## Contents

1. **Introduction** 7
   - 1.1 Decision Tasks 8
   - 1.2 Communication 9
   - 1.3 Failures 9
   - 1.4 Timing 10
     - 1.4.1 Processes and Protocols 10
   - 1.5 Chapter Notes 12

2. **Elements of Combinatorial Topology** 13
   - 2.1 Basic Concepts 13
   - 2.2 Distributed Computing 15
   - 2.3 Chapter Notes 16
   - 2.4 Exercises 16

3. **Manifolds, Impossibility, and Separation** 17
   - 3.1 Manifold Complexes 17
   - 3.2 Immediate Snapshots 18
   - 3.3 Manifold Protocols 23
   - 3.4 Set Agreement 25
   - 3.5 Anonymous Protocols 28
   - 3.6 Weak Symmetry-Breaking 29
   - 3.7 Anonymous Set Agreement versus Weak Symmetry Breaking 30
   - 3.8 Chapter Notes 35
   - 3.9 Exercises 35

4. **Connectivity** 37
   - 4.1 Consensus and Path-Connectivity 37
   - 4.2 Consensus in Asynchronous Read-Write Memory 39
   - 4.3 Set Agreement and Connectivity in Higher Dimensions 43
CONTENTS

4.4 Set Agreement and Read-Write memory .................................. 49
  4.4.1 Critical States .............................................. 53
4.5 Chapter Notes ......................................................... 54
4.6 Exercises ............................................................ 54

5 Colorless Tasks .................................................................. 57
  5.1 Pseudospheres ............................................................ 58
  5.2 Wait-Free Read-Write Memory ......................................... 61
    5.2.1 Read-Write Protocols and Pseudospheres ..................... 61
    5.2.2 Necessary and Sufficient Conditions ............................ 62
  5.3 Read-Write Memory with $k$-Set Agreement ......................... 64
  5.4 Decidability ............................................................. 66
    5.4.1 Loop Agreement ................................................. 66
    5.4.2 Read-Write Memory ............................................. 68
    5.4.3 Read-Write Memory ............................................. 68
  5.5 Chapter Notes ........................................................... 69
  5.6 Exercises ............................................................... 70

6 Colorless Tasks .................................................................. 71
  6.1 Definitions ............................................................... 71
  6.2 Map Implies Algorithm ................................................ 73
  6.3 Algorithm Implies Map ................................................. 73
Part I: Undergraduate Course
Chapter 5

Colorless Tasks

For some tasks, we do not care about process identity. Any input legal for one process is legal for the others, and the same is true for their outputs. For example, in the consensus task, it is legal to replace one process’s input with another’s. It is trivially legal to replace outputs because all outputs are the same. The same is true for $k$-set agreement: replacing one process’s input or output with another’s is legal, even though the output values are not necessarily the same.

Since we often refer to process IDs as colors, we call any task where process IDs are not important a colorless task. These tasks are quite common, and, because of their symmetry, they are somewhat easier to analyze than general tasks.

Not all interesting tasks are colorless. For example, in weak symmetry-breaking, it is not legal to replace one process’s output with another’s: if one process decides 0, it is not legal for them all to decide zero.

Here is an informal model for colorless tasks. We will give a more formal model, expressed in terms of conventional task specifications, in the next section. A colorless complex is one whose vertexes are not labeled with process IDs. We mark colorless complexes with “stars” (for example, $\mathcal{C}^*$). A colorless task specification is a triple $(\mathcal{I}^*, \mathcal{O}^*, \Delta^*)$, where $\mathcal{I}^*$ is a colorless input complex, $\mathcal{O}^*$ is a colorless output complex, and $\Delta^*$ is a carrier map from $\mathcal{I}^*$ to $\mathcal{O}^*$. This triple does not mention the number of processes.

Informally, each process starts on a vertex of the colorless input complex $\mathcal{I}^*$. Unlike in regular task specifications, a process is not required to start out on a vertex labeled with its process ID. Instead, the processes can start on any vertex, and multiple processes may start out on the same vertex, as long as all processes start out on a single simplex $\sigma^*$ of $\mathcal{I}^*$. They halt
on the vertexes of a simplex \( \tau \in \Delta^*(\sigma) \), where multiple processes can halt on the same vertex. The problem specification does not restrict the number of processes.

