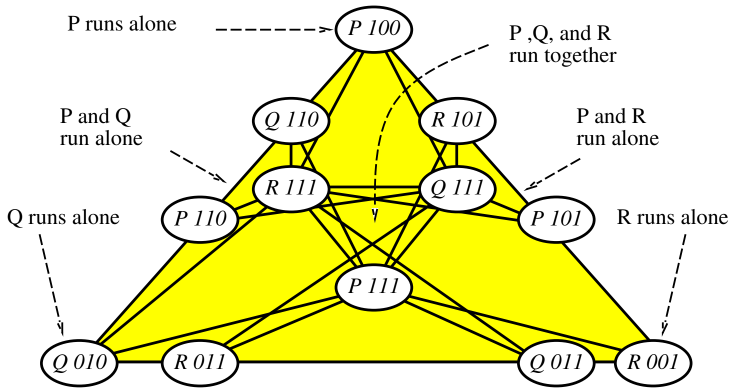

Figure 25: A one-round protocol complex

in the interior of the complex represent executions in which all processes' operations are interleaved: each process observes each of the others' values.

Our proof strategy is as follows. For each input simplex S^m, where $0 \leq m \leq n$, we identify a subdivision $\sigma(S^m)$ with a subcomplex of the protocol complex $\mathcal{P}(S^m)$, and construct the simplicial map μ in terms of the decision map δ. The construction is based on the following notion:

Definition 4.4 *A* span *for a protocol complex $\mathcal{P}(\mathcal{I})$ is a subdivision $\sigma(\mathcal{I})$ and a color-preserving simplicial map $\phi : \sigma(\mathcal{I}) \to \mathcal{P}(\mathcal{I})$, such that for every simplex $S \in \sigma(\mathcal{I})$,*

$$\phi(S) \in \mathcal{P}(carrier(S, \sigma(\mathcal{I}))). \tag{1}$$

A span σ is thus a subdivision of the input complex with the property that for each input simplex S^m, $0 \leq m \leq n$, the subdivision $\sigma(S^m)$ is mapped to a subcomplex of the protocol complex $\mathcal{P}(S^m)$. This construction is illustrated in Figure 26. The left-hand side shows a three-process input simplex (oval vertexes) which is subdivided (round vertexes) and mapped to a subcomplex of the protocol complex.

The "only if" direction of our main theorem will follow from showing that "if there is a protocol complex then there is a span." The required subdivision σ is the chromatic subdivision of $\mathcal{I}$ induced by the span, and the simplicial map $\mu(\vec{v})$ is just $\delta(\phi(\vec{v}))$, the composition of the span map and the task decision map.

We need to construct the span because the protocol complex itself is not necessarily a subdivision of the input complex (unlike the simple example presented in Figure 13). For example, the one-round three-process protocol

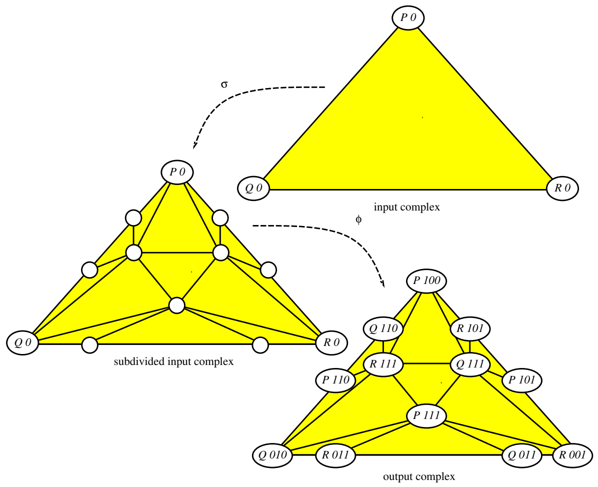

Figure 26: A span

complex of Figure 25 is a not subdivided 2-simplex, although it does contain the subdivided 2-simplex shown on the right-hand side of Figure 26.

We construct a span for a given protocol inductively by dimension, successively extending a span defined over the k-skeleton to the $(k+1)$-skeleton, as in Figure 31.

Here are the principal steps of our construction.

- We show that there are no topological "obstructions" to extending ϕ from the k-skeleton to the $(k+1)$-skeleton. Section 4.4 provides this first step, showing that for each input simplex S^m, $0 \leq m \leq n$, $\mathcal{P}(S^m)$ is m-connected.

- To maintain the color-preserving nature of σ and ϕ, we check that ϕ can be extended in a way that does not "collapse" simplexes, that is, it does not map higher-dimensional simplexes to lower-dimensional simplexes.

- The key to showing this "non-collapsing" property is the following *local* topological property: for every input simplex S^m, $0 \leq m \leq n$, the link of any k-simplex in $\mathcal{P}(S^m)$ is $(m-k-2)$-connected, a property we call *link-connectivity*. We address this issue in Section 4.5.

- We complete the proof in Section 4.6 by using connectivity and link-connectivity properties to show that any protocol has a span.

Both m-connectivity and link-connectivity are topological properties of complexes.

4.3 Basic Lemmas

We begin with some general lemmas about simplicial complexes.

Definition 4.5 *Complexes* $\mathcal{C}_0, \ldots, \mathcal{C}_q$ cover $\mathcal{C}$ *if* $\mathcal{C} = \cup_{i=0}^{q} \mathcal{C}_i$*. For any index set* U*, define* $\mathcal{C}_U = \cap_{i \in U} \mathcal{C}_i$*.*

Lemma 4.6 *If* $\mathcal{C}_0, \cdots, \mathcal{C}_q$ *cover* $\mathcal{C}$*, then for any index sets* U *and* V*,*

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

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

We will need the following inductive generalization of Theorem 2.38.

Lemma 4.7 *Let* $\mathcal{C}_0, \ldots, \mathcal{C}_q$ *cover* $\mathcal{C}$*, and* $k > 0$ *be such that for all* U*,* $\mathcal{C}_U$ *is* $(k - |U|)$*-connected. If* $U_0, \ldots, U_\ell$ *are index sets of a given size* u*,* $\ell + u \leq q + 1$*, such that for each distinct* i *and* j*,* $\{i\} = U_i - U_j$*, then*

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

Proof: If $k = u - 1$, then this lemma simply states that the union of non-empty complexes is non-empty. Let $k > u - 1$. We argue by induction on ℓ. The base case, when $\ell = 0$, is just the hypothesis. For the induction step, when $\ell > 0$, assume that every

$$\mathcal{A} = \bigcup_{i=0}^{\ell-1} \mathcal{C}_{U_i}$$

is $(k - u)$-connected. By the hypothesis,

$$\mathcal{B} = \mathcal{C}_{U_\ell}$$

is $(k - u)$-connected. Their intersection is

$$\mathcal{A} \cap \mathcal{B} = \left(\bigcup_{i=0}^{\ell-1} \mathcal{C}_{U_i} \right) \cap \mathcal{C}_{U_\ell} = \bigcup_{i=0}^{\ell-1} \mathcal{C}_{U_i \cup U_\ell}.$$

Let $V_i = U_i \cup U_\ell$, for $0 \leq i < \ell$. Notice that each $|V_i| = u+1$, and for each distinct i and j, $\{i\} = V_i - V_j$.

$$\mathcal{A} \cap \mathcal{B} = \bigcup_{i=0}^{\ell-1} \mathcal{C}_{V_i}.$$

The $\mathcal{C}_{V_i}$ satisfy the conditions of the induction hypothesis, so $\mathcal{A} \cap \mathcal{B}$ is $(k-u-1)$-connected, and the claim now follows from Theorem 2.38. ■

4.4 Connectivity

We now prove a remarkable property of wait-free read/write protocol complexes: for any input simplex S^m, $0 \leq m \leq n$, the protocol complex $\mathcal{P}(S^m)$ is m-connected. In other words, every protocol complex in this model has no "holes".

4.4.1 The Reachable Complex

We start with some definitions capturing the way in which the set of executions starting in any global state define a *reachable complex.*

Definition 4.8 *Let S^m be an input simplex, $0 \leq m \leq n$, and let s be a global state reached by executing $\mathcal{P}$ from the initial state given by S^m. A simplex R^m of $\mathcal{P}(S^m)$ is* reachable *from 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 s, written $\mathcal{R}(s)$, is the complex generated by reachable simplexes from s.*

Notice that the reachable complex from the initial state with participating processes and their inputs given by S^m is just $\mathcal{P}(S^m)$. For brevity, we say a state is reachable from an input simplex S^n if it is reachable from the initial state whose process ids and inputs are given by 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.

Definition 4.9 *For a pending process P_i in state s, define the reachable complex $\mathcal{R}_i(s)$ to be the subcomplex of the protocol complex that is reachable 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 4.6 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.

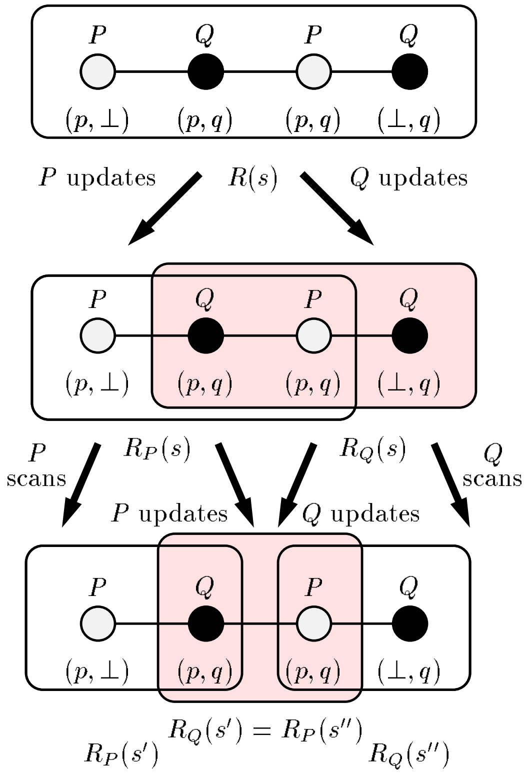

Figure 27: Connectivity of reachable complexes

4.4.2 Evolving Connectivity

We now give an informal example showing how the connectivity of the reachable complex evolves as an execution unfolds. Consider the one-round execution of Figure 13, illustrated in Figure 27.

The input complex consists of the single simplex $S^1 = \{\langle P, p\rangle, \langle Q, q\rangle\}$, and the protocol's initial state is s. The reachable complex $\mathcal{P}(S^1)$ encompasses three 1-simplexes. There are two pending operations in s: an update by P and an update by Q. If P goes first, the reachable complex $\mathcal{R}_P(s)$ encompasses the simplexes $\{\langle P, (p, \perp)\rangle, \langle Q, (p, q)\rangle\}$, $\{\langle P, (p, q)\rangle, \langle Q, (p, q)\rangle\}$, and their faces. The reachable complex $\mathcal{R}_Q(s)$ is defined symmetrically. Let

$U = \{P, Q\}$, the set of participating processes. Clearly, $\{\mathcal{R}_i(s) | i \in U\}$ cover $\mathcal{R}(s)$. $\mathcal{R}_U(s) = \mathcal{R}_P(s) \cap \mathcal{R}_Q(s)$ is the single simplex $\{\langle P, (p, q)\rangle, \langle Q, (p, q)\rangle\}$ reached in the execution in which both updates occurred before either scan (since updates commute, their order does not matter). $\mathcal{R}_U(s)$ is 1-connected. To show that $\mathcal{P}(S^1)$ is 1-connected, Theorem 2.38 implies that it is enough to show that both $\mathcal{R}_P(s)$ and $\mathcal{R}_Q(s)$ are 0-connected (connected in the graph-theoretic sense).

First, let us check that $\mathcal{R}_P(s)$ is connected (a symmetric argument holds for $\mathcal{R}_Q(s)$). Let s' be the global state if P updates $a[P]$ in s, and s'' the global state if Q updates $a[Q]$. There are two pending operations in s': Q's update changing $a[Q]$ to q, and P's scan. If P goes first in s', the reachable complex consists of the 1-simplex $\{\langle P, (p, \perp)\rangle, \langle Q, (p, q)\rangle\}$. If Q goes first, the reachable complex consists of $\{\langle P, (p, q)\rangle, \langle Q, (p, q)\rangle\}$. Their intersection $\mathcal{R}_U(s') = \mathcal{R}_P(s') \cap \mathcal{R}_Q(s')$ cannot contain vertexes of P since in $\mathcal{R}_Q(s')$, Q's update of $a[Q]$ to q must be reflected in the view returned by P's scan. Thus, this intersection is non-empty, containing the 0-simplex $\langle Q, (p, q)\rangle\}$ of Q, which is 0-connected. The 1-connectivity of $\mathcal{R}_P(s)$ follows from Theorem 2.38, since each outcome 1-simplex is connected and their intersection is 0-connected.

4.4.3 Proof of Connectivity

We now present the complete proof. Instead of exhaustively considering all executions, as in the example above, we concentrate on a specific "critical" state and argue by contradiction.

Definition 4.10 *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's execution to a state where $\wp$ henceforth holds.*

Lemma 4.11 *If $\wp$ is a property that does not hold in some state s but does hold in the final state of every execution, 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, repeatedly pick a non-critical pending process and run it until it is no longer non-critical. Because the protocol must eventually terminate in a state where $\wp$ holds, advancing non-critical processes in this way will eventually leave the protocol in a state where $\wp$ does not hold, but all processes are either decided or about to make $\wp$ henceforth *true*. This state is the desired critical state. ∎

We will now show that in any state reachable from any input simplex S^n, $\mathcal{P}(S^n)$ satisfies the conditions of Lemma 4.7. Informally, our proof strategy proceeds by contradiction. Assume the claim is initially false. Since the reachable complex eventually shrinks to a single simplex, it eventually satisfies the desired properties, so by Lemma 4.11 we can 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. A similar strategy was used by Fischer, Lynch, and Paterson to prove the impossibility of asynchronous consensus [19].

Lemma 4.12 *For any input simplex S^n, $\mathcal{P}(S^n)$ is n-connected.*

Proof: By way of contradiction, let $\mathcal{P}$ be an $(n+1)$-process protocol for which the claim is false. Pick $\mathcal{P}$ so that n is minimal. Let $\wp$ be the property "$\mathcal{R}(s)$ is n-connected". If s is any final state of $\mathcal{P}$, then $\mathcal{R}(s)$ is a single simplex, which is n-connected (Lemma 2.39) Because $\wp$ holds in every final state, Lemma 4.11 implies that $\wp$ has a critical state s.

We claim that for every pending set U, $\mathcal{R}_U(s)$ is $(n-|U|+1)$-connected. We proceed by a case analysis. Since $\wp$ is *true* in any final state, we can restrict our attention to non-empty pending sets.

Suppose U consists entirely of scans. In every execution leading to a simplex in $\mathcal{R}_U(s)$, each pending scan is ordered before any update. Because scans commute, each such execution is equivalent to one in which all processes in U perform their scans before any other operation occurs. If s' is the state reached from s by executing all pending scans in U, then $\mathcal{R}(s') = \mathcal{R}_U(s)$. Because s is critical, $\mathcal{R}(s') = \mathcal{R}_U(s)$ is n-connected. Because $|U| > 0$, $\mathcal{R}_U(s)$ is $(n-|U|+1)$-connected.

Suppose U consists entirely of updates. Recall that in normal form protocols, processes update an atomic snapshot memory a, where each new value is distinct from any earlier value. In every execution leading to a simplex in $\mathcal{R}_U(s)$, each pending update must be ordered before any scan. Because updates commute, each such execution is equivalent to one in which all processes in U perform their updates before any other operation occurs. If s' is the state reached from s by executing all pending updates in U, then $\mathcal{R}(s') = \mathcal{R}_U(s)$. Because s is critical, $\mathcal{R}(s') = \mathcal{R}_U(s)$ is n-connected. Because $|U| > 0$, $\mathcal{R}_U(s)$ is $(n-|U|+1)$-connected.

Finally, suppose both scans and updates appear in U. Let $U = R \cup W$, where R (respectively W) is the set of processes with pending scans (updates). Suppose $P_i \in R$ is about to scan, and $P_j \in W$ is about to update

$a[j]$ from v to v'. In every simplex in $\mathcal{R}_i(s)$, P_i's scan returns v, while in every simplex in $\mathcal{R}_j(s)$, it returns v'. As a result, P_i has no vertexes in $\mathcal{R}_i(s) \cap \mathcal{R}_j(s)$. More generally, $\mathcal{R}_U(s)$ contains no vertex of any process in R. In every execution leading to a simplex in $\mathcal{R}_U(s)$, each update in W is ordered before any scan by a process in $ids(\mathcal{R}_U(s))$. Conversely, any execution from s by processes not in R in which all updates in W precede any other operation is in $\mathcal{R}_U(s)$. Let s' be the state reached from s by executing all pending updates in W. Since updates commute, their order is unimportant.