In the next section, we give a formal definition of colorless tasks, expressed in terms of conventional task specifications. Later, we prove some theorems characterizing when they can be solved.

### 5.1 Pseudospheres

Consider the input complex for the 3-process binary consensus task, in which each process starts with a binary value, and all must agree on some process's input. Each process in the set \( \{P, Q, R\} \) can independently be assigned a vertex from the simplex \( \{0,1\} \). The resulting complex is the octahedron shown in Figure 5.1. Each process is assigned 0 at the simplex at the “North Pole”, and each process is assigned 1 at the “South Pole”. The processes are assigned mixed values in the simplexes along the “equator”. We leave it as an exercise to show that the complex constructed by independently assigning binary values to a set of \( n+1 \) processes is homeomorphic to an \( n \)-sphere.

What if we independently assign values from a larger set? Figure ?? shows the complex constructed by independently assigning each process a vertex from the 2-simplex \( \{0,1,2\} \). It consists of three octahedrons, one for each pair of vertexes, where the octahedron generated by \( \{0,1\} \) is “glued” to the octahedron generated by \( \{0,2\} \) at the simplex in which all processes are assigned 0, and similarly for the other values. When we move from assigning two values to three, the resulting complex is no longer homeomorphic to a sphere, but it shares many of the combinatorial properties of spheres. For this reason, we call such a complex a pseudosphere. We are now ready to define pseudospheres formally.

**Definition 5.1.1.** Let \( U \) be a set of \( m+1 \) process IDs and \( \sigma_0, \ldots, \sigma_m \) a sequence of simplexes. The pseudosphere \( \Psi(U, \sigma_0, \ldots, \sigma_m) \) is the following complex:

- Each vertex is a pair \( \langle P_i, \vec{s} \rangle \), where \( P_i \in U \) and \( \vec{s} \in \sigma_i \).
- A set of vertexes \( \{ \langle P_{i_0}, \vec{s}_{i_0} \rangle, \ldots, \langle P_{i_k}, \vec{s}_{i_k} \rangle \} \) define a simplex if the \( P_{i_j} \) are distinct and each \( \vec{s}_{i_j} \in \sigma_{i_j} \), for \( 0 \leq j \leq k \).

For brevity, define \( \Psi(U, \sigma) \) to be \( \Psi(U, \sigma, \ldots, \sigma) \).

It is convenient to extend this definition to complexes.
5.1. PSEUDOSPHERES

Figure 5.1: The pseudosphere complex $\Psi(\{P, Q, R\}, \{0, 1\})$

Figure 5.2: The pseudosphere complex $\Psi(\{P, Q, R\}, \{0, 1, 2\})$
Definition 5.1.2. Let $U$ be a set of $m + 1$ process IDs and $C$ a complex. The pseudosphere $\Psi(U, C)$ is the following complex:

- Each vertex is a pair $\langle P_i, \mathbf{s}_i \rangle$, where $P_i \in U$ and $\mathbf{s}_i$ a vertex of $C$.
- A set of vertexes $\{\langle P_{i_0}, \mathbf{s}_0 \rangle, \ldots, \langle P_{i_k}, \mathbf{s}_k \rangle\}$ define a simplex if the $P_{i_j}$ are distinct and each $\mathbf{s}_j \in \sigma$ for $\sigma \in C$, $0 \leq j \leq k$.

Because any process can start with any input, the input complexes for consensus and $k$-set agreement are typically pseudospheres.

Pseudospheres satisfy the following simple combinatorial properties.

Lemma 5.1.3. If each $\sigma_i$ is a single vertex set $\{\mathbf{s}_i\}$, then

$$\Psi(U; \{\mathbf{s}_0\}, \ldots, \{\mathbf{s}_m\}) \cong \{\mathbf{s}_0, \ldots, \mathbf{s}_m\}.$$ 

Lemma 5.1.4. Let $U = \{P_{i_0}, \ldots, P_{i_m}\}$ and $\sigma_j = \emptyset$.

$$\Psi(U; \sigma_0, \ldots, \sigma_m) \cong \Psi(U \setminus \{P_{i_j}\}, \sigma_0, \ldots, \hat{\sigma}_j, \ldots, \sigma_m), \quad (5.1.1)$$

where circumflex denotes omission.

Pseudospheres are closed under intersection.

Lemma 5.1.5.

$$\Psi(U, \sigma_0, \ldots, \sigma_m) \cap \Psi(U; \tau_0, \ldots, \tau_m) \cong \Psi(U, \sigma_0 \cap \tau_0, \ldots, \sigma_m \cap \tau_m) \quad (5.1.2)$$

Pseudospheres also have nice connectivity properties under wait-free read-write protocols. Finally, define the projection map $\pi : \Psi(U, C) \to C$ to discard process IDs: $\pi(\langle P_i, \mathbf{s} \rangle) = \mathbf{s}$.

We are now ready to give a formal definition for colorless tasks. A complex is colorless if its vertexes are not labeled with process IDs. We will indicate colorless complexes with “stars” ($C^*$). A colorless task specification is a triple $(I^*, O^*, \Delta^*)$, where $I^*$ is a colorless input complex, $O^*$ is a colorless output complex, and $\Delta^*$ is a carrier map from $I^*$ to $O^*$. This triple does not mention the number of processes.

A colorless task defines a family of tasks, $(\Psi(\Pi, I^*), \Psi(\Pi, O^*), \Delta)$, where $\Pi$ is a set of $n + 1$ process IDs. In each such task, the input complex is a pseudosphere over $I^*$, the output complex is a pseudosphere over $O^*$, and $\Delta$ is the carrier map defined to commute with $\pi$ and $\Delta^*$:

$$\pi(\Delta(\sigma)) = \Delta^*(\pi(\sigma))$$
5.2 Wait-Free Read-Write Memory

We will prove a simple theorem that completely characterizes when it is possible to solve a colorless task in wait-free read-write memory. The theorem states that a colorless task \((I^*, O^*, \Delta^*)\) has a wait-free read-write protocol if and only if there is a continuous map between their polyhedrons,

\[ d : |I^*| \to |O^*|, \]

(5.2.1)
carried by \(\Delta^*\). This theorem draws a fundamental connection between topology and computation: a task has a wait-free read-write protocol if and only if the output complex contains a continuous image of the input complex.

5.2.1 Read-Write Protocols and Pseudospheres

Lemma 5.2.1. For every wait-free read-write protocol \(P(\cdot)\), and every sequence \(\sigma_0, \ldots, \sigma_m\) of non-empty simplexes, the protocol complex \(P(\Psi(U; \sigma_0, \ldots, \sigma_n))\) is \((|U| - 2)\)-connected.

Proof. Define the following partial order on sequences of simplexes: \((\sigma_0, \ldots, \sigma_n) \prec (\tau_0, \ldots, \tau_n)\) if each \(\sigma_i \subseteq \tau_i\), and for at least one set, the inclusion is strict.

We proceed by induction on \(|U|\). For the base case, \(|U| = 1\), so \(|U| - 2 = -1\), and the claim follows because \(P(\Psi(\{P_0\} ; \sigma_0))\) is non-empty, hence \((-1)\)-connected.

For the induction step for \(|U|\), assume the claim when \(|U| \leq n\). We next proceed by induction on the set sequence. For the base case, if the \(\sigma_i\) are single vertexes, then \(\Psi(U; \sigma_0, \ldots, \sigma_n)\) is a single \(n\)-simplex, and

\[ P(\Psi(U; \sigma_0, \ldots, \sigma_n)) \cong P(\sigma), \]

which is \((n - 1)\)-connected by Theorem 4.4.10.

For the induction step for the sequences, assume the claim for every sequence less than \(\sigma_0, \ldots, \sigma_n\). There is an index \(i\) such that \(\sigma_i = \{\vec{s}_j| j \in I\}\), where the index set \(I\) has more than one element. Let

\[ K_j = P(\Psi(U; \sigma_0, \ldots, \{\vec{s}_j\}, \ldots, \sigma_n)) \]

The \(K_j\) subcomplexes cover \(P(\Psi(U, \sigma_0, \ldots, \sigma_n))\), and each one is \((n - 1)\)-connected by the induction hypothesis for set sequences.