Let $\mathcal{P}'$ be the $(n - |R| + 1)$-process protocol with initial state s' identical to $\mathcal{P}$ except that the processes in R do not participate. Let $\mathcal{R}'(s')$ be the reachable complex for $\mathcal{P}'$ from s'. We have just argued that $\mathcal{R}_U(s) = \mathcal{P}'(s')$. Because $|R| > 0$, and $\mathcal{P}$ was chosen to be minimal, $\mathcal{R}'(s)$ is $(n - |R| + 1)$-connected, and because $|U| > |R|$, it is also $(n - |U| + 1)$-connected.

In all cases, we have shown that $\mathcal{R}_U(s)$ is $(n - |U| + 1)$-connected. By Lemma 4.7, $\mathcal{R}(s) = \cup \mathcal{R}_i(s)$ is n-connected. It follows that $\wp$ holds in s, contradicting our assumption that s is a critical state for $\wp$. We have shown that $\mathcal{R}(s)$ is n-connected for every state s. If s is the initial state given by S^n, then $\mathcal{R}(s) = \mathcal{P}(S^n)$ is n-connected for every input simplex S^n. ∎

For any input simplex S^m, $0 \leq m \leq n$, $\mathcal{P}(S^m)$ can be considered the protocol complex for an $(m+1)$-process protocol.

Corollary 4.13 *For any input simplex S^m, $0 \leq m \leq n$, $\mathcal{P}(S^m)$ is m-connected.*

Note that for any input n-complex $\mathcal{I}$, $\mathcal{P}(\mathcal{I})$ is not necessarily n-connected, since $\mathcal{I}$ itself may not be n-connected.

4.5 Link Connectivity

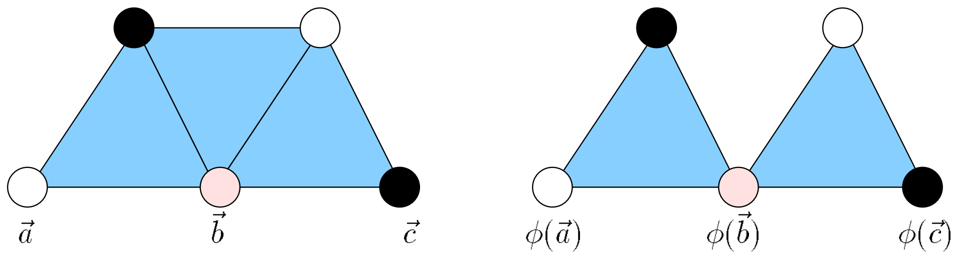

Figure 28: Complexes $\mathcal{A}$ (left) and $\mathcal{B}$ (right)

Given Corollary 4.13, it is not hard to construct a subdivision $\sigma(\mathcal{I})$ and a simplicial map $\mu : \sigma(\mathcal{I}) \rightarrow \mathcal{P}(\mathcal{I})$ satisfying Equation 1. This construction suffices to show this paper's principal impossibility results (such as the impossibility of k-set agreement), but it is not yet enough to construct an algorithm.

The missing property is that μ must be *color preserving*: for every vertex $\vec{v}$, $id(\mu(\vec{v}))$ must equal $id(\vec{v})$. This property cannot be taken for granted: the existence of a simplicial map does not always guarantee the existence of a color-preserving simplicial map. Consider the following simple example. Let $\mathcal{A}$ and $\mathcal{B}$ be the colored complexes shown in Figure 28, and ϕ the simplicial map carrying vertexes $\vec{a}$, $\vec{b}$ and $\vec{c}$ to $\phi(\vec{a})$, $\phi(\vec{b})$, and $\phi(\vec{c})$, respectively. Does there exist a chromatic subdivision σ of $\mathcal{A}$ and a simplicial map $\psi : \sigma(\mathcal{A}) \rightarrow \mathcal{B}$ extending ϕ? It is not difficult to construct a chromatic subdivision σ and a *non* color-preserving simplicial map satisfying these conditions, but it turns out that no color-preserving map is possible. For any subdivision σ, one can show that $lk(\vec{b}, \sigma(\mathcal{A}))$ must be connected. By contrast, $lk(\phi(\vec{b}), \mathcal{B})$ is not connected. Some vertex $\vec{x}$ in $lk(\vec{b}, \sigma(\mathcal{A}))$ must map to $\phi(\vec{a})$, and some $\vec{y}$ to $\phi(\vec{c})$, and the path between $\vec{x}$ and $\vec{y}$ in $lk(\vec{b}, \sigma(\mathcal{A}))$ must map to a path linking the disconnected components of $lk(\phi(\vec{b}), \mathcal{B})$, a contradiction.

To ensure that μ is color preserving, we must prove one more property of each protocol complex $\mathcal{P}(S^m)$, $0 \leq m \leq n$. In addition to being m-connected (Corollary 4.13), $\mathcal{P}(S^m)$ is also *link-connected*.

Definition 4.14 *A p-complex $\mathcal{C}$ is* link-connected *if for all simplexes $T^q \in \mathcal{C}$, $0 \leq q \leq p$, $lk(T^q, \mathcal{C})$ is $(p-q-2)$-connected.*

A p-complex can be p-connected without being link-connected, and vice-versa.

Lemma 4.15 *If $\mathcal{C}$ is link-connected, so is $lk(T, \mathcal{C})$ for any simplex $T \in \mathcal{C}$.*

Proof: For every simplex S of $\mathcal{C}$ and T of $lk(S, \mathcal{C})$,

$$lk(T, lk(S, \mathcal{C})) = lk(T \cdot S, \mathcal{C}).$$

■

Lemma 4.16 *For every input simplex S^n, and simplex $T^m \in \mathcal{P}(S^n)$, $lk(T^m, \mathcal{P}(S^n))$ is $(n-m-2)$-connected.*

Proof: The proof resembles the proof of Lemma 4.12.

By way of contradiction, let $\mathcal{P}$ be an $(n+1)$-process protocol for which the claim is false. Pick $\mathcal{P}$ so that n is minimal. For a global state s, let $\mathcal{R}(s)$ be the reachable complex from s, and $\mathcal{Q}(s) = lk(T^m, \mathcal{R}(s))$ (empty if T^m is not in $\mathcal{R}(s)$). Let $\wp$ be the property

$$T^m \in \mathcal{R}(s) \Rightarrow \mathcal{Q}(s) \text{ is } (n-m-2)\text{-connected.}$$

Initially, $\wp$ is false by assumption. In every final state s, either T^m is not in $\mathcal{R}(s)$, or $\mathcal{Q}(s)$ is a single $(n-m-2)$-simplex (which is $(n-m-2)$-connected). Either way, $\wp$ holds in every final state. By Lemma 4.11, $\wp$ has a critical state s. Notice that because $\wp$ is *false* in s, T^m is in $\mathcal{R}(s)$.

As usual, for each P_i pending in s, $\mathcal{R}_i(s)$ is the reachable complex after P_i executes its operation, and for each set U of pending processes in s, $\mathcal{R}_U(s) = \cap_{i \in U} \mathcal{R}_i(s)$. Define $\mathcal{Q}_i(s) = lk(T^m, \mathcal{R}_i(s))$, and $\mathcal{Q}_U(s) = lk(T^m, \mathcal{R}_U(s))$, The $\mathcal{Q}_i(s)$ cover $\mathcal{Q}(s)$, and Lemma 4.6 applies.

Define a pending operation to be *preserving* if it leaves T^m within the reachable complex. There must be a preserving operation pending in s, because otherwise T^m would not be reachable, and $\wp$ would be *true*. Let U be any non-empty set of pending preserving operations in s. For each P_i in U, that process's pending operation is preserving, so T^m is in each $\mathcal{R}_i(s)$, and therefore in $\mathcal{R}_U(s)$. We now show, by case analysis, that any $\mathcal{Q}_U(s)$ is $(n-m-|U|-1)$-connected.

Suppose U consists entirely of scans. In every execution leading to a simplex in $\mathcal{R}_U(s)$, each pending scan is ordered before any update. Because scans commute, each such execution is equivalent to one in which all processes in U perform their scans before any other operation occurs. If s' is the state reached from s by executing all pending scans in U, then $\mathcal{R}(s') = \mathcal{R}_U(s)$ and $\mathcal{Q}(s') = \mathcal{Q}_U(s)$. Because s is critical, $\wp$ holds in s'. Since T^m is in $\mathcal{R}_U(s) = \mathcal{R}(s')$, $\mathcal{Q}(s') = \mathcal{Q}_U(s)$ is $(n-m-2)$-connected. Because $|U| > 0$, $\mathcal{Q}_U(s)$ is $(n-m-|U|-1)$-connected.

Suppose U consists entirely of updates. Recall that in normal form protocols, processes update an atomic snapshot memory a, where each new value is distinct from any earlier value. In every execution leading to a simplex in $\mathcal{R}_U(s)$, each pending update must be ordered before any scan. Because updates commute, each such execution is equivalent to one in which all processes in U perform their updates before any other operation occurs. If s' is the state reached from s by executing all pending updates in U, then $\mathcal{R}(s') = \mathcal{R}_U(s)$ and $\mathcal{Q}_U(s) = \mathcal{Q}(s')$. Because s is critical, $\mathcal{Q}(s') = \mathcal{Q}_U(s)$ is $(n-m-2)$-connected, and because $|U| > 0$, $\mathcal{Q}_U(s)$ is $(n-m-|U|-1)$-connected.

Finally, suppose both scans and updates appear in U. Let $U = R \cup W$, where R (W) is the set of processes with pending preserving scans (updates). Suppose $P_i \in R$ is about to scan, and $P_j \in W$ is about to update $a[j]$ from v to v'. In every simplex in $\mathcal{R}_i(s)$, P_i's scan returns v, while in every simplex in $\mathcal{R}_j(s)$, it returns v'. As a result, P_i has no vertexes in $\mathcal{R}_i(s) \cap \mathcal{R}_j(s)$. More generally, $\mathcal{R}_U(s)$ contains no vertex of any process in R. In every execution leading to a simplex in $\mathcal{R}_U(s)$, each update in W is ordered before any scan by a process in $ids(\mathcal{R}_U(s))$. Conversely, any execution from s by processes not in R in which all updates in W precede any other operation is in $\mathcal{R}_U(s)$. Let s' be the state reached from s by executing all pending updates in W. Since updates commute, their order is unimportant.

Let $\mathcal{P}'$ be the $(n - |R| + 1)$-process protocol with initial state s' identical to $\mathcal{P}$ except that the processes in R do not participate. Let $\mathcal{R}'(s')$ be the reachable complex for $\mathcal{P}'$ from s', and $\mathcal{Q}'(s') = lk(T^m, \mathcal{R}'(s'))$. We have just argued that $\mathcal{R}_U(s) = \mathcal{P}'(s')$, and $\mathcal{Q}_U(s) = \mathcal{Q}'(s')$. Because $|R| > 0$, and $\mathcal{P}$ was chosen to be minimal, $\mathcal{Q}'(s') = \mathcal{Q}_U(s)$ is $(n - m - |R| - 1)$-connected. Because $|U| > |R|$, $\mathcal{Q}_U(s)$ is also $(n - m - |U| - 1)$-connected.

In all cases, we have shown that $\mathcal{Q}_U(s)$ is $(n - m - |U| - 1)$-connected. By Lemma 4.7, $\mathcal{Q}(s) = \cup \mathcal{Q}_i(s)$ is $(n - m - 2)$-connected. It follows that $\wp$ holds in s, contradicting our assumption that s is a critical state for $\wp$. We have shown that $\wp$ must hold in every state, and that every $\mathcal{Q}(s)$ is $(n - m - 2)$-connected. In particular, for every input simplex S^n, $\mathcal{P}(S^n)$ is link-connected. ∎

For any input simplex S^m, $0 \leq m \leq n$, $\mathcal{P}(S^m)$ can be considered the protocol complex for an $(m+1)$-process protocol.

Corollary 4.17 *For any input simplex S^m, $0 \leq m \leq n$, $\mathcal{P}(S^m)$ is link-connected.*

4.6 Every Protocol Has A Span

For our inductive construction, we will need to show that any color and carrier-preserving map from a subdivision of the i-skeleton of $\mathcal{I}$ to $\mathcal{P}(\mathcal{I})$ can be extended up one dimension, to a color and carrier-preserving map of the $(i+1)$-skeleton.

Recall that a simplicial map *collapses* a simplex of non-zero dimension if it maps that simplex to a vertex. A map is *non-collapsing* if it collapses no simplexes. Clearly, color-preserving maps are non-collapsing. Conversely, if a color-preserving map $\sigma(skel^i(\mathcal{A})) \rightarrow \mathcal{B}$ has a non-collapsing extension $\sigma(skel^{i+1}(\mathcal{A})) \rightarrow \mathcal{B}$, then that extension is also color-preserving.

Consequently, we focus on the circumstances under which maps have non-collapsing extensions.

The following lemma appears in Glaser [21, Theorem IV.2].

Lemma 4.18 *Let $\mathcal{A}$, $\mathcal{B}$, and $\mathcal{C}$ be complexes such that $\mathcal{A} \subset \mathcal{B}$, and $f : |\mathcal{B}| \rightarrow |\mathcal{C}|$ a continuous map such that the vertex map induced by 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 $\phi : \tau(\mathcal{B}) \rightarrow \mathcal{C}$ extending the restriction of f to $|\mathcal{A}|$.*

Definition 4.19 *Let σ be a subdivision of $skel^{p-1}(\mathcal{C})$, for some pure complex $\mathcal{C}$. The subdivision of $skel^{p}(S^p)$ obtained by* starring *σ [37, p. 85] is defined as follows. Let $S_0^p, \ldots, S_L^p$ be the p-simplexes of $skel^p(\mathcal{C})$. For $0 \leq i \leq L$, let $\vec{w}_i$ be the barycenter of $|S_i^p|$. Each $\vec{w}_i \cdot \sigma(skel^{p-1}(S_i^p))$ is a subdivision of S_i^p, and the union of these complexes as i ranges from 0 to L is a subdivision of $skel^p(\mathcal{C})$ that agrees with σ on the $(p-1)$-skeleton*

If σ is the trivial subdivision, we can apply this construction to the boundary complex of a single simplex S^p in $\mathcal{C}$, in which case we speak of starring S^p in $\mathcal{C}$.

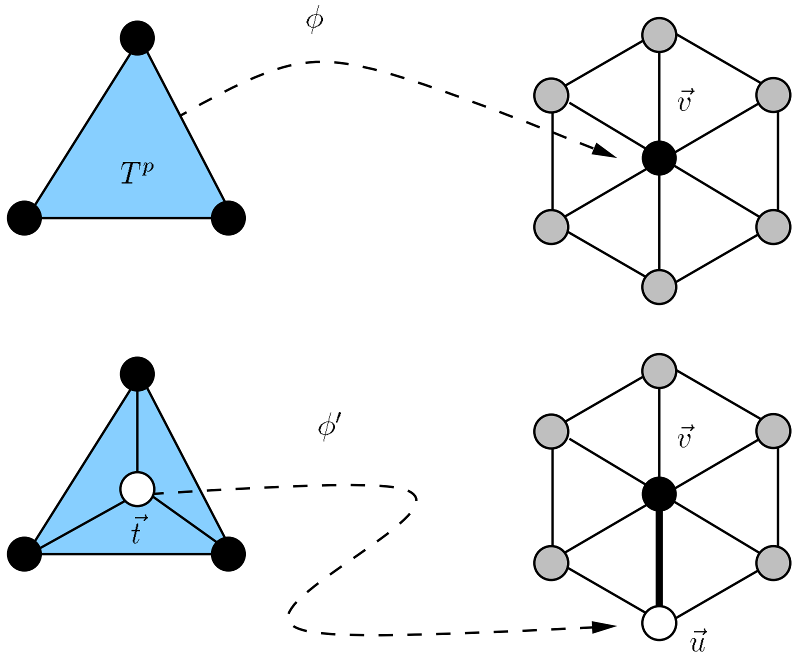

Figure 29: Eliminating simplex collapse: top dimension

Lemma 4.20 *Let $\mathcal{A}$ be a $(p-1)$-sphere, $\mathcal{B}$ a p-disk having $\mathcal{A}$ as boundary, and $\mathcal{C}$ a complex that is $(p-1)$-connected and link-connected. If $\eta : \mathcal{A} \to \mathcal{C}$ is a simplicial map, then*

1. *there exists a subdivision τ of $\mathcal{B}$ such that $\tau(\mathcal{A}) = \mathcal{A}$,*
2. *a simplicial map $\phi : \tau(\mathcal{B}) \to \mathcal{C}$ that agrees with η on $\mathcal{A}$, and*
3. *ϕ collapses no internal simplexes of $\tau(\mathcal{B})$.*

Proof: Because $\mathcal{C}$ is $(p-1)$-connected, the continuous map $|\eta|$ on $\mathcal{A}$ can be extended to a continuous $f : |\mathcal{B}| \to |\mathcal{C}|$ whose restriction to $\mathcal{A}$ is simplicial. Conditions 1 and 2 follow immediately from Lemma 4.18.

If Condition 3 does not hold, choose τ and ϕ to minimize (1) the dimension of $\mathcal{C}$, (2) the largest dimension of any collapsed simplex, and (3) the number of collapsing simplexes of that dimension. We will demonstrate a contradiction by "adjusting" τ and ϕ to collapse one fewer simplex of maximal dimension. To adjust ϕ, we subdivide the collapsed simplex T by inserting a new vertex at the barycenter, and then extending ϕ to send that new vertex to an vertex adjacent to $\phi(T)$, resulting in a new subdivision and map that collapses one fewer simplex of maximal dimension.

Suppose ϕ collapses a simplex T^p, where p is the maximal dimension of any simplex in $\mathcal{B}$. As illustrated in Figure 29, where T^p is a triangle, starring T^p yields a new subdivision $\tau'(\mathcal{B})$ Let $\phi(T^p) = \vec{v}$, and let $\vec{t}$ be the barycenter of T^p. If $T_0^{p-1}, \ldots, T_p^{p-1}$ are the $(p-1)$-faces of T^p, then starring T^p (Definition 4.19) yields a new subdivision $\tau'(\mathcal{B})$. Pick $\vec{u}$ such that $(\vec{v}, \vec{u})$ is a 1-simplex in $\mathcal{C}$. Define $\phi' : \tau'(\mathcal{B}) \to \mathcal{C}$ such that $\phi'(\vec{t}) = \vec{u}$, and elsewhere $\phi' = \phi$. We have constructed a subdivision τ' and simplicial map ϕ' satisfying Conditions 1 and 2, but collapsing one fewer p-simplex, a contradiction.

Suppose ϕ collapses an internal simplex $T^q \in \tau(\mathcal{B})$ to $\vec{v}$, where $1 \leq q < p$, but collapses no internal simplexes of higher dimension. Our approach is illustrated in Figure 30, where T^q is an edge. Define the subdivision $\tau'(T^q)$ by starring T^q in $\tau(\mathcal{B})$.

Because $q < p$, then $lk(T^q, \mathcal{B})$ is non-empty. The vertexes of any $L^{p-q-1} \in lk(T^q, \mathcal{B})$ are affinely independent of the vertexes of T^q, so $\vec{t}$ is affinely independent of each L^{p-q-1}. Moreover, each $\vec{t} \cdot L^{p-q-1}$ is a simplex, and $\vec{t} \cdot lk(T^q, \mathcal{B})$ is a complex. Because $\mathcal{B}$ is a manifold with boundary, Lemma 2.36 implies that $lk(T^q, \mathcal{B})$ is a $(p-q-1)$-sphere, and hence $\vec{t} \cdot lk(T^q, \mathcal{B})$ is a $(p-q)$-disk. Because ϕ does not collapse any $(q+1)$-simplexes, ϕ does not send any vertex of $lk(T^q, \mathcal{B})$ to $\vec{v}$, so $\phi : lk(T^q, \mathcal{B}) \to lk(\vec{v}, \mathcal{C})$.

We have a simplicial map ϕ carrying the $(p-q-1)$-sphere $lk(T^q, \mathcal{B})$ to $lk(\vec{v}, \mathcal{C})$, which is $(p-2)$-connected by Lemma 4.15. Recall that the dimension of $\mathcal{C}$ is the smallest for which Condition 3 fails, and $\dim(lk(\vec{v}, \mathcal{C})) < \dim(\mathcal{C})$, so all three conditions are satisfied: (1) there is a subdivision ρ of $\vec{t} \cdot lk(T^q, \mathcal{B})$, (2) a simplicial map $\psi : \rho(\vec{t} \cdot lk(T^q, \mathcal{B})) \to lk(\vec{v}, \mathcal{C})$ that agrees with ϕ on $lk(T^q, \mathcal{B})$, and (3) ψ collapses no internal simplexes of $\rho(\vec{t} \cdot lk(T^q, \mathcal{B}))$.

The complex $\rho(\vec{t} \cdot lk(T^q, \mathcal{B})) \cdot \dot{T}^{q-1}$ is a subdivision of $st(T^q, \mathcal{B})$ that leaves its boundary unchanged. Replacing $st(T^q, \mathcal{B})$ in $\tau(\mathcal{B})$ by this subdivision yields a subdivision $\tau'(\mathcal{B})$. Define ϕ' to agree with ψ on $\rho(\vec{t} \cdot lk(T^q, \mathcal{B}))$, and with ϕ elsewhere. Condition 2 ensures that ϕ and ψ agree on $lk(T^q, \mathcal{B})$, so this map is well-defined. The complex $\tau'(\mathcal{B})$ and map ϕ' satisfy Conditions 1 and 2, but the map collapses one fewer q-simplex, a contradiction. ∎

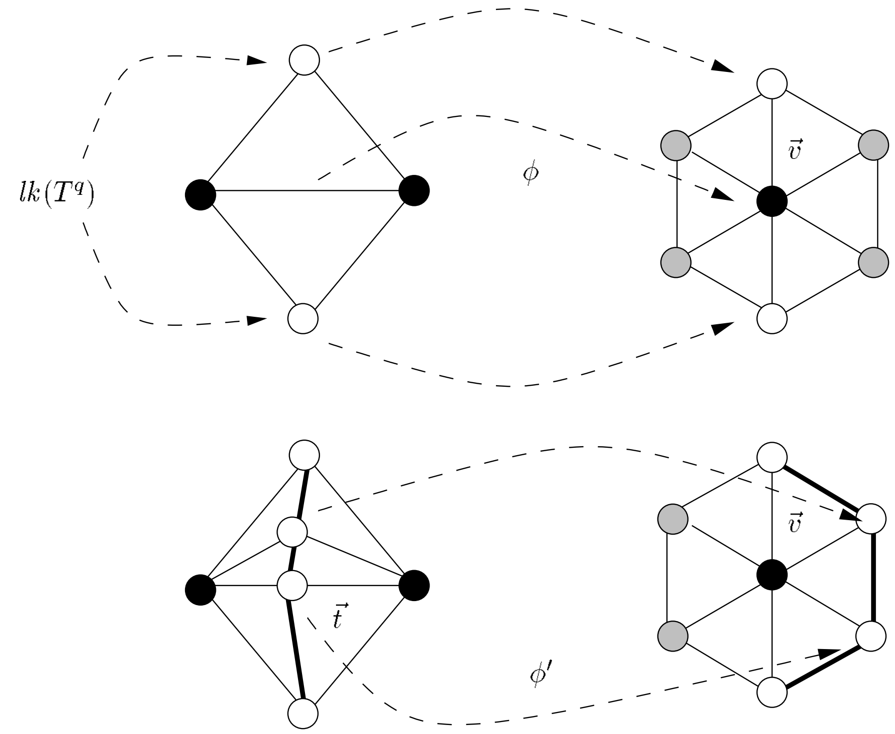

Figure 30: Eliminating simplex collapse: intermediate dimensions

Lemma 4.21 *Let S^p be a p-simplex, $\dot{S}^{p-1}$ its boundary complex, and σ a chromatic subdivision of $\dot{S}^{p-1}$. Let $\mathcal{C}$ be a p-colored complex that is $(p-1)$-*

connected and link-connected, and

$$\phi : \sigma(\dot{\mathcal{S}}^{p-1}) \to \mathcal{C}$$

a color-preserving simplicial map. There exist a chromatic subdivision $\hat{\sigma}$ of S^p, and a color-preserving simplicial map

$$\hat{\phi} : \hat{\sigma}(S^p) \to \mathcal{C}$$

such that $\hat{\sigma}$ agrees with σ on $\dot{\mathcal{S}}^{p-1}$, and $\hat{\phi}$ agrees with ϕ on $\sigma(\dot{\mathcal{S}}^{p-1})$.

Proof: The complex $\sigma(\dot{\mathcal{S}}^{p-1})$ is a $(p-1)$-sphere, and the subdivision of S^p constructed by starring $\sigma(\dot{\mathcal{S}}^{p-1})$ is a p-disk having $\sigma(\dot{\mathcal{S}}^{p-1})$ as boundary. By Lemma 4.20, ϕ and σ can be extended to $\hat{\phi}$ and $\hat{\sigma}$ such that $\hat{\sigma}$ agrees with σ on $\dot{\mathcal{S}}^{p-1}$, and $\hat{\phi}$ is a simplicial map that agrees with ϕ on $\sigma(\dot{\mathcal{S}}^{p-1})$, and collapses no internal simplexes of $\hat{\sigma}(S^p)$.

It remains to check that $\hat{\phi}$ is color-preserving. Say that T^p is a *boundary simplex* if it can be expressed as $\vec{t} \cdot T^{p-1}$, where T^{p-1} is in $\sigma(\dot{\mathcal{S}}^{p-1})$. Because ϕ is chromatic on $\sigma(\dot{\mathcal{S}}^{p-1})$, it carries the face T^{p-1} to a simplex labeled with $(p-1)$ out of the p colors labeling $\mathcal{C}$. Because ϕ does not collapse T^p, it must carry $\vec{t}$ to a vertex labeled with the only remaining color. Therefore, $\hat{\phi}$ is color-preserving on the boundary complex T^p.

Because $\hat{\sigma}(S^p)$ is a p-manifold with boundary, for every simplex T^p in $\hat{\sigma}(S^p)$, there is a sequence of simplexes $S_0^p, \ldots, S_\ell^p$ such that S_0^p is a boundary simplex, $S_\ell^p = T^p$, and $S_i^p \cap S_{i+1}^p$ is an $(n-1)$-simplex. The claim now follows by inductively extending the same argument along the sequence. ∎

We now have the tools needed to construct a span.

Lemma 4.22 *Every wait-free protocol complex has a span.*

Proof: We build up σ and ϕ by induction on the skeleton of $\mathcal{I}$ as depicted in Figure 31. For each $\vec{v} \in \mathcal{I}$, 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 is color-preserving, and satisfies Equation 1 of Definition 4.4.

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

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

satisfying Equation 1. Let τ be the subdivision of $skel^k(\mathcal{I})$ obtained by starring σ_{k-1}. Let $S_0^k, \ldots, S_K^k$ be all the k-simplexes in $skel^k(\mathcal{I})$. For each

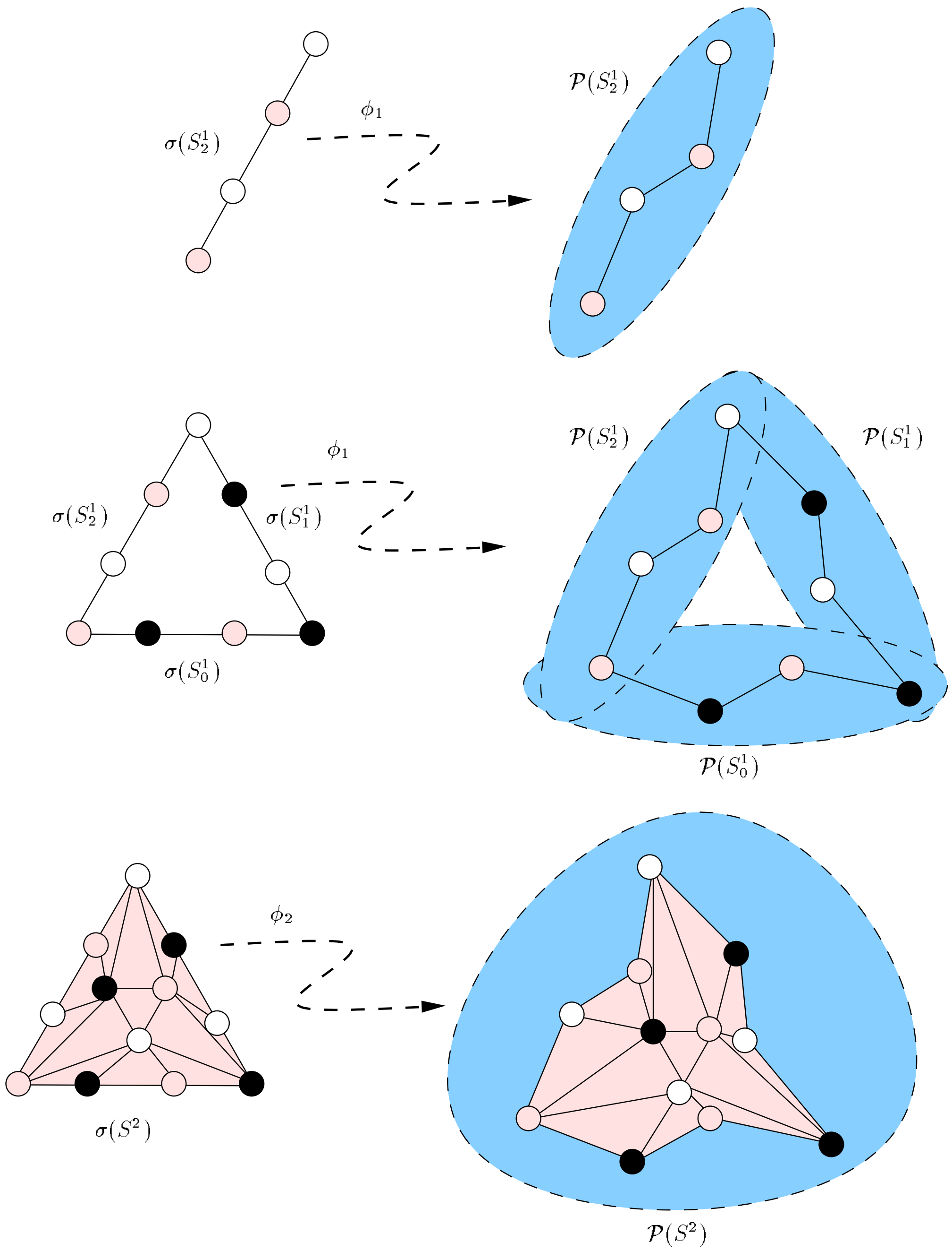

Figure 31: Inductive span construction

S_i^k, $0 \leq i \leq K$, let $\mathcal{A}_i$ be the complex $\tau(skel^{k-1}(S_i^k))$, and $\mathcal{B}_i$ the complex $\tau(S_i^k)$, and $\mathcal{C}_i$ the complex $\mathcal{P}(S_i^k)$. By Lemma 4.21, there exists a chromatic subdivision τ_i of $\mathcal{B}_i$ such that $\tau_i(\mathcal{A}_i) = \mathcal{A}_i$, a color-preserving simplicial map $\psi_i : \tau(\mathcal{B}_i) \rightarrow \mathcal{C}_i$ that agrees with ϕ_{k-1} on the $\mathcal{A}_i$. Because the τ_i agree on $skel^{k-1}(\mathcal{I})$, they induce a subdivision $\sigma_k(skel^k(\mathcal{I}))$. Because the ψ_i also agree on $\sigma_{k-1}(skel^{k-1}(\mathcal{I}))$, they induce a color-preserving simplicial map $\phi_k : \sigma_k(skel^k(\mathcal{I})) \rightarrow \mathcal{P}(\mathcal{I})$. Finally, it is immediate from the construction that σ_k and ϕ_k satisfy Equation 1.

The desired subdivision σ is σ_n, and the desired map ϕ is ϕ_n. (Notice that $skel^n(\mathcal{I}) = \mathcal{I}$.) ∎

Theorem 4.23 *If a decision task $\langle \mathcal{I}, \mathcal{O}, \Delta \rangle$ has a wait-free read/write protocol, then there exists a chromatic subdivision $\sigma(\mathcal{I})$ and a color-preserving simplicial map $\mu : \sigma(\mathcal{I}) \rightarrow \mathcal{O}$ such that for each simplex S^m in $\sigma(\mathcal{I})$, $\mu(S) \in \Delta(carrier(S, \mathcal{I}))$.*

Proof: Suppose a protocol exists. By Lemma 4.22, the protocol has a span. Let σ be the chromatic subdivision of $\mathcal{I}$ induced by the span, and let $\mu(S) = \delta(\phi(S))$, the composition of the span map and the decision map. ∎

5 Sufficiency

In this section we show how to construct an algorithm for any task satisfying the conditions of the asynchronous computability theorem. We now give an overview of this construction.

We introduce the *standard chromatic subdivision* of a complex $\mathcal{I}$, denoted $\chi(\mathcal{I})$, the chromatic analogue of the classical barycentric subdivision. This subdivision is illustrated in Figure 33. We also introduce the *iterated* standard chromatic subdivision $\chi^K(\mathcal{I})$, and the notion of a chromatic subdivision holding a subcomplex fixed.

We will show that we can assume without loss of generality, that the span subdivision $\sigma(\mathcal{I})$ has the form $\chi^K(\mathcal{I})$, for some sufficiently large K. To solve the task, each process P_i begins by placing a token on an input vertex $\vec{s}_i$. As vertexes of $\mathcal{I}$, they are "close", since they span a simplex S. As vertexes of $\chi^K(\mathcal{I})$, however, they are "far apart", since they lie on the boundary of the subcomplex $\chi^K(S)$. The key insight is that we can construct a protocol for this task by reduction to a variation on *approximate agreement* [4, 18], in which the processes start out at the vertexes of S, and after a process of negotiation eventually converge to the vertexes of a single simplex in $\chi^K(S)$.