For any index set \(J\), where \(|J| > 1\), let

\[ L_j = \bigcap_{j \in J} K_j \]
Applying Lemma 5.1.4 and 5.1.5,
\[
L_J = \bigcap_{j \in J} \mathcal{P}(\Psi(U; \sigma_0, \ldots, \{\vec{s}_i\}, \ldots, \sigma_n))
= \mathcal{P}(\bigcap_{j \in J} \Psi(U; \sigma_0, \ldots, \{\vec{s}_i\}, \ldots, \sigma_n))
= \mathcal{P}(\Psi(U \setminus \{P_i\}, \sigma_0, \ldots, \{\vec{s}_i\}, \ldots, \sigma_n)).
\]
This complex is \(\mathcal{P}(\cdot)\) applied to an input complex where \(P_i\) is missing. We can view the absence of \(P_i\) as defining a new \(n\)-process protocol, so by the induction hypothesis for \(n\), each \(L_J\) is \((n - 2)\)-connected. The Nerve Lemma implies that \(\mathcal{P}(\Psi(U, \sigma_0, \ldots, \sigma_n))\) is \((n - 1)\)-connected if and only if the nerve complex \(\mathcal{N}(\mathcal{K}_0, \ldots, \mathcal{K}_n)\) is \((n - 1)\)-connected.

Because all \(L_J\) are non-empty, the nerve of this covering is an \((n - 1)\)-simplex, which is \((n - 2)\)-connected, completing the proof.

Henceforth, if \(\mathcal{P}(\cdot)\) is a wait-free read-write protocol, we use \(\mathcal{P}(I^*)\) has shorthand for \(\mathcal{P}(\Psi(I, I^*))\). Using this compact notation, we can reformulate Lemma 5.2.1 as follows.

**Corollary 5.2.2.** Let \(\mathcal{P}(\cdot)\) be the protocol complex for an \((n + 1)\)-process wait-free read-protocol for the colorless task \((I^*, O^*, \Delta^*)\). For every simplex \(\sigma^* \in I^*\), \(\mathcal{P}(\sigma^*)\) is \((n - 1)\)-connected.

### 5.2.2 Necessary and Sufficient Conditions

We are now ready to prove our necessary and sufficient conditions for solving a colorless task in wait-free read-write memory.

**Lemma 5.2.3.** If there is a wait-free read-write protocol for the colorless task \((I^*, O^*, \Delta^*)\), then there is a continuous map
\[
f : |I^*| \to |O^*|
\]
carried by \(\Delta^*\).

**Proof.** For \(0 \leq d \leq n\), we inductively construct a sequence of continuous maps \(f_d : |\text{skel}^d(I^*)| \to |\mathcal{P}(I^*)|\). For the base case, let \(f_0\) send any vertex \(\vec{v}\) of \(I^*\) to any vertex of \(\mathcal{P}(\{\vec{v}\})\).

For the induction hypothesis, assume we have constructed \(f_{d-1} : |\text{skel}^{d-1}(I^*)| \to |\mathcal{P}(I^*)|\). This map sends the \((d - 1)\)-skeleton of each uncolored \(d\)-simplex \(\sigma^*\) in \(\text{skel}^d(I^*)\) to \(\mathcal{P}(\sigma^*) \subset \mathcal{P}(I^*)\). By Corollary 5.2.2, \(\mathcal{P}(\sigma^*)\) is \((n - 1)\)-connected, so any map of the \((d - 1)\)-sphere (namely, the \((d - 1)\)-skeleton of
σ, can be extended to a continuous map of the $d$-disk (namely, $\sigma$), yielding a continuous map

$$f_d : |\text{skel}^d(I^*)| \to |P(I^*)|.$$  

The decision map $\delta : P(I^*) \to O^*$ induces a continuous map $|\delta| : |P(I^*)| \to |O^*|$ carried by $\Delta^*$. The map $f$ is the composition of $f_n$ and $|\delta|$.

$$|I^*| \xrightarrow{f_n} |P(I)| \xrightarrow{|\delta|} O^*.$$  

For the other direction, we must construct a protocol given a continuous map on the polyhedrons. The following building-block task will come in useful. In the barycentric agreement task $(I^*, \text{Bary} I^*, \text{Bary})$, the processes start on the vertexes of a simplex $\sigma$ in the colorless complex $I^*$. (Multiple processes may start on the same vertex.) The processes halt on the vertex of a simplex in the (colorless) barycentric subdivision $\text{Bary} \sigma$. (Multiple processes may halt on the same vertex.) Any protocol for this task can be repeated to solve the iterated barycentric agreement task $(I^*, \text{Bary}^N I^*, \text{Bary}^N)$, for $N > 0$.