Once P_i has converged on a vertex $\vec{t_i}$ of matching color, it solves the original task by choosing $\mu(\vec{t_i})$.

We call this process *simplex agreement*. Simplex agreement on $\chi(\mathcal{I})$ is solved by the elegant "participating set" protocol of Borowsky and Gafni [12]. Simplex agreement on $\chi^K(\mathcal{I})$ is solved by iterating that protocol K times.

Our construction relies on the following theorem, proved below.

> Let S^n be a colored n-simplex, χ the standard chromatic subdivision, and σ an arbitrary chromatic subdivision. For sufficiently large K, there exists a color and carrier-preserving simplicial map
> $$\phi : \chi^K(S^n) \to \sigma(S^n).$$

This theorem implies that any algorithm for simplex agreement on $\chi^K(S^n)$ yields an algorithm for simplex agreement on an arbitrary $\sigma(S^n)$. Note that this theorem is expressed entirely in terms of combinatorial topology.

5.1 Proof Strategy

To establish the intuition underlying our proof, we give an informal outline of a proof that for sufficiently large K, there exists a carrier-preserving (but not necessarily color-preserving) simplicial map $\phi : \chi^K(S^n) \to \sigma(S^n)$. This claim is a special case of the well-known finite simplicial approximation theorem [37, p.89].

Here are some useful notions from classical point-set topology.

Definition 5.1 *An* open cover *for a geometric complex $\mathcal{C}$ is a finite collection of open sets $U_0, \ldots, U_k$ such that $\mathcal{C} \subseteq \cup_{i=0}^{k} U_i$.*

The following result is standard [37, p.89].

Lemma 5.2 *Let $\mathcal{C}$ be a complex*[6]*, and $U_0, \ldots, U_k$ an open cover for $\mathcal{C}$. There exists a $\lambda > 0$ (called a* Lebesgue number*) such that any set of diameter less than λ lies in a single U_i.*

The open stars around vertexes of $\sigma(S^n)$ form an open cover for $\sigma(S^n)$ with Lebesgue number λ. By choosing K sufficiently large, we can ensure that the diameter of any vertex's star in $\chi^K(S^n)$ is less than λ. It follows that for each vertex $\vec{x}$ in $\chi^K(S^n)$, there is an $\vec{s}$ in $\sigma(S^n)$ such that

$$st(\vec{x}, \chi^K(S^n)) \subset st^\circ(\vec{s}, \sigma(S^n)). \tag{2}$$

[6] In fact, $\mathcal{C}$ could be any compact space.

Let $\phi(\vec{x}) = \vec{s}$. It is easily shown that ϕ is the desired carrier-preserving simplicial map. Unfortunately, ϕ is not necessarily color-preserving: $id(\vec{x})$ may not equal $id(\phi(\vec{x}))$.

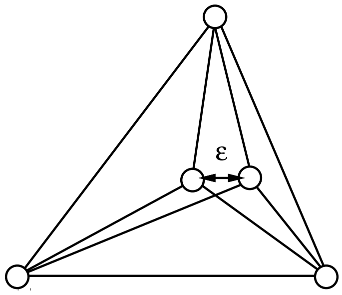

Figure 32: An ϵ-perturbation

Our goal is to extend Equation 2 to require that $\vec{x}$ and $\vec{s}$ have matching colors: for each vertex $\vec{x}$ in $\chi^K(S^n)$, there is an $\vec{s}$ in $\sigma(S^n)$ such that

$$st(\vec{x}, \chi^K(S^n)) \subset st^\circ(\vec{s}, \sigma(S^n)) \text{ and } id(\vec{x}) = id(\vec{s}). \qquad (3)$$

An immediate difficulty is that $\vec{x}$ itself may not lie in the open star of any $\vec{s}$ of matching color. This situation occurs exactly when $\vec{x}$ lies in $lk(\vec{s}, \sigma(S^n))$, for $\vec{s}$ of matching color. In such a case, however, we can displace $\vec{x}$ by some small distance ϵ within its carrier to bring it within the open star of $\vec{s}$. We refer to such a process as an *ϵ-perturbation* (Figure 32). By applying a suitable ϵ-perturbation, we can ensure that every $\vec{x}$ lies within an open star of matching color.

The *extended star* of a simplex S is the union of the stars of its vertices. For a sufficiently large K, we can ensure that for every n-simplex X^n in $\chi^K(S^n)$, the extended star of X^n lies within some open star $st^\circ(\vec{s}, \sigma(S^n))$. Since the vertexes of X^n are labeled with all n colors, X^n must include one vertex $\vec{x}$ whose color matches that of $\vec{s}$. By construction, $\vec{x}$ and $\vec{s}$ satisfy Equation 3. The vertex map $\phi(\vec{x}) = \vec{s}$ is color and carrier-preserving, and it is defined on one vertex of each n-simplex of $\chi^K(S^n)$.

The remaining vertexes for which ϕ is *not* defined span an $(n-1)$-dimensional subcomplex of $\chi^K(S^n)$. We repeat essentially the same construction in a sequence of rounds. In each round, by applying a perturbation and further subdivision, we extend ϕ to one more vertex of each remaining simplex, and the dimension of the subcomplex on which ϕ remains undefined drops by one. After $n+1$ rounds, we have constructed a color and carrier-preserving vertex map ϕ. Finally, we check that ϕ is simplicial.

Our proof proceeds as follows. In Section 5.2, we define the standard chromatic subdivision. In Section 5.3, we define the simplex agreement task and give an algorithm for solving it on the standard chromatic subdivision. In Section 5.4, if we iterate the standard chromatic subdivision a sufficient number of times, then there is a color and carrier-preserving map from the iterated standard chromatic subdivision to any chromatic subdivision. This claim implies that any simplex agreement protocol for the iterated chromatic subdivision yields a simplex agreement protocol for any chromatic subdivision.

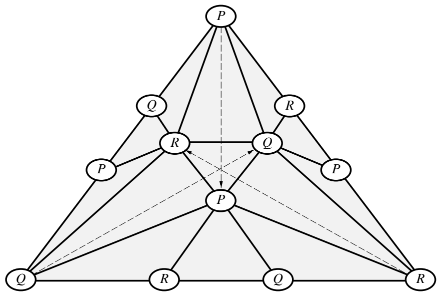

Figure 33: Standard chromatic subdivision

5.2 The Chromatic Subdivision

We start with a purely combinatorial definition of the standard chromatic subdivision. This definition is analogous to the combinatorial definition of the standard barycentric subdivision in Definition 2.27. Let $S^n = (\vec{s}_0, \dots, \vec{s}_n)$, where $id(\vec{s}_i) = P_i$.

Definition 5.3 *In the* standard chromatic subdivision *of S^n, denoted $\chi(S^n)$, each n-simplex has the form $(\langle P_0, S_0\rangle, \dots, \langle P_n, S_n\rangle)$, 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 33.

Definition 5.4 *The* iterated *standard chromatic subdivision $\chi^K(S^n)$ is the result of iterating the standard chromatic subdivision K times.*

It is sometimes useful to restrict subdivisions to a subcomplex.

Definition 5.5 *Let $\mathcal{C}$ be a complex with subcomplexes $\mathcal{A}$ and $\mathcal{B}$ such that every simplex C of $\mathcal{C}$ can be written uniquely as $A \cdot B$ (the* join *of A and B), where A is a simplex of $\mathcal{A}$, and B of $\mathcal{B}$. (Either A or B could be empty.) Any subdivision γ of $\mathcal{A}$ induces a subdivision $\gamma(\mathcal{C}/\mathcal{B})$ of $\mathcal{C}$, called "*γ of $\mathcal{C}$ holding $\mathcal{B}$ fixed"*, defined to be the complex of all joins $A' \cdot B$ where $A' \in \gamma(\mathcal{A})$, $B \in \mathcal{B}$, and $carrier(A', \mathcal{A}) \cdot B$ a simplex of $\mathcal{C}$.*

For the standard chromatic subdivision, the process can be repeated: $\chi^2(\mathcal{C}/\mathcal{B}) = \chi(\chi(\mathcal{C}/\mathcal{B})/\mathcal{B})$, and so on.

Lemma 5.6 *There is a color and carrier-preserving simplicial map*

$$\phi : \chi^K(\mathcal{C}) \to \chi^K(\mathcal{C}/\mathcal{B}).$$

Proof: Let $\vec{x}$ be a vertex in $\chi^K(\mathcal{C})$. Vertex $\vec{x}$ lies on simplex C, which can be uniquely decomposed as $A \cdot B$, where $A \in \chi^K(\mathcal{A})$ and $B \in \mathcal{B}$. If $id(\vec{x}) \in ids(\mathcal{B})$, then $\phi(\vec{x})$ is the unique vertex of B with matching id. Otherwise, $\phi(\vec{x}) = \vec{x}$. ∎

We now give an equivalent geometric definition of of the standard chromatic subdivision. A proof that the two definitions are equivalent is given by Hoest [31]. To avoid notational clutter, we use $f\langle P, v\rangle$ as shorthand for $f(\langle P, v\rangle)$.

Definition 5.7 *We construct the following homeomorphism $\iota : |\chi(S)| \to |S|$ inductively by dimension. Assume inductively that there exist homeomorphisms*

$$\iota_i : |\chi(face_i(S))| \to |face_i(S)|.$$

Let $\vec{b} = \sum_{i=0}^{m}(\vec{s}_i/(m+1))$ be the barycenter of S, where $m = \dim(S)$, and δ any value such that $0 < \delta < 1/(m+1)$. Define for id P_i and simplex R,

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

See Figure 33.

For any value of δ such that $0 < \delta < 1/(m+1)$, this definition gives an exact geometric construction for the chromatic subdivision. We will use this construction for the remainder of this section.

Definition 5.8 *The* mesh *of a complex is the maximum diameter of any simplex.*

Lemma 5.9 $mesh(\chi(S^n)) \leq \frac{n}{n+1} diam(S^n)$.

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 $\chi(\dot{\mathcal{S}}^{n-1})$. From Definition 5.7, 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 $mesh(\beta(S^n)) \leq (n/(n+1))diam(S^n)$ [37, Theorem 15.4], $|\vec{x} - \vec{x}_i| \leq (n/(n+1))diam(S^n)$. Finally, for any central vertex $\vec{x}_j$, $|\vec{x}_i - \vec{x}_j| = \delta|\vec{s}_i - \vec{s}_j|$, and the claim follows because $\delta < 1/(n+1)$. ■

Lemma 5.9 implies that by taking sufficiently large K, $mesh(\chi^K(\mathcal{I}))$ can be made arbitrarily small.

5.3 Simplex Agreement

Consider a task $\langle \mathcal{I}, \mathcal{O}, \Delta \rangle$ together with a subdivision σ and map $\mu : \sigma(\mathcal{I}) \to \mathcal{O}$ satisfying the conditions of the theorem. As described above, we will reduce any protocol to a "simplex agreement" protocol in which processes converge to the vertexes of a single simplex in $\sigma(\mathcal{I})$. More precisely:

Definition 5.10 *Let $\mathcal{I}$ be an $(n+1)$-colored complex, and σ a chromatic subdivision of $\mathcal{I}$. The* simplex agreement task *$\langle \mathcal{I}, \sigma(\mathcal{I}), \sigma \rangle$ has input complex $\mathcal{I}$, output complex $\sigma(\mathcal{I})$, and a task specification*

$$\sigma = \{(S^m, T^m) | T^m \in \sigma(S^m)\}.$$

For brevity, "simplex agreement on $\sigma(\mathcal{I})$" means $\langle \mathcal{I}, \sigma(\mathcal{I}), \sigma \rangle$.

Given a task $\langle \mathcal{I}, \mathcal{O}, \Delta \rangle$ satisfying the conditions of Theorem 3.1, let $\sigma(\mathcal{I})$ and $\mu : \sigma(\mathcal{I}) \to \mathcal{O}$ be the subdivision and simplicial map guaranteed by the theorem. Any protocol that solves the simplex agreement task $\langle \mathcal{I}, \sigma(\mathcal{I}), \sigma \rangle$ can be adapted to solve $\langle \mathcal{I}, \mathcal{O}, \Delta \rangle$ simply by applying μ to the result of the simplex agreement protocol.

Combining these observations yields a protocol for any task $\langle \mathcal{I}, \mathcal{O}, \Delta \rangle$ that satisfies the conditions of Theorem 3.1: each process executes the simplex agreement protocol for $\chi^K(\mathcal{I})$, and applies ϕ and then μ to the result.

Lemma 5.11 *There exists a wait-free protocol for simplex agreement on $\chi(\mathcal{I})$.*

```
procedure ParticipatingSet(int me, array F)
  array view_F[0..n] = {null, ..., null}
  repeat
    F[me] := F[me]-1;
    for i in 0 .. n do
      view_F[i] := F[i]
    S = {i | view_F[i] <= F[i]}
    until |S| >= F[me]
  return S;
```

Figure 34: The Participating Set Protocol

```
shared
  array F[1..k][0..n] = {n+2, ..., n+2};        // all n+2
  array vertex[0..k][0..n] = {null, ..., null} // all null

SimplexAgree(int me, vertex input, int k)
  array S[1..k] = {{}, ..., {}};
  vertex[0][me] = input;
  for i in 1 .. k do
    S[i] = ParticipatingSet(me, F[i]);
    vertex[i][me] = <i, {vertex[i-1][j] | j in S[i]}>
  return vertex[k][me];
```

Figure 35: The Iterated Participating Set Protocol

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 [12], developed as part of their "immediate snapshot" algorithm. Their elegant wait-free solution appears in Figure 34. Borowsky [10] gives a proof of this algorithm. ■

Lemma 5.12 *There is a wait-free solution for simplex agreement on $\chi^K(\mathcal{I})$, for any $K > 0$.*

Proof: Iterate the participating set algorithm [12] K times. ■

5.4 Mapping Subdivisions

In this section we prove a number of lemmas about subdivisions. Recall from Definition 2.21 that any point $\vec{s}$ in $|\mathcal{I}|$ has a *barycentric representation*

$$\vec{s} = \sum_{i=0}^{n} s_i \cdot \vec{s}_i$$

where each $0 \leq s_i \leq 1$, $\sum_i s_i = 1$, and the $\vec{s}_i$ span a simplex of $\mathcal{I}$. The s_i are called the *barycentric coordinates* of $\vec{s}$ with respect to $\mathcal{I}$, and $carrier(\vec{s}, \mathcal{I})$ is the simplex of $\mathcal{I}$ spanned by the $\vec{s}_i$ for which $s_i > 0$.

Recall that a sequence of vertexes $\vec{s}_0, \ldots, \vec{s}_k$ is *affinely independent* if $\vec{s}_1 - \vec{s}_0, \ldots, \vec{s}_k - \vec{s}_0$ are linearly independent. (The vertexes of a simplex are affinely independent by definition.) Any sequence of vertexes $\vec{s}_0, \vec{s}_1, \ldots, \vec{s}_k$ determines a *hyperplane*, denoted $hyper(\vec{s}_0, \vec{s}_1, \ldots, \vec{s}_k)$, defined to be the set of points expressible as

$$\vec{s} = \sum_{i=0}^{k} s_i \cdot \vec{s}_i$$

where $\sum_{i=0}^{k} s_i = 1$. A hyperplane's *dimension* is one less than the size of the smallest set of vertexes that determines that hyperplane. If $\vec{s}$ is a vertex of any chromatic subdivision of S^n, and $\vec{s}_i$ the vertex of S^n such that $id(\vec{s}) = id(\vec{s}_i)$, then $\vec{s}_i \in carrier(\vec{s}, S^n)$. This terminology extends to simplexes in a natural way. Simplexes $A = (\vec{a}_0, \ldots, \vec{a}_k)$ and $B = (\vec{b}_0, \ldots, \vec{b}_\ell)$ are *affinely independent* if the sequence $\vec{a}_0, \ldots, \vec{a}_k, \vec{b}_0, \ldots, \vec{b}_\ell$ is affinely independent. Define the hyperplane $hyper(A)$ to be $hyper(\vec{a}_0, \ldots, \vec{a}_k)$. If H is a hyperplane, and $\vec{a}$ a point of H, the *open ϵ-ball* around $\vec{a}$ in H is the set of points $\vec{b}$ in H such that $|\vec{a} - \vec{b}| < \epsilon$.

Lemma 5.13 *Let H and K be hyperplanes, and $\vec{a}$ a point in H but not in K. For some $\epsilon > 0$, the open ϵ-ball around $\vec{a}$ does not intersect K.*

Proof: $H \cap K$ is a closed subset of H, and its complement $H - K$ is open and non-empty, so there exists $\epsilon > 0$ such that $H - K$ contains the open ϵ-ball around $\vec{a}$ in H. ■

Informally, an *ϵ-perturbation* of a subdivision is a new subdivision constructed by slightly displacing some set of vertexes within their respective carriers (Figure 32). Formally,

Definition 5.14 *Let α be a subdivision of $\mathcal{I}$, and $\epsilon > 0$. A subdivision α_* of $\mathcal{I}$ is an* ϵ-perturbation *of α if there is an isomorphism $\iota : \alpha(\mathcal{I}) \to \alpha_*(\mathcal{I})$ such that for every vertex $\vec{a}$ of $\alpha(\mathcal{I})$, $carrier(\vec{a}, \mathcal{I}) = carrier(\iota(\vec{a}), \mathcal{I})$ and $|\vec{a} - \iota(\vec{a})| < \epsilon$.*

For brevity, when we speak of *perturbing* a vertex $\vec{a}$ by ϵ in a subdivision $\alpha(\mathcal{I})$, we mean constructing a new subdivision $\alpha_*(\mathcal{I})$ by replacing $\vec{a}$ with a $\vec{b}$ that lies in the open ϵ-ball around $\vec{a}$ in $carrier(\vec{a}, \mathcal{I})$.

We now show that any vertex of a subdivision can be perturbed by a sufficiently small amount. We exploit the following lemma from Munkres [37, Lemma 15.2].

Lemma 5.15 *If $\{K_i\}$ is a collection of complexes in Euclidean space, and if every $|K_i| \cap |K_j|$ is the polyhedron of a subcomplex of both K_i and K_j, then $\cup K_i$ is a complex.*

Lemma 5.16 *Let α be a subdivision of $\mathcal{I}$, and $\vec{a}$ a vertex of $\alpha(\mathcal{I})$. There exists $\epsilon_0 > 0$ such that any perturbation of $\vec{a}$ by ϵ_0 in α yields an ϵ_0-perturbation α_* of $\alpha(\mathcal{I})$.*

Proof: Let $C = carrier(\vec{a}, \mathcal{I})$. Because $\alpha(C)$ is a manifold with boundary, by Lemma 2.36, $lk(\vec{a}, \alpha(C))$ is a sphere, and $\vec{a}$ is an interior point of $st^\circ(\vec{a}, \alpha(C))$. There thus exists $\epsilon > 0$ such that the open ϵ-ball around $\vec{a}$ in $|C|$ lies in $st^\circ(\vec{a}, \alpha(C))$. Let $A_0^{n-1}, \ldots, A_N^{n-1}$ be the $(n-1)$-simplexes of $lk(\vec{a}, \alpha(\mathcal{I}))$. For $0 \leq i \leq N$, each $\vec{a} \cdot A_i^{n-1}$ is a simplex of $\alpha(\mathcal{I})$, hence $\vec{a}$ is affinely independent of A_i^{n-1}, and $\vec{a}$ does not lie on $hyper(A_i^{n-1})$. By Lemma 5.13, there exists $\varepsilon_i > 0$ such that every point $\vec{b}$ of $hyper(C)$ within ε_i of $\vec{a}$ does not lie on $hyper(A_i^{n-1})$. Let $\epsilon_0 = \min(\epsilon, \varepsilon_0, \ldots, \varepsilon_N)$. It follows that every $\vec{b}$ within ϵ_0 of $\vec{a}$ in $\alpha(C)$ lies within $st^\circ(\vec{a}, \alpha(C))$, and is affinely independent of $A_0^{n-1}, \ldots, A_N^{n-1}$. Each $\vec{b} \cdot A_i^{n-1}$ is thus a simplex, and $\vec{b} \cdot lk(\vec{a}, \alpha(\mathcal{I}))$ is a complex with polyhedron identical to the polyhedron of $\vec{a} \cdot lk(\vec{a}, \alpha(\mathcal{I})) = st(\vec{a}, \alpha(\mathcal{I}))$.

Let $\mathcal{A}$ be the complex consisting of all simplexes in $\alpha(\mathcal{I})$ that do not contain $\vec{a}$, and let $\mathcal{B} = \vec{b} \cdot lk(\vec{a}, \alpha(\mathcal{I}))$. The intersection $|\mathcal{A}| \cap |\mathcal{B}|$ is the polyhedron of the complex $lk(\vec{a}, \alpha(\mathcal{I}))$, so by Lemma 5.15, $\mathcal{A} \cup \mathcal{B}$ is a complex. Since $|\mathcal{A} \cup \mathcal{B}| = |\mathcal{I}|$, this complex is the desired subdivision $\alpha_*(\mathcal{I})$. ∎

Lemma 5.17 *If α is a subdivision of $\mathcal{I}$, $\vec{x}$ a point of $|\alpha(\mathcal{I})|$, $A = carrier(\vec{x}, \alpha(\mathcal{I}))$, and $\vec{a}$ a vertex of A, then $\vec{x} \in st^\circ(\vec{a}, \alpha(\mathcal{I}))$.*

Proof: If not, then $\vec{x}$ lies on a proper face of A that does not contain $\vec{a}$, contradicting the hypothesis that $A = carrier(\vec{x}, \alpha(\mathcal{I}))$. ∎

In the remainder of this section, let $\alpha(\mathcal{I})$ and $\beta(\mathcal{I})$ be two possible chromatic subdivisions of a complex $\mathcal{I}$, and $\mathcal{B}$ a subcomplex of $\beta(\mathcal{I})$. We focus on the relation between α and $\mathcal{B}$.

Definition 5.18 *Subdivision α is a* chromatic cover *for $\mathcal{B}$ if every simplex B of $\mathcal{B}$ is covered by the open stars of vertexes of $\alpha(\mathcal{I})$ labeled with process ids from $ids(B)$:*

$$(\forall B \in \mathcal{B}) \quad B \subset \bigcup_{id(\vec{a}) \in ids(B)} st^\circ(\vec{a}, \alpha(\mathcal{I})).$$

Definition 5.19 *Simplexes A in $\alpha(\mathcal{I})$ and B in $\mathcal{B}$ are* mismatched *if $ids(A)$ and $ids(B)$ are disjoint, but $|A|$ intersects $|B|$.*

Lemma 5.20 *Subdivision α is a chromatic cover for $\mathcal{B}$ if and only if $\alpha(\mathcal{I})$ and $\mathcal{B}$ contain no mismatched simplexes.*

Proof: Assume there are no mismatched simplexes: for all $A \in \alpha(\mathcal{I})$ and $B \in \mathcal{B}$, if $ids(A)$ and $ids(B)$ are disjoint, so are $|A|$ and $|B|$. Let $\vec{x}$ be a point of $|\mathcal{B}|$, $A = carrier(\vec{x}, \alpha(\mathcal{I}))$, and $B = carrier(\vec{x}, \mathcal{B})$. Since $|A|$ and $|B|$ intersect at $\vec{x}$, $ids(A)$ and $ids(B)$ must also intersect, and therefore A contains a vertex $\vec{a}$ such that $id(\vec{a}) \in ids(B)$. By Lemma 5.17, $\vec{x} \in st^\circ(\vec{a}, \alpha(\mathcal{I}))$. It follows that α is a chromatic cover for $\mathcal{B}$.

Assume that α is a chromatic cover for $\mathcal{B}$. If A is a simplex of $\alpha(\mathcal{I})$, then for any vertex $\vec{a}$ of $\alpha(\mathcal{I})$ where $id(\vec{a}) \notin ids(A)$, $|A|$ does not intersect $st^\circ(\vec{a}, \alpha(\mathcal{I}))$. Because α is a chromatic cover for $\mathcal{B}$, if $ids(A)$ and $ids(B)$ are disjoint, then every point of $|B|$ lies in such an open star $st^\circ(\vec{a}, \alpha(\mathcal{I}))$, so $|A|$ and $|B|$ are disjoint. ∎

Next we show that we can perturb any vertex of any subdivision of $\mathcal{B}$ without introducing any additional mismatches.

Lemma 5.21 *Let γ be a subdivision of $\mathcal{B}$, and $\vec{g}$ a vertex of $\gamma(\mathcal{B})$. There exists $\epsilon_1 > 0$ such that any perturbation of $\vec{g}$ by ϵ_1 yields an ϵ_1-perturbation γ_* of γ such that the number of mismatched simplexes between $\alpha(\mathcal{I})$ and $\gamma_*(\mathcal{B})$ is no greater than between $\alpha(\mathcal{I})$ and $\gamma(\mathcal{B})$.*

Proof: Lemma 5.16 states that there exists $\epsilon_0 > 0$ such that γ remains a subdivision if we perturb $\vec{g}$ by ϵ_0. Let $\{G_i\}$ be the set of simplexes of $\gamma(\mathcal{B})$ that have $\vec{g}$ as a vertex, and let $\{A_{ij}\}$ be the following set of simplexes of $\alpha(\mathcal{I})$ that are not mismatched with G_i: $ids(G_i) \cap ids(A_{ij}) = \emptyset$ and $|G_i| \cap |A_{ij}| = \emptyset$.

Let δ_{ij} be the minimum distance from any point of $|G_i|$ to any point of $|A_{ij}|$ (well-defined because both are closed sets). Define

$$\epsilon_1 = \min(\epsilon_0, \min_{ij}(\delta_{ij}))$$

This minimum is well-defined and positive because δ_{ij} ranges over a finite number of positive distances. Perturbing $\vec{g}$ by ϵ_1 yields a new subdivision γ_* with no additional mismatched simplexes. ■

Lemma 5.22 *If α is a chromatic cover for $\mathcal{B}$, then for all $\epsilon > 0$, α is a chromatic cover for an ϵ-perturbation of $\chi(\mathcal{B})$.*

Proof: If $\alpha(\mathcal{I})$ is not a chromatic cover for $\chi(\mathcal{B})$, then by Lemma 5.20 there exist mismatched simplexes $C = (\vec{c}_0, \ldots, \vec{c}_c)$ of $\chi(\mathcal{B})$, and $A = (\vec{a}_0, \ldots, \vec{a}_a)$ of $\alpha(\mathcal{I})$. Pick C and A to have minimal dimensions a and c in the sense that no proper face of C intersects A, and vice-versa, and let $B = \mathit{carrier}(C, \mathcal{B})$ of dimension b. Because B is covered by open stars of the form $st^\circ(\vec{a}, \alpha(\mathcal{I}))$, where $\mathit{id}(\vec{a}) \in \mathit{ids}(B)$, and because A has minimal dimension, $\mathit{ids}(A) \subseteq \mathit{ids}(B) - \mathit{ids}(C)$, or $a \leq b - c - 1$.

Because C is a simplex of $\chi(B)$, C includes a central vertex of B whose carrier in $\mathcal{B}$ is B. By reindexing, let this vertex be $\vec{c}_c$. By Lemma 5.21, there exists $\epsilon_1 > 0$ such that any perturbation of $\vec{c}_c$ by ϵ_1 introduces no additional mismatches. Let $\epsilon_2 = \min(\epsilon, \epsilon_1)$.

Let $H = \mathit{hyper}(\vec{a}_0, \ldots, \vec{a}_a, \vec{c}_0, \ldots, \vec{c}_{c-1})$. H has dimension at most $a + c$, which is strictly less than b, so B contains a point $\vec{b}$ not in H. Let

$$\varepsilon = |\vec{b} - \vec{c}_c|/\epsilon_1$$
$$\vec{c} = (1 - \varepsilon) \cdot \vec{c}_c + \varepsilon \cdot \vec{b}$$

Because $\vec{c} \in \mathit{carrier}(\vec{c}_c, \mathcal{B}) = B$, replacing $\vec{c}_c$ by $\vec{c}$ in $\chi(\mathcal{B})$ yields an ϵ_1-perturbation $\chi_*(\mathcal{B})$ with no additional mismatches. Let $C' = (\vec{c}_0, \ldots, \vec{c}_{c-1}, \vec{c})$. We claim that C' does not intersect A. Consider the barycentric coordinates of points of $|C'|$ with respect to C. Because C has minimal dimension, $(\vec{c}_0, \ldots, \vec{c}_{c-1})$ does not intersect A, so any point of $|C'|$ whose c-th barycentric coordinate is zero does not lie in $|A|$. By construction, $\vec{c}$ does not lie in the hyperplane H (or in $|A|$) and neither does any point whose c-th barycentric coordinate is positive. P We have constructed an ϵ-perturbation $\chi_*(\mathcal{B})$ of $\chi(\mathcal{B})$ with strictly fewer mismatches with $\alpha(\mathcal{I})$. The claim now follows from a simple inductive argument. ■

Lemma 5.23 *If α is a chromatic cover for $\mathcal{B}$, then for all $\epsilon > 0$, and $K \geq 1$, $\alpha(\mathcal{I})$ is a chromatic cover for an ϵ-perturbation of $\chi^K(\mathcal{B})$.*

Proof: Let $\epsilon_i = (1 - \frac{1}{2^i})\epsilon$. We show inductively that for any $1 \leq i \leq K$, α is a chromatic cover for an ϵ_i-perturbation of $\chi^i(\mathcal{B})$. For the base case, $\alpha(\mathcal{I})$ is a chromatic cover for $\mathcal{B}$ by construction. As induction hypothesis, assume $\alpha(\mathcal{I})$ is a chromatic cover for $\chi^i_*(\mathcal{B})$, an ϵ_i-perturbation of $\chi^i(\mathcal{B})$. Lemma 5.22 states that $\alpha(\mathcal{I})$ is a chromatic cover for an $\frac{\epsilon}{2^{i+1}}$-perturbation $\chi^{i+1}_*(\mathcal{B})$ of $\chi(\chi^i_*(\mathcal{B}))$. Every vertex of $\chi^i_*(\mathcal{B})$ is displaced by at most ϵ_i from its corresponding vertex in $\chi^i(\mathcal{B})$, and the last perturbation adds at most $\epsilon/2^{i+1}$, yielding a final maximal displacement of $\epsilon_i + \epsilon/2^{i+1} = \epsilon_{i+1}$. which is an $((i+1)\epsilon/K)$-perturbation of $\chi^{i+1}(\mathcal{B})$. We have shown that α is a chromatic cover for an ϵ_i-perturbation of χ_i, and the lemma follows because $\epsilon_i < \epsilon$, for all $i \geq 1$. ∎

Definition 5.24 *Define the* extended star $st^*(S, \mathcal{I})$ *of a simplex S in $\mathcal{I}$ to be*

$$st^*(S, \mathcal{I}) = \bigcup_{\vec{s} \in S} st(\vec{s}, \mathcal{I}).$$

The next two lemmas are left as an exercise for the reader.

Lemma 5.25 *If α is a subdivision of $\mathcal{I}$, and A a simplex of $\alpha(\mathcal{I})$,*

$$diam(st^*(A, \alpha(\mathcal{I}))) \leq 3 \cdot mesh(\alpha(\mathcal{I})).$$

Lemma 5.26 *If β is a subdivision of $\mathcal{B}$, and β_* an ϵ-perturbation of β, then $mesh(\beta_*(\mathcal{B})) < mesh(\beta(\mathcal{B})) + 2\epsilon$.*

We will need the following lemma from Spanier.

Lemma 5.27 ([39, 2.1.25]) *A set of vertexes $\vec{v}_0, \ldots, \vec{v}_m$ belong to a common m-simplex of $\mathcal{I}$ if and only if*

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

The next lemma is technical, but it encompasses most of the work needed to prove our main theorem.

Lemma 5.28 *If α is a chromatic cover for $\mathcal{B}$, then there exist $K \geq 0$, a subdivision γ of $\mathcal{B}$, and color and carrier-preserving simplicial maps ξ and ψ, where*

$$\chi^K(\mathcal{B}) \xrightarrow{\xi} \gamma(\mathcal{B}) \xrightarrow{\psi} \alpha(\mathcal{I}) \tag{4}$$

such that every simplex X in $\gamma(\mathcal{B})$ includes a vertex $\vec{y}$ where

$$\vec{y} \in \bigcap_{\vec{x} \in X} st^\circ(\psi(\vec{x}), \alpha(\mathcal{I})). \tag{5}$$

Proof: We argue by induction on the dimension of $\mathcal{B}$. In the base case, this dimension is zero. Because α is a chromatic cover for $\mathcal{B}$, each vertex $\vec{b}$ of $\mathcal{B}$ lies in $st^\circ(\vec{a}, \alpha(\mathcal{I}))$, for a unique $\vec{a}$ where $id(\vec{b}) = id(\vec{a})$. Define $K = 0$, ξ the identity map, γ the trivial subdivision that introduces no new vertexes, and $\psi(\vec{b}) = \vec{a}$.

For the induction hypothesis, assume the claim for subcomplexes of dimension less than i. Consider $\mathcal{B}$ of dimension i. Because α is a chromatic cover for $\mathcal{B}$, each i-simplex B of $\mathcal{B}$ has a covering by open sets $st^\circ(\vec{a}, \alpha(\mathcal{I}))$, where $id(\vec{a}) \in ids(B)$.