Using only atomic reads and writes, it is possible to implement an atomic snapshot protocol that returns an instantaneous “snapshot” of the entire memory (see the chapter notes). We will assume we have such an algorithm as a “black box”. Here is how to implement barycentric agreement in wait-free read-write memory. The processes share an array $a$.

1. Each $P_i$ writes its input vertex to $a[i]$
2. Each $P_i$ takes a snapshot of the array, yielding a face $\sigma'$ of the input simplex $\sigma$.
3. $P_i$ decides $\sigma'$, interpreted as a vertex of $\text{Bary} \sigma$.

Because snapshots are instantaneous, successive snapshots are ordered by inclusion, implying that all views produced by snapshots are valid simplexes of the barycentric subdivision.

Lemma 5.2.4. If there exists a continuous map

$$f : |I^*| \to |O^*|$$

carried by $\Delta$, then there is a wait-free read-write protocol for the colorless task $(I^*, O^*, \Delta^*)$.  

□
Proof. We are given a continuous map

\[ f : |I^*| \rightarrow |O^*| \]

carried by \( \Delta^* \). This map has a simplicial approximation

\[ \phi : \text{Bary}^N I^* \rightarrow O^* \]

also carried by \( \Delta^* \). The protocol can be broken up into the following steps.

1. Use iterated barycentric agreement to solve the task \((I^*, \text{Bary}^N I^*, \text{Bary}^N)\).

2. Apply \( \phi \) to the vertex computed in the previous step.

The result is a protocol that solves the colorless task \((I^*, O^*, \Delta^*)\).

\[ \square \]

Lemmas \([5.2.3]\) and \([5.2.4]\) imply the following theorem.

**Theorem 5.2.5.** The colorless task \((I^*, O^*, \Delta^*)\) has a wait-free read-write protocol if and only if there exists a continuous map \( f : |I^*| \rightarrow |O^*| \) carried by \( \Delta^* \).

### 5.3 Read-Write Memory with \( k \)-Set Agreement

Practically all modern multiprocessor architectures provide synchronization primitives more powerful than simple read or write instructions. For example, the *test-and-set* instruction atomically swaps the value *true* for the contents of a memory location. For example, if we augment read-write memory with *test-and-set*, it is possible to solve \( k \)-set agreement for \( k = \lceil \frac{n+1}{2} \rceil \) (see Exercise \([5.2]\)).

In this section we generalize the previous section’s necessary and sufficient results to any architecture that provides additional synchronization primitives powerful enough to solve \( k \)-set agreement, for \( k \leq n \).

**Lemma 5.3.1.** If there is a wait-free protocol for the colorless task \((I^*, O^*, \Delta^*)\), using read-write memory and \( k \)-set agreement objects, then there is a continuous map

\[ f : |\text{skel}^{k-1} I^*| \rightarrow |O^*| \]

carried by \( \Delta^* \).
5.3. READ-WRITE MEMORY WITH K-SET AGREEMENT

**Proof.** The specification for k-set agreement objects is non-deterministic. Any protocol that uses “black-box” k-set agreement objects must continue to work if we restrict the non-determinism of the k-set agreement objects. It is convenient to consider the following simple restriction: if k or fewer processes participate in the k-set agreement protocol, each process decides its own input.

As a result, a wait-free k-process read-write protocol augmented by these restricted k-set agreement objects is indistinguishable from a wait-free k-process read-write protocol. More precisely, if \(|U| = k\), then \(\mathcal{P}(\Psi(U, I^*))\) is \((k - 2)\)-connected.

Following the proof of Lemma 5.2.3, for \(0 \leq d \leq n\), we inductively construct a sequence of continuous maps \(f_d : |\text{skel}^d(I^*)| \rightarrow |\mathcal{P}(I^*)|\), for \(0 \leq d < k\). For the base case, let \(f_0\) send any vertex \(\vec{v}\) of \(I^*\) to any vertex of \(\mathcal{P}(\{\vec{v}\})\).