Let λ be the minimum Lebesgue number of any such covering (well-defined because $\mathcal{B}$ is finite). By Lemma 5.9, we can pick K_0 large enough that $mesh(\chi^{K_0}(\mathcal{B})) < \lambda/9$. By Lemma 5.23, $\chi^{K_0}(\mathcal{B})$ has a $(\lambda/9)$-perturbation $\chi_*^{K_0}(\mathcal{B})$ for which α is a chromatic cover. By Lemma 5.26, this perturbation adds at most $2\lambda/9$ to the diameter of any simplex in the subdivision, so $mesh(\chi_*^{K_0}(\mathcal{B})) < \lambda/3$. By Lemma 5.25, for every $X^i \in \chi_*^{K_0}(\mathcal{B})$,

$$diam(st^*(X^i, \chi_*^{K_0}(\mathcal{B}))) < 3 \cdot mesh(\chi_*^{K_0}(\mathcal{B})) < \lambda,$$

so $st^*(X^i, \chi_*^{K_0}(\mathcal{B})) \subset st^\circ(\vec{a}, \alpha(\mathcal{I}))$ for some vertex $\vec{a}$ in $\alpha(\mathcal{I})$ where $id(\vec{a}) \in ids(X^i)$. X^i has at least one vertex $\vec{x}$ such that

$$st(\vec{x}, \chi_*^{K_0}(\mathcal{B})) \subset st^\circ(\vec{a}, \alpha(\mathcal{I})) \tag{6}$$

for some $\vec{a}$ in $\alpha(\mathcal{I})$ where $id(\vec{a}) \in ids(X^i)$. Define $\psi(\vec{x}) = \vec{a}$. Let $\mathcal{B}_0$ be the subcomplex of $\chi_*^{K_0}(\mathcal{B})$ spanned by vertexes that satisfy Equation 6, and $\mathcal{B}_1$ the subcomplex spanned by vertexes that do not. $\mathcal{B}_1$ has dimension at most $i-1$ because each i-simplex of $\chi_*^{K_0}(\mathcal{B})$ includes at least one vertex in $\mathcal{B}_0$.

By the induction hypothesis, there exist $K_1 \geq 0$, a subdivision γ' of $\mathcal{B}_1$, and color and carrier-preserving simplicial maps ξ' and ψ

$$\chi^{K_1}(\mathcal{B}_1) \xrightarrow{\xi'} \gamma'(\mathcal{B}_1) \xrightarrow{\psi} \alpha(\mathcal{I}) \tag{7}$$

such that every simplex X_1 in $\gamma'(\mathcal{B}_1)$ includes a vertex $\vec{y}$ that

$$\vec{y} \in \bigcap_{\vec{x} \in X_1} st^\circ(\psi(\vec{x}), \alpha(\mathcal{I})). \tag{8}$$

Let $\gamma(\mathcal{B}) = \gamma'(\mathcal{B}_1/\mathcal{B}_0)$, the subdivision of $\mathcal{B}$ constructed by subdividing $\mathcal{B}_1$ by γ' while leaving $\mathcal{B}_0$ fixed. We have defined a color and carrier-preserving vertex map $\psi : \gamma(\mathcal{B}) \to \alpha(\mathcal{I})$. We now prove that ψ is simplicial. Every simplex X in $\gamma(\mathcal{B})$ is a join $X_0 \cdot X_1$, where X_0 is a simplex in $\mathcal{B}_0$ and X_1 in $\gamma'(\mathcal{B}_1)$. For every vertex $\vec{x}_0$ of X_0,

$$\vec{y} \in X_1 \subset st(\vec{x}_0, \chi_*^{K_0}(\mathcal{B})).$$

By Equation 6,

$$st(\vec{x}_0, \chi_*^{K_0}(\mathcal{B}) \subset st^\circ(\psi(\vec{x}_0), \alpha(\mathcal{I})).$$

Combining these equations with Equation 8,

$$\vec{y} \in \bigcap_{\vec{x} \in X} st^\circ(\psi(\vec{x}), \alpha(\mathcal{I})).$$

By Lemma 5.27, the $\psi(\vec{x})$ span a vertex of $\alpha(\mathcal{I})$, so ψ is a simplicial map. So far we have shown that there exists a simplicial map

$$\gamma(\mathcal{B}) \xrightarrow{\psi} \alpha(\mathcal{I})$$

such that every simplex X in $\gamma(\mathcal{B})$ includes a vertex $\vec{y}$ that

$$\vec{y} \in \bigcap_{\vec{x} \in X} st^\circ(\psi(\vec{x}), \alpha(\mathcal{I})).$$

To complete the proof, we show that

$$\xi' : \chi^{K_1}(\mathcal{B}_1) \to \gamma'(\mathcal{B}_1)$$

can be extended to

$$\xi'' : \chi^{K_1}(\mathcal{B}/\mathcal{B}_0) \to \gamma(\mathcal{B})$$

by making ξ'' the identity on vertexes of $\mathcal{B}_0$. If X_0 is a simplex of $\mathcal{B}_0$, and X_1 in $\chi^{K_1}(\mathcal{B}_1)$, then the join $X_0 \cdot X_1$ is in $\chi^{K_1}(\mathcal{B}/\mathcal{B}_0)$ if and only if $carrier(X_1, \chi_*^{K_0}(\mathcal{B})) \cdot X_0$ is a simplex of $\chi_*^{K_0}(\mathcal{B})$. Similarly, if Y_1 is a simplex of $\gamma'(\mathcal{B}_1)$, $X_0 \cdot Y_1$ is in $\gamma(\mathcal{B}) = \gamma'(\mathcal{B}/\mathcal{B}_0)$ if and only if $carrier(Y_1, \chi_*^{K_0}(\mathcal{B})) \cdot X_0$ is a simplex of $\chi_*^{K_0}(\mathcal{B})$. By the induction hypothesis, ξ' is carrier-preserving, so $carrier(X_1, \chi_*^{K_0}(\mathcal{B})) = carrier(\xi'(X_1), \chi_*^{K_0}(\mathcal{B}))$, and therefore ξ'' is a simplicial map.

By Lemma 5.6, there is a color and carrier-preserving simplicial map

$$\chi^{K_1}(\chi_*^{K_0}(\mathcal{B})) \to \chi^{K_1}(\mathcal{B}/\mathcal{B}_0)$$

and an isomorphism

$$\chi^{K_0+K_1}(\mathcal{B}) \to \chi^{K_1}(\chi_*^{K_0}(\mathcal{B}))$$

Composing these maps yields a color and carrier-preserving simplicial map

$$\xi : \chi^{K_0+K_1}(\mathcal{B}) \to \gamma(\mathcal{B}).$$

completing the proof of Equation 4. ■

Theorem 5.29 *If σ is a chromatic subdivision of a complex $\mathcal{I}$, then there exists $K \geq 0$ and a color and carrier-preserving simplicial map*

$$\phi : \chi^K(\mathcal{I}) \to \sigma(\mathcal{I}).$$

Proof: Note that $\sigma(\mathcal{I})$ is a chromatic cover for $\mathcal{I}$, so by Lemma 5.28, there exists $K > 0$ and a color and carrier-preserving simplicial maps

$$\chi^K(\mathcal{I}) \xrightarrow{\xi} \gamma(\mathcal{I}) \xrightarrow{\psi} \sigma(\mathcal{I})$$

Define $\phi : \chi^K(\mathcal{I}) \to \sigma(\mathcal{I})$ to be the composition of ξ and ψ. The map ϕ is simplicial, color-preserving, and carrier-preserving. ■

Theorem 5.30 *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 color-preserving simplicial map*

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

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

Proof: Suppose the participating processes start with inputs given by simplex S^m. By Theorem 5.29, there exists a color and carrier-preserving map

$$\phi : \chi^K(S^m) \to \sigma(S^m).$$

By hypothesis, there exists a simplicial map:

$$\mu : \sigma(S^m) \to \Delta(S^m).$$

The protocol has the following steps:

1. Use the iterated participating set protocol to agree on a simplex in $\chi^K(S^m)$.

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

■

This completes the proof of the Asynchronous Computability Theorem.

Our construction shows that we can assume without loss of generality that the span σ is an iterated chromatic subdivision.

Corollary 5.31 *A decision task $\langle \mathcal{I}, \mathcal{O}, \Delta \rangle$ has a wait-free protocol using read-write memory if and only if there exists a $K \geq 0$ and a color-preserving simplicial map*

$$\mu : \chi^K(\mathcal{I}) \to \mathcal{O}$$

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

6 Renaming with a Small Number of Names

In the renaming task of Attiya *et al.* [7, 6], processes are issued unique input names from a large name space, and must choose unique output names taken from a smaller name space. To rule out trivial solutions, protocols must be *anonymous* [6, Section 17.3], meaning that the value any process chooses does not depend in any way on the value of any participant's process id (including its own). Informally, a process may choose its output value based only on the input name it received, and on how its memory accesses are interleaved with the memory accesses of the other processes. (We give a formal definition of anonymity below.)

In the message-passing model, Attiya et al. [7] showed that renaming has a wait-free solution when $K \geq 2n+1$, and none when $K \leq n+2$. Bar-Noy and Dolev [8] extended their upper bound solution to the shared read-write memory model. Whether a protocol exists for $n+2 < K \leq 2n$ names remained open until 1993, when Herlihy and Shavit [28] showed that no such protocol exists. Henceforth, by *renaming*, we mean the renaming task where $K \leq 2n$.

The restriction to anonymous protocols implies that the asynchronous computability theorem does not apply. Nevertheless, a variant of this theorem can be devised for anonymous protocols. In this section we show how to adapt the asynchronous computability theorem to prove that the renaming task has no anonymous wait-free read/write protocol. This section differs from previous sections in the level of technical detail: we make explicit use of elementary homology theory (as presented by [37]). Our original proof appeared in 1993 [28]. Here we present a simplified version of that proof, based on the chain map proof methodology developed by Herlihy and Rajsbaum [22].

6.1 Proof Outline

The key insight underlying our proof is that the anonymity requirement makes it impossible to break symmetry among certain input configurations. We capture and formalize this intuition through the following sequence of steps.

- We begin by giving a formal statement of the notion of anonymity [6]. We prove a variant of the asynchronous computability theorem for anonymous protocols. This theorem states that the map from a subdivision of the input complex to the output complex must satisfy certain symmetry properties. The theorem statement exploits the observation of Corollary 5.31 that we can restrict our attention to the iterated standard chromatic subdivision.

- We reduce the renaming task itself to a *reduced renaming* task in which processes choose boolean values instead of names. The resulting output complex is smaller and easier to work with. In particular, the complex has a single "hole" which will act as an obstruction to any protocol. For a larger number of names, the corresponding output complex has no hole, so the existence of this hole characterizes the boundary between solvable and unsolvable instances of renaming.

- We identify a subcomplex of the input complex satisfying two properties: the subcomplex itself is "solid", with no holes, and the inputs along its boundary are symmetric.

- We then apply the anonymous computability theorem to show that the simplicial map guaranteed by the theorem wraps the symmetric boundary of the solid subcomplex around the output complex's hole a non-zero number of times, a contradiction.

6.2 Anonymity

We now give a formal definition of the notion of anonymity used by Attiya and Welch [6] to define the renaming task. Let π be any permutation of $0, \ldots, n$. The permutation π acts on any labeled simplex by replacing each occurrence of a process id P in the label with the process id $\pi(P)$. Here are some examples.

- For an input or output simplex S^n, π sends each vertex $\langle P, v\rangle$ to $\langle \pi(P), v\rangle$.
- For the chromatic subdivision $\chi(S^n)$, π sends each vertex $\langle P, S\rangle$ to $\langle \pi(P), \pi(S)\rangle$, where S is a face of S^n. The reader is invited to check that $\pi(\chi(S)) = \chi(\pi(S))$.
- For a protocol complex $\mathcal{P}(\mathcal{I})$, π sends each vertex $\langle P, e\rangle$ to $\langle \pi(P), \pi(e)\rangle$, where $\pi(e)$ denotes the execution view in which each process id Q is replaced by $\pi(Q)$.

Definition 6.1 *A complex $\mathcal{C}$ is* symmetric *if π induces a simplicial map from $\mathcal{C}$ to itself, denoted $\pi : \mathcal{C} \to \mathcal{C}$. If $\mathcal{A}$ and $\mathcal{B}$ are symmetric complexes, then $\phi : \mathcal{A} \to \mathcal{B}$ is* symmetric under permutation *if $\pi(\phi(\vec{v})) = \phi(\pi(\vec{v}))$ for any permutation π. A task specification $\langle \mathcal{I}, \mathcal{O}, \Delta\rangle$ is* symmetric *if $\mathcal{I}$ and $\mathcal{O}$ are symmetric, and for all $S^n \in \mathcal{I}$, $\Delta(\pi(S^n)) = \pi(\Delta(S^n))$.*

In short, the problem specification depends only on input values, not process ids. This restriction is weak: all tasks considered in this paper (excluding our contrived quasi-consensus example of Section 3.2) have symmetric specifications.

In Section 5, we showed that any protocol can be expressed as an iterated "immediate snapshot" algorithm. This kind of protocol has the property that in any valid execution, replacing each process id P with $\pi(P)$ yields another valid execution. As a result, we can assume without loss of generality that any protocol complex is symmetric.

Definition 6.2 *A protocol $\mathcal{P}$ is* anonymous *if the decision map δ is symmetric under permutation: for every simplex T in $\mathcal{P}(S^n)$, $\pi(\delta(T)) = \delta(\pi(T))$.*

We now give the anonymous variant of the asynchronous computability theorem. The proof appears in the appendix, but it is essentially the same as the proof of the original, except that we use a specific symmetric subdivision based on the standard chromatic subdivision.

Theorem 6.3 (Anonymous Computability Theorem) *A symmetric decision task $\langle \mathcal{I}^n, \mathcal{O}^n, \Delta\rangle$ has a wait-free anonymous protocol using read-write memory if and only if there exists an integer K and a color-preserving simplicial map*

$$\mu : \chi^K(\mathcal{I}) \to \mathcal{O}$$

symmetric under permutation, such that for each simplex X in $\chi^K(\mathcal{I})$, $\mu(X) \in \Delta(carrier(X, \mathcal{I}))$.

6.3 Reduced Renaming

We now simplify the task by reducing the size of the output complex. Consider the following *reduced renaming* task.

Reduced Renaming Each 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 6.4 *If an anonymous renaming protocol exists, then so does an anonymous* reduced *renaming protocol.*

Proof: Any renaming protocol (anonymous or not) can be transformed into a reduced renaming protocol simply by taking the parity of the names chosen. If the original protocol was anonymous, so is the reduced protocol. ■

For example, the annulus of Figure 37 is the reduced output for the original three-process torus-shaped output complex of Figure 8.

6.4 A Three-Process Example

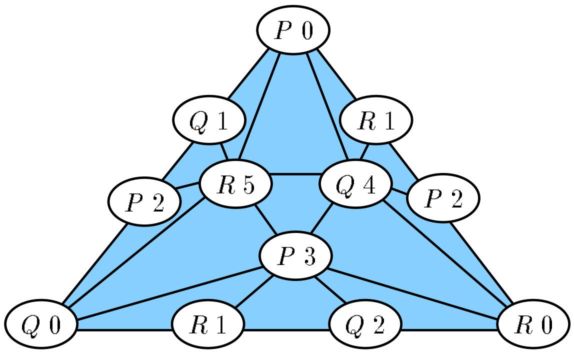

Figure 36: Input subcomplex $\mathcal{T}^2$

The full proof of the renaming lower bound is unavoidably technical, due to the need to reason formally about the structure of high-dimensional complexes. Nevertheless, the need for formal rigor should not be allowed

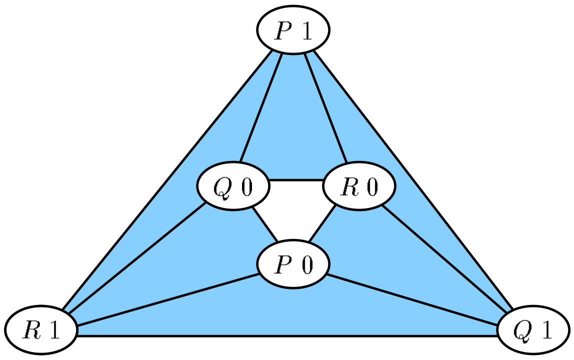

Figure 37: Reduced Renaming Output Complex $\mathcal{O}^2$

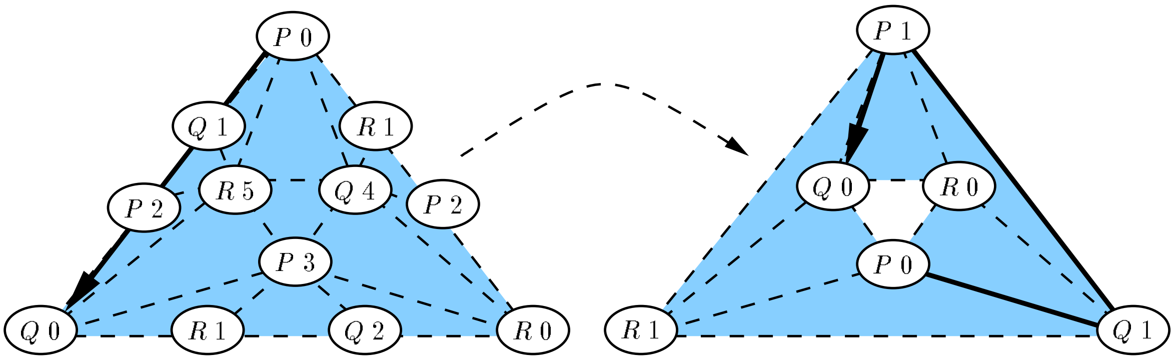

Figure 38: Mapping an edge of the input complex

to obscure the inherent geometric simplicity of the proof's essence. In this section, we describe a three-process example that illustrates the basic ideas underlying the full proof.