For the induction hypothesis, assume we have constructed \(f_{d-1} : |\text{skel}^{d-1}(I^*)| \rightarrow |\mathcal{P}(I^*)|\). This map sends the \((d-1)\)-skeleton of each uncolored \(d\)-simplex \(\sigma^*\) in \(\text{skel}^d(I^*)\) to \(\mathcal{P}(\sigma^*) \subset \mathcal{P}(I^*)\). If \(d < k\), then \(\mathcal{P}(\sigma^*)\) is \((n-1)\)-connected, so any map of the \((d-1)\)-sphere (namely, the \((d-1)\)-skeleton of \(\sigma\), can be extended to a continuous map of the \(d\)-disk (namely, \(\sigma\)), yielding a continuous map

\[
 f_d : |\text{skel}^d(I^*)| \rightarrow |\mathcal{P}(I^*)|.
\]

The decision map \(\delta : \mathcal{P}(I^*) \rightarrow O^*\) induces a continuous map \(|\delta| : |\mathcal{P}(I^*)| \rightarrow |O^*|\) carried by \(\Delta^*\). The map \(d\) is the composition of \(f_{k-1}\) and \(|\delta|\).

\[
 |I^*| \xrightarrow{f_{k-1}} |\mathcal{P}(\text{skel}^{k-1} I^*)| \xrightarrow{|\delta|} O^*.
\]

Lemma 5.3.2. If there exists a continuous map

\[
 f : |\text{skel}^{k-1} I^*| \rightarrow |O^*|
\]

carried by \(\Delta\), then there is a wait-free protocol using read-write memory and k-set agreement for the colorless task \((I^*, O^*, \Delta^*)\).

**Proof.** The map \(f\) has a simplicial approximation

\[
 \phi : \text{Bary}^N \text{skel}^{k-1} I^* \rightarrow O^*
\]

also carried by \(\Delta^*\). The protocol can be broken up into the following steps.
1. Use the \( k \)-set agreement protocol to solve the task \((I^*, \text{skel}^{k-1} I^*, \text{skel}^{k-1})\), then

2. Solve the task \((\text{skel}^{k-1} I^*, \text{Bary}^N \text{skel}^{k-1} I^*, \text{Bary}^N)\) using \( N \) iterations of the barycentric agreement protocol.

3. Apply \( \phi \) to the vertex computed in the previous step.

\[\square\]

Lemmas 5.3.1 and ?? imply the following theorem.

**Theorem 5.3.3.** The colorless task \((I^*, O^*, \Delta^*)\) has a protocol using wait-free read-write memory augmented by \( k \)-set agreement objects if and only if there exists a continuous map \( f : |\text{skel}^{k-1} I^*| \rightarrow |O^*| \) carried by \( \Delta^* \).

### 5.4 Decidability

Now that we have necessary and sufficient conditions for a task to be solvable in the wait-free read-write model, it seems natural to ask whether we can automate the process of deciding whether a given task has a protocol. Can we write a program (that is, a Turing machine) that takes a task description as input, and returns to us a Boolean indicating whether a protocol exists? The answer depends on the model of computation. For wait-free read-write memory, or wait-free read-write memory augmented with \( k \)-set agreement objects where \( k > 2 \), the answer is **no**: there exists a family of tasks for which it is **undecidable** whether a protocol exists. For wait-free read-write memory augmented with either 1-set or 2-set agreement, however, the answer is **yes**: for every colorless task, it is **undecidable** whether a protocol exists.

#### 5.4.1 Loop Agreement

We will argue by reduction to the **contractibility** problem. Let \( \mathcal{K} \) be a finite (uncolored) 2-dimensional simplicial complex. An **edge loop** \( \kappa \) is the complex given by an alternating sequence of vertexes and edges \( \bar{v}_0, e_0, \bar{v}_1, e_1, \ldots, \bar{v}_\ell \) such that edge \( e_i = \{ \bar{v}_i, \bar{v}_{i+1} \} \), and \( \bar{v}_\ell = \bar{v}_0 \).

It is sometimes convenient to think of a loop as a continuous map from the unit interval \( I = [0, 1] \) to \( |\mathcal{K}| \):

\[\kappa : I \rightarrow |\mathcal{K}|\]
such that $\kappa(0) = \kappa(1) = k_0$. A loop is contractible if it can be continuously deformed to the single vertex $\vec{v}_0$, while leaving that point fixed. More precisely, there exists a continuous map $h : I \times I \rightarrow |C|$, called a homotopy, such that $h(s, 0) = \kappa(s)$, $h(s, 1) = \vec{v}_0$, and $h(0, t) = h(1, t) = \vec{v}_0$ for all $t \in [0, 1]$.