The output complex $\mathcal{O}^2$ for three-process reduced renaming, shown in Figure 37, is the annulus constructed by taking the binary 2-sphere over P, Q, and R, and removing the all-zero and all-one simplexes. As we will explain, the resulting hole will act as an "obstruction" to any anonymous protocol. The boundary between four and five names is exactly the boundary between a reduced renaming task whose output complex has a hole (impossible) and one that does not (possible). For notational convenience, we will use $P0$ as shorthand for $\langle P, 0\rangle$, and so on.

Assume we have an anonymous protocol for three-process reduced renaming. Consider the subcomplex $\mathcal{T}$ of the input complex shown in Figure 36. This subcomplex is isomorphic to $\chi(S^2)$, the standard chromatic sub-

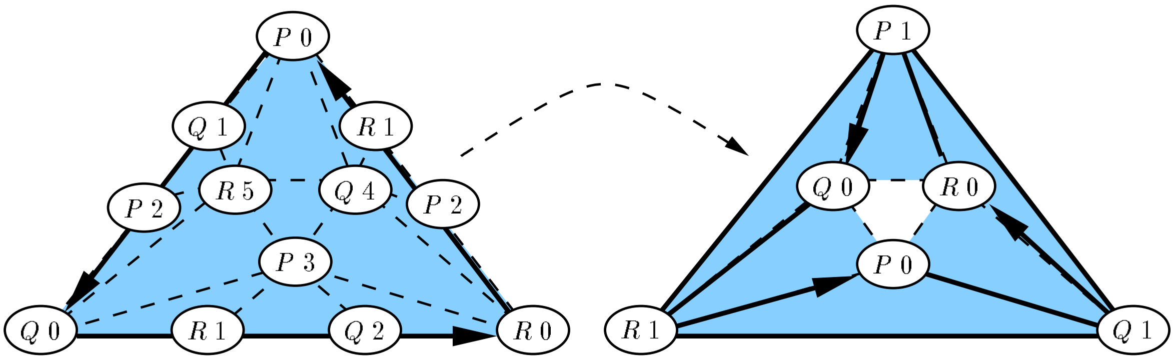

Figure 39: Mapping the boundary of the input complex

division of a 2-simplex. The anonymous computability theorem states that there is a simplicial map μ carrying a subdivision of $\mathcal{T}^2$ to the reduced renaming output complex $\mathcal{O}^2$. We will argue that any such map must wrap the boundary of $\mathcal{T}^2$ around the hole in $\mathcal{O}^2$ a non-zero number of times. As stated formally in Section 6.6, no continuous map can carry the boundary of a "solid" region ($\mathcal{T}^2$) to the boundary of a hole, so we have a contradiction.

The inputs along the boundary of $\mathcal{T}^2$ are symmetric in the following sense. Let $\mathcal{T}_P$, $\mathcal{T}_Q$, and $\mathcal{T}_R$ denote the subdivided edges opposite the "corner" vertexes $P0$, $Q0$, and $R0$. Informally, traversing any $\mathcal{T}_X$ from the smaller id toward the larger id, we encounter the same sequence of input values[7]. More precisely, let π_P and π_Q be the following permutations:

	P	Q	R
π_P	Q	R	P
π_Q	P	R	Q

The symmetry property is that $\pi_P(\mathcal{T}_R) = \mathcal{T}_P$ and $\pi_Q(\mathcal{T}_R) = \mathcal{T}_Q$.

The anonymous computability theorem guarantees a simplicial map $\mu : \chi^K(\tau(S^2)) \to \mathcal{O}^2$ symmetric under permutation. Assume, without loss of generality, that if P runs solo, it chooses an even name: $\mu(P0) = P0$. Because the protocol is anonymous, $\mu(Q0) = Q0$. The map μ carries the subdivided edge $\chi^K(\mathcal{T}_R)$ to a path linking $P0$ to $Q0$ in $\mathcal{O}^2$. For the sake of this example, let us assume that

$$\mu(\mathcal{T}_R) = (P0, Q1, P1, Q0), \tag{9}$$

as illustrated in Figure 38.

[7] Note that vertex labels in $\mathcal{T}^2$ are not necessarily unique.

Because the protocol is anonymous, μ maps $\mathcal{T}_P$ and $\mathcal{T}_R$ to $\mathcal{O}$ in a symmetric way. More precisely,

$$\begin{aligned} \mu(\mathcal{T}_P) &= \mu(\pi_P(\mathcal{T}_R)) \\ &= \pi_P(\mu(\mathcal{T}_R)). \\ &= \pi_P(P0, Q1, P1, Q0) \\ &= (Q0, R1, Q1, R0). \end{aligned} \tag{10}$$

By similar reasoning,

$$\mu(\mathcal{T}_Q) = (R0, P1, R1, P0). \tag{11}$$

Combining Equations 9, 10, and 11 shows that μ wraps the boundary complex of $\mathcal{T}^2$ around the hole in $\mathcal{O}^2$ twice in the counter-clockwise direction, as illustrated in Figure 39.

No matter what path we choose for $\mu(\mathcal{T}_R)$, we will see that μ wraps the boundary of $\mathcal{T}^2$ around the hole $3k+1$ times, for some integer value k. (A positive k corresponds to a clockwise orientation, and a negative value is counter-clockwise.) In this example, $k = -1$.

Although this example does not constitute a proof, all the key elements of the full proof are represented. The reduced renaming protocol for $n+1$ processes and $2n$ names also has a hole, corresponding to the "missing" all-zero simplex 0^n. (As before, the boundary between $2n$ and $2n+1$ names is the boundary between an output complex with a hole, and one without). We construct an input subcomplex $\mathcal{T}^n$, isomorphic to $\chi(S^n)$, such that input names are symmetric on the boundary of $\mathcal{T}^n$. We formalize the claim that μ cannot wrap the boundary of $\mathcal{T}$ around the hole in $\mathcal{O}^n$ a non-zero number of times. We then exploit the symmetry of $\mathcal{T}^n$ and μ to prove that μ does indeed map the boundary of $\mathcal{T}^n$ around the hole $(n+1)k+1$ times, for some integer value k (again, positive and negative values indicate distinct orientations). The value $(n+1)k+1$ is non-zero for any k, yielding the desired contradiction.

6.5 A Symmetric Input Subcomplex

We now formalize the construction of the "symmetric input subcomplex" $\mathcal{T}^n$ illustrated (for $n = 2$) in Figure 36.

Definition 6.5 *For $0 \le m \le n$, and $0 \le i \le m$, define the permutation π_i^m*

to be

$$\pi_i^m(k) = \begin{cases} k & \text{if } 0 \leq k < i \\ k+1 & \text{if } i \leq k < m \\ i & \text{if } k = m \end{cases}$$

The permutations π_i^n are just the generalizations of π_P and π_Q defined over $\mathcal{T}^2$ in the previous section.

Lemma 6.6 *For $i < j$, $0 \leq k \leq m-2$,*

$$\pi_i^m(\pi_{j-1}^{m-1}(k)) = \pi_j^m(\pi_i^{m-1}(k)).$$

Proof: Both composite permutations map $0, \ldots, m-1$ to $0, \ldots, \hat{\imath}, \ldots, \hat{\jmath}, \ldots, m$, respectively. ∎

The symmetric input subcomplex T^n is isomorphic to the standard chromatic subdivision of an n-simplex by a color-preserving isomorphism denoted by

$$\iota : \chi(S^n) \to \mathcal{T}^n$$

Let $\mathcal{T}_i^{n-1}$ be the subcomplex $\iota(\mathit{face}_i(S^n))$, which we call the *$i$-th face* of $\mathcal{T}^n$. We require that the faces of $\mathcal{T}^n$ satisfy the following symmetry property:

$$\pi_i^n(\mathcal{T}_n^{n-1}) = \mathcal{T}_i^{n-1}. \tag{12}$$

The subcomplex $\mathcal{T}^n$ is constructed inductively. Complex $\mathcal{T}^0$ is the single vertex $\langle P_0, 0\rangle$. Assume inductively that we have constructed T^{m-1}, for $0 < m \leq n$, colored with process ids $P_0, \ldots, P_{m-1}$, satisfying Equation 12, using $m(m+1)/2$ distinct input values. We construct $\mathcal{T}^m$ by assigning input values to $\chi(S^m)$. As an operator, each permutation π_i^m defines an isomorphism

$$\pi_i^m : \mathcal{T}^{m-1} \to \mathcal{T}_i^{m-1}.$$

Assign each vertex in $\mathcal{T}_i^{m-1}$ the value of its matching vertex in $\mathcal{T}^{m-1}$, and assign each central vertex of $\mathcal{T}^m$ labeled with P_i the input value $m(m+1)/2 - m + i = m(m-1)/2 + i$.

To show that this assignment of values is well-defined, we must check that the value assigned to a face vertex does not depend on the choice of π_i^m.

When $m = 0$, all face vertexes are assigned 0, Assume $m > 0$. For permutations π_i^m and π_j^m, where $i < j$,

$$\begin{aligned} \pi_i^m(T^{m-1}) \cap \pi_j^m(T^{m-1}) &= \pi_i^m(T_{j-1}^{m-1}) \\ &= \pi_i^m(\pi_{j-1}^{m-1}(T^{m-2})). \end{aligned}$$

By a similar argument,

$$\pi_i^m(T^{m-1}) \cap \pi_j^m(T^{m-1}) = \pi_j^m(\pi_i^{m-1}(T^{m-2})).$$

By Lemma 6.6, $\pi_i^m(\pi_{j-1}^{m-1}(T^{m-2})) = \pi_j^m(\pi_i^{m-1}(T^{m-2}))$, so the value assigned is independent of π_i.

6.6 Orientation, Cycles, and Boundaries

This section reviews some standard technical definitions needed for our proof. Our discussion closely follows that of Munkres [37, Section 1.13], which the reader is encouraged to consult for more detail.

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 $(\vec{s}_0, \vec{s}_1)$ is just a direction, either from $\vec{s}_0$ to $\vec{s}_1$, or vice-versa. An orientation of a 2-simplex $(\vec{s}_0, \vec{s}_1, \vec{s}_2)$ can either be clockwise, as in $(\vec{s}_0, \vec{s}_1, \vec{s}_2)$, or counterclockwise $(\vec{s}_0, \vec{s}_2, \vec{s}_1)$. 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 $S^d = (\vec{s}_0, \ldots \vec{s}_d)$ be an oriented d-simplex. Define $face_i(S^d)$, the i^{th} *face* of S^d, to be the $(d-1)$-simplex $(\vec{s}_0, \ldots, \hat{s}_i, \ldots, \vec{s}_d)$, 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:

$$\partial_{q-1}\partial_q\alpha = 0. \tag{13}$$

(Henceforth, we 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 13 implies that that every boundary is a cycle.

Definition 6.7 *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* $\phi : C(\mathcal{K}) \to C(\mathcal{L})$ is a family of homomorphisms

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

that preserve cycles and boundaries: $\partial' \circ \phi_q = \phi_{q-1} \circ \partial$. Any subdivision σ of $\mathcal{K}$ induces a chain map in the obvious way,

$$C(\mathcal{K}) \to C(\sigma(\mathcal{K})),$$

as does any simplicial map $\phi : \mathcal{K} \to \mathcal{L}$

$$C(\mathcal{K}) \to C(\mathcal{L}),$$

or permutation operator $\pi : \mathcal{K} \to \mathcal{K}$

$$C(\mathcal{K}) \to C(\mathcal{K}).$$

We use the following facts ([37, 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 6.8 *Every $(n-1)$-cycle of $\mathcal{B}^{n-1}$ is homologous to $k \cdot B$, for some integer k, and every q-cycle is a boundary, for $q < n-1$.*

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

Let $\mathcal{A} \subset \mathcal{K}$ be complexes.

Definition 6.9 *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 6.10 *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 familiar with standard algebraic topology. The q-th simplicial homology group $H_q(\mathcal{K})$ for a complex $\mathcal{K}$ is the quotient group of the q-dimensional group of cycles by the q-dimensional group of boundaries. Lemma 6.8 states the well-known fact 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 6.10 is a special case of the classical theorem [37, P.108] that any complex has the same homology as its deformation retracts.

6.7 Impossibility of Renaming

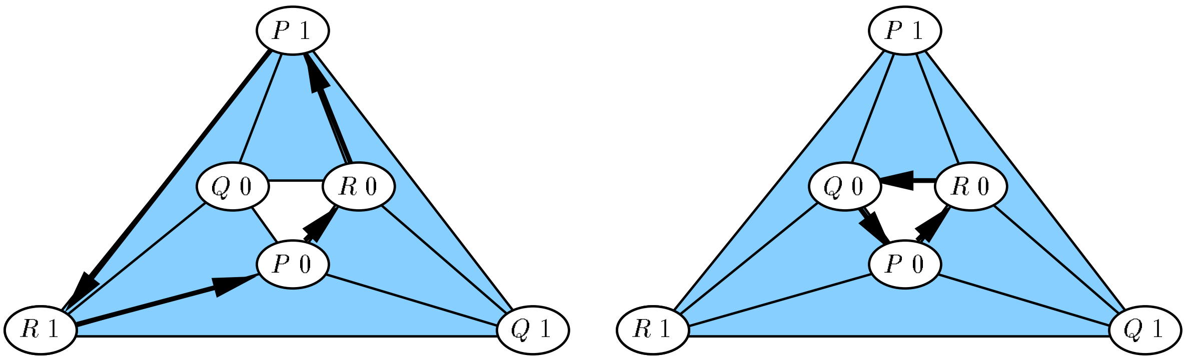

Figure 40: Generators B_Q and $\partial 0^2$

Recall that reduced renaming requires each processes to choose a binary value, but $n+1$ processes may not all choose 1, or all choose 0. The output complex $\mathcal{O}^n$ is thus constructed by taking a binary n-sphere and removing the all-zero simplex 0^n and the all-one simplex 1^n. Let $\dot{0}^{n-1}$ be the boundary complex of 0^n with the orientation induced by the standard orientation of 0^n. This complex is an $(n-1)$-sphere with generator ∂S^n. Because $\dot{0}^{n-1}$ is a deformation retract of $\mathcal{O}^n$, ∂S^n is also a generator for $\mathcal{O}^n$.

Let $\mathcal{O}_i^{n-1}$ be the subcomplex of $\mathcal{O}^n$ colored with process ids $P_0, \ldots, \widehat{P_i}, \ldots, P_n$. This complex 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. Because $\mathcal{O}_i^{n-1}$ is a deformation retract of $\mathcal{O}^n$, B_i is also a generator of $\mathcal{O}^n$. These generators are shown in Figure 40.

Algebraically, these generators are related as follows:

Lemma 6.11 $B_i \sim (-1)^i \partial 0^n$.

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 $\partial 0^n$. ∎

Recall that subdivisions, permutation operators, and simplicial maps all induce chain maps. The standard chromatic subdivision χ induces a chain map

$$C(S^n) \to C(\chi(S^n)).$$

The isomorphism $\iota : \chi(S^n) \to \mathcal{T}^n$ (where $\mathcal{T}^n$ is the symmetric input subcomplex constructed in Section 6.5) induces a chain map

$$C(\chi(S^n)) \to C(\mathcal{T}^n).$$

Assume by way of contradiction that there exists an anonymous protocol for reduced renaming. The anonymous computability theorem (Theorem 6.3) gives a simplicial map $\mu : \chi^K(\mathcal{T}^n) \to \mathcal{O}^n$ inducing chain maps

$$C(\mathcal{T}^n) \to C(\chi^K(\mathcal{T}^n))$$
$$C(\chi^K(\mathcal{T}^n)) \to C(\mathcal{O}^n)$$

Let

$$a : C(S^n) \to C(\mathcal{O}^n)$$

be the result of composing these maps.

The chain ∂S^n is a boundary in $C(S^n)$, and therefore $a(\partial S^n)$ must also be a boundary in $C(\mathcal{O}^n)$. We will exploit the symmetry properties of the subdivisions and maps used to construct a, together with the annulus structure of the reduced renaming output complex $\mathcal{O}^n$, to show that $a(\partial S^n)$ *cannot* be a boundary chain in $C(\mathcal{O}^n)$. As a result of this contradiction, we must conclude that there is no anonymous reduced renaming protocol.

Because $\mathcal{T}^n$ is symmetric on its boundary, and because μ is symmetric under permutation, the chain maps induced by the permutation operators π_i^n, for $0 \le i \le n$, satisfy the following commutative property: for every chain κ of $C(face_n(S^n))$, $\pi_i^n(a(\kappa)) = a(\pi_i^n(\kappa))$. The conventional way to represent such relations is by a *commutative diagram*:

$$\begin{array}{ccc} C(face_n(\mathcal{S}^n)) & \xrightarrow{a} & C(\mathcal{O}_n^{n-1}) \\ \big\downarrow \pi_i^n & & \big\downarrow \pi_i^n \\ C(face_i(\mathcal{S}^n)) & \xrightarrow{a} & C(\mathcal{O}_i^{n-1}) \end{array}$$

Let $\dot{\mathcal{S}}^{n-1}$ be the boundary complex of S^n, the complex consisting of all proper faces of S^n. The color-preserving isomorphism $\dot{\mathcal{S}}^{n-1} \to \dot{0}^{n-1}$ induces a chain map

$$z : C(\dot{\mathcal{S}}^{n-1}) \to C(\mathcal{O}^n).$$

Note that $\partial 0^n = z(\partial S^n)$. This map also commutes with π_i^n:

$$\begin{array}{ccc} C(face_n(\mathcal{S}^n)) & \xrightarrow{z} & C(\mathcal{O}_n^{n-1}) \\ \big\downarrow \pi_i^n & & \big\downarrow \pi_i^n \\ C(face_i(\mathcal{S}^n)) & \xrightarrow{z} & C(\mathcal{O}_i^{n-1}) \end{array}$$

Lemma 6.12 *For $0 \leq i < n$, there is a chain map*

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

such that for every face S^m of S^n, $0 \leq m < n$

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

is an m-cycle of $\mathcal{O}^n$. Moreover, for every $0 \leq m < n$, the following diagram commutes:

$$\begin{array}{ccc} C(face_n(\mathcal{S}^n)) & \xrightarrow{d} & C(\mathcal{O}_n^{n-1}) \\ \big\downarrow \pi_i^n & & \big\downarrow \pi_i^n \\ C(face_i(\mathcal{S}^n)) & \xrightarrow{d} & C(\mathcal{O}_i^{n-1}) \end{array}$$

Proof: We argue by induction on m. When $m = 1$, $ids(S^1) = \{i, j\}$. The 1-chains $a(S^1)$ and $z(S^1)$ have the same 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 6.10, every m-cycle of $\mathcal{O}^n$ is a boundary (for $m < n-1$), so there exists an $(m+1)$-chain $d(S^m)$ such that

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

Because $S^m = \mathit{face}_{m+1}(S^{m+1})$,

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

Applying π_i^n to both sides,

$$\begin{aligned}
&\pi_i^n(\partial d(\mathit{face}_{m+1}(S^{m+1}))) = \\
&\quad \pi_i^n\left(a(\mathit{face}_{m+1}(S^{m+1})) - z(\mathit{face}_{m+1}(S^{m+1})) - d(\partial \mathit{face}_{m+1}(S^{m+1}))\right).
\end{aligned}$$

By the induction hypothesis, π_i^n commutes with d, a, and z, and it commutes with the boundary operator because it is a chain map.

$$\begin{aligned}
&\partial d(\pi_i^n(\mathit{face}_{m+1}(S^{m+1}))) = \\
&\qquad a(\pi_i^n(\mathit{face}_{m+1}(S^{m+1})) - z(\pi_i^n(\mathit{face}_{m+1}(S^{m+1}))) \\
&\qquad\qquad - d(\partial \pi_i^n(\mathit{face}_{m+1}(S^{m+1}))).
\end{aligned}$$

Because $\pi_i^n(\mathit{face}_{m+1}(S^{m+1})) = \mathit{face}_i(S^{m+1}))$,

$$\partial d(\mathit{face}_i(S^{m+1})) = a(\mathit{face}_i(S^{m+1}) - z(\mathit{face}_i(S^{m+1})) - d(\partial \mathit{face}_i(S^{m+1})).$$

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 6.13 *There is no wait-free reduced renaming protocol.*

Proof: As noted above, it suffices to prove that the chain $a(\partial S^n)$ is not a boundary in $C(\mathcal{O}^n)$. By Lemma 6.12,

$$a(\mathit{face}_n(S^{n-1})) - z(\mathit{face}_n(S^{n-1})) - d(\partial(\mathit{face}_n(S^{n-1})))$$

is an $(n-1)$-cycle of $\mathcal{O}^n$. Lemma 6.10 implies that this cycle is homologous to $k \cdot \partial B_n$, for some integer k. Recall that $\pi_i^n(B_n) = B_i$.

$$\pi_i^n\left(a(face_n(S^n)) - z(face_n(S^n)) - d(\partial(face_n(S^n)))\right) \sim k \cdot \pi_i^n(B_n) = k \cdot B_i.$$

Recall that $\pi_i^n(face_n(S^n)) = face_i(S_n)$. Taking the alternating sum over the $(n-1)$-dimensional faces of S^n yields:

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

Because $\partial\partial S^n = 0$, $z(\partial S^n) = 0^n$, and $B_i \sim (-1)^i \partial 0^n$ (Lemma 6.11),

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

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 6.14 *There is no wait-free $(n+1)$-process read/write renaming protocol with $2n$ or fewer names.*

7 Discussion

Since the conference version of this paper appeared, our topological model has yielded a variety of additional results. Herlihy and Rajsbaum [25] consider protocol complexes for protocols that employ more powerful primitives than read/write memory. Chaudhuri, Herlihy, Lynch, and Tuttle [14] give the first tight topology-based lower bounds on set agreement in the synchronous fail-stop model. Attiya and Rajsbaum cast our topological model in an equivalent "combinatorial" representation [5]. Borowsky and Gafni [10, 12] base a key part of their simulation method on our notion of spans. Herlihy and Rajsbaum [25] use homology theory to derive further impossibility results for set agreement, and to unify a variety of known impossibility results in terms of the algebraic theory of chain maps and chain complexes [22]. The impossibility proof for renaming reflects the influence of this paper.

The graph theoretic characterization of Biran, Moran and Zaks[9] also provides an effective procedure for deciding whether a task has a 1-resilient message passing protocol. By contrast, Gafni and Koutsoupias [20] use topological techniques to show that it is undecidable in general whether wait-free read-write tasks have a read-write protocol. Herlihy and Rajsbaum [26] extend these techniques to characterize task decidability in a variety of computational models. Herlihy, Rajsbaum, and Tuttle [27] give a simple round-by-round construction that unifies the synchronous, semi-synchronous, and asynchronous message-passing models of distributed computation within a common formalism based on a topological construction called a *pseudosphere*.

Herlihy and Rajsbaum [23] use topological techniques to give the first complete characterization of the computational power of a non-trivial family of synchronization primitives, encompassing both read-write memory and three-process set agreement.

The topological framework is also of use in modeling complexity. Hoest and Shavit [32] analyze the round complexity of protocols in the iterated immediate snapshot (IIS) model of Borowsky and Gafni. By introducing a novel form of span called the *non-uniform chromatic subdivision*, they refine our topological computability model into a theorem that states that the time complexity of any IIS protocol is directly proportional to the level of non-uniform chromatic subdivisions necessary to allow a simplicial map from a task's input complex to its output complex. In other words, the more you need to subdivide in order for a map to exist, the higher the complexity of your algorithm.

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. We look forward to the day when knowledge of elementary combinatorial and algebraic topology is considered as essential to theoretical computer science as knowledge of graph theory or probability theory.

Acknowledgments

We wish to thank the many people whose comments have helped to improve this paper over the past six years: Yehuda Afek, Hagit Attiya, Elizabeth Borowsky, Faith Fich, Gunnar Hoest, Alan Fekete, Eli Gafni, Nancy Lynch and her students, Shlomo Moran, Lyle Ramshaw, Eric Ruppert, Mark Tuttle, Gideon Stupp, and most of all, Sergio Rajsbaum and John Havlicek.

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] H. Attiya and J. Welch. *Distributed Computing: fundamentals, simulations and advanced topics.* McGraw Hill, London, 1998. ISBN 0-07-7093526.

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

[8] 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.

[9] 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.

[10] E. Borowsky. Capturing the power of resiliency and set consensus in distributed systems. Technical report, University of California Los Angeles, Los Angeles, California, 1995.

[11] 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.

[12] 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.

[13] S. Chaudhuri. Agreement is harder than consensus: Set consensus problems in totally asynchronous systems. In *Proceedings Of The Ninth Annual ACM Symposium On Principles of Distributed Computing*, pages 311–234, August 1990.

[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. Three-processor tasks are undecidable. `http://daphne.cs.ucla.edu/eli/undec.ps`, 1996.

[21] L.C. Glaser. *Geometrical Combinatorial Topology*, volume 1. Van Nostrand Reinhold, New York, 1970.

[22] 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.

[23] M. Herlihy and S. Rajsbaum. A wait-free classification of loop agreement tasks. In *12th International Symposium on Distributed Computing*, September 1998.

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

[25] 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.

[26] M.P. Herlihy and S. Rajsbaum. On the decidability of distributed decision problems. In *29th Annual Symposium on Theory of Computing*, 1997.

[27] M.P. Herlihy, S. Rajsbaum, and M.R. Tuttle. Unifying synchronous and asynchronous message-passing models. In *12th International Symposium on Distributed Computing*, September 1998.

[28] 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.

[29] 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.

[30] 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.

[31] G Hoest. *Towards a Topological Characterization of Asynchronous Complexity*. PhD thesis, Mass. Institute of Technology, Cambridge, MA, September 1997.

[32] G. Hoest and N. Shavit. Towards a topological characterization of asynchronous complexity. In *Proceedings of the 16th Annual ACM Symposium on Principles of Distributed Computing*, pages 199–208, 1997.

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

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

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

[36] 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.

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

[38] 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.

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

A Appendix

A.1 Connectivity

Theorem A.1 *If $\mathcal{A}$ and $\mathcal{B}$ are each n-connected, and $\mathcal{A} \cap \mathcal{B}$ is $(n-1)$-connected, then $\mathcal{A} \cup \mathcal{B}$ is n-connected.*

Proof: Recall that a complex $\mathcal{C}$ is n-connected if its homotopy groups $\pi_1(C), \ldots, \pi_n(\mathcal{C})$ vanish [39].

By induction on n. For the base case, when $n = 1$, this theorem is just the Siefert/Van Kampen theorem [39, p.151]. For the induction step, assume $\mathcal{A}$ and $\mathcal{B}$ are n-connected, and $\mathcal{A} \cap \mathcal{B}$ is $(n-1)$-connected, and $\mathcal{A} \cup \mathcal{B}$ is $(n-1)$-connected. The homology groups $H_n(\mathcal{A})$, $H_n(\mathcal{B})$, and $H_{n-1}(\mathcal{A} \cap \mathcal{B})$ all vanish. and by the Mayer-Vietoris sequence [39, p.186], so does the homology group $H_n(\mathcal{A} \cup \mathcal{B})$. By the Hurewicz Isomorphism Theorem [39, p.394], $\pi_n(\mathcal{A} \cup \mathcal{B})$ must also vanish and therefore $\mathcal{A} \cup \mathcal{B}$ is n-connected. ∎

A.2 Anonymous Computability Theorem

Definition A.2 *The* chromatic K-extension *of a color-preserving simplicial map $\phi : \mathcal{A} \to \mathcal{B}$ is the map $\psi : \chi^K(\mathcal{A}) \to \mathcal{B}$ defined by $\psi(\vec{x}) = \phi(\vec{y})$, where $\vec{y}$ is the unique vertex in $carrier(\vec{x}, \mathcal{A})$ of matching color.*

Lemma A.3 *The chromatic K-extension of any color-preserving simplicial map is a color-preserving simplicial map.*

Proof: Define $\xi : \chi^K(\mathcal{A}) \to \mathcal{A}$ to be the map carrying each $\vec{x}$ to $\vec{y}$, the unique vertex in *carrier*$(\vec{x}, \mathcal{A})$ of matching color. By Lemma 5.27, ξ is a simplicial map. The chromatic K-extension of ϕ is the composition of ϕ and ξ, both color-preserving simplicial maps. ∎

Lemma A.4 *Let $\mathcal{A}$ be a $(p-1)$-sphere, $\mathcal{B}$ a p-disk having $\mathcal{A}$ as boundary, and $\mathcal{C}$ a complex that is $(p-1)$-connected and link-connected. If $\phi : \mathcal{A} \to \mathcal{C}$ is a color-preserving simplicial map, then there exists a color-preserving simplicial map*

$$\psi : \chi^K(\mathcal{B}) \to \mathcal{A}$$

for some $K \geq 0$, such that ψ restricted to $\chi^K(\mathcal{A})$ is the K-chromatic extension of ϕ.

Proof: Lemma 4.21 states that there exists a chromatic subdivision σ and simplicial map

$$\hat{\phi} : \sigma(\mathcal{B}) \to \mathcal{C}$$

such that $\sigma(\mathcal{A}) = \mathcal{A}$, and $\hat{\phi}$ agrees with ϕ on $\mathcal{A}$.

By Theorem 5.29, there exists $K \geq 0$ and a color and carrier-preserving simplicial map

$$\gamma : \chi^K(\mathcal{B}) \to \sigma(\mathcal{B}).$$

The composition of $\hat{\phi}$ and γ yields the desired map ψ. ∎

An *anonymous span* is a color-preserving map $\phi : \chi^K(\mathcal{I}) \to \mathcal{P}$, for some $K \geq 0$, such that ϕ is symmetric under permutation, and for every simplex X in $\chi^K(\mathcal{I})$,

$$\phi(X) \in \mathcal{P}(carrier(X, \chi^K(\mathcal{I}))). \tag{14}$$

Definition A.5 *Let $\mathcal{C}$ be a symmetric complex. Two k-simplexes S_0^k and S_1^k belong to the same k-*orbit *if $S_0^k = \pi(S_1^k)$, for some permutation π.*

The set of k-orbits partition the k-simplexes of $\mathcal{C}$ into equivalence classes.

Lemma A.6 *Every wait-free anonymous protocol complex has an anonymous span.*

Proof: We build up the span inductively. For each $\vec{v} \in \mathcal{I}$, define $\phi_0(\vec{v}) = \mathcal{P}(\vec{v})$, the unique vertex corresponding to a solo execution. This map trivially satisfies Equation 14, and is symmetric under permutation.

Assume inductively that for some $K_{k-1} \geq 0$, we have a color-preserving simplicial map

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

symmetric under permutation, and satisfying Equation 14. Let $S_0^k, \ldots, S_L^k$ be a set of k-simplexes constructed by choosing one k-simplex from each k-orbit of $skel^k(\mathcal{I})$.

For $0 \leq i \leq L$, $\chi^{K_{k-1}}(skel^{k-1}(S_i^k))$ is a $(k-1)$-sphere, and the subdivision of S_i^k constructed by starring $\chi^{K_{k-1}}$ is a k-disk, so by Lemma A.4, for some $L_i \geq 0$,

$$\phi_{k-1} : \chi^{K_{k-1}}(\dot{\mathcal{S}}_i^{k-1}) \rightarrow \mathcal{P}(S_i^k)$$

can be extended to a color-preserving simplicial map

$$\psi : \chi^{K_{k-1}+L_i}(S_i^k) \rightarrow \mathcal{P}(S_i^k),$$

such that ψ restricted to $\chi^{K_{k-1}+L_i}(skel^{k-1}(S_i^k))$ is the L_i-chromatic extension of ϕ_{k-1}. Let $K_k = \max_{i=0}^{L}(K_{k-1} + L_i)$. For $0 \leq i \leq L$, define

$$\psi_i : \chi^{K_k}(S_i^k) \rightarrow \mathcal{P}(S_i^k)$$

to be the L_i-chromatic extension of ψ. The restriction of ψ_i to $skel^{k-1}(S_i^k)$ is the $(K_k - K_{k-1})$-chromatic extension of ϕ_{k-1}, so for every $0 \leq i, j \leq L$, ψ_i and ψ_j agree on the intersection of their domains. Together they define a map

$$\phi_k : \chi^{K_k}(skel^{k-1}(\mathcal{I})) \rightarrow \mathcal{P}$$

satisfying Equation 14. This completes the induction step of the proof. ■

Theorem A.7 (Anonymous Computability Theorem) *A symmetric decision task $\langle \mathcal{I}^n, \mathcal{O}^n, \Delta \rangle$ has a wait-free anonymous protocol using read-write memory if and only if there exists an integer K and a color-preserving simplicial map*

$$\mu : \chi^K(\mathcal{I}) \rightarrow \mathcal{O}$$

symmetric under permutation, such that for each simplex X in $\chi^K(\mathcal{I})$, $\mu(X) \in \Delta(carrier(X, \mathcal{I}))$.

Proof: The "if" part follows immediately from the protocol construction in Section 5.

The "only if" part follows from the existence of the anonymous span guaranteed by Lemma A.6. ■