Yet another way to think about edge loops is to consider $\kappa$ to be a continuous map

$$|\kappa| : |\partial \sigma^2| \rightarrow |\mathcal{K}|,$$

where $\sigma^2$ is a 2-simplex.

**Lemma 5.4.1 ([20] 1.3.12).** The following two conditions are equivalent:

1. $|\kappa(\partial \sigma^2)|$ is contractible, and
2. $|\kappa|$ can be extended to a continuous map $f : |\sigma^2| \to |\mathcal{K}|$.

The question whether a loop in a finite simplicial complex is contractible is known to be undecidable.

In a loop agreement task $(\sigma^2, \mathcal{K}, \Delta_\kappa)$, $\sigma^2$ is a complex consisting of a single 2-simplex and its faces, and $\mathcal{K}$ is a finite (uncolored) 2-dimensional simplicial complex. The task is characterized by an edge loop $\kappa$, and three distinct vertexes $\vec{k}_0, \vec{k}_1, \vec{k}_2$ in $\kappa$. For distinct $i, j$, and $k$, let $\kappa_{ij}$ be the sub-path of $\kappa$ linking $\vec{k}_i$ to $\vec{k}_j$ without passing through $\vec{k}_k$. In this task, the processes converge on a simplex in $\mathcal{K}$. If all processes start on a single vertex $\vec{s}_i$, they converge on the corresponding vertex $\vec{k}_i$. If they start on two distinct input vertexes, $\vec{s}_i$ and $\vec{s}_j$, they converge on some simplex along the path linking $\vec{k}_i$ and $\vec{k}_j$. Finally, if the processes start on all three input vertexes, they converge to any simplex of $\mathcal{K}$. Formally, for each input simplex $\tau \subseteq \sigma^2$,

$$\Delta_\kappa(\tau) = \begin{cases} \vec{k}_i & \text{if } \tau = \{\vec{s}_i\}, \\ \kappa_{ij} & \text{if } \tau = \{\vec{s}_i, \vec{s}_j\}, i < j, \text{ and} \\ \mathcal{C} & \text{if } \tau = \sigma = \{\vec{s}_0, \vec{s}_1, \vec{s}_2\}. \end{cases}$$

### 5.4.2 Read-Write Memory

Theorem 5.4.2. The loop agreement task $(\sigma^2, \mathcal{K}, \Delta_\kappa)$ has a protocol in the wait-free read-write model if and only if $\kappa$ is contractible.

Proof. By Theorem 5.2.5, $(\sigma^2, \mathcal{K}, \Delta_\kappa)$ has a wait-free read-write protocol if and only if there exists a continuous map $f : |\sigma^2| \to |\mathcal{K}|$ carried by $\Delta_\kappa$.

If $\kappa$ is contractible, then by Lemma 5.4.1 there is a continuous map $f : |\sigma^2| \to |\mathcal{K}|$ carried by $\Delta_\kappa$, and hence a protocol.

If there exists a protocol, then there is a decision map $\delta : \text{Bary}^N \sigma^2 \to \mathcal{K}$ sending $\partial \sigma^2$ to the loop $\kappa$. The simplicial decision map $\delta$ induces a piece-wise linear map $|\delta| : |\text{Bary}^N \sigma^2| \to |\mathcal{K}|$. By Lemma 5.4.1 $\kappa$ is contractible. \qed

A loop agreement task has a solution in the wait-free read-write model if and only if that loop is contractible. But loop contractibility is undecidable, and therefore so is the question whether a loop agreement task has a protocol in this model.

### 5.4.3 Read-Write Memory

When $k > 2$, this result generalizes immediately to read-write memory augmented by $k$-set agreement objects. As in the proof of Lemma 5.3.1, we
can assume without loss of generality that if a \( k \)-set agreement object is accessed by only 3 processes, then each process decides its own input, and the resulting protocol complex is indistinguishable from a read-write protocol complex.

**Corollary 5.4.3.** The loop agreement task \( (\sigma^2, \mathcal{K}, \Delta_\kappa) \) has a protocol in the wait-free read-write model if and only if \( \kappa \) is contractible.

If follows that whether a loop agreement task has a solution for 3 processes in this model is undecidable.

**Theorem 5.4.4.** If \( k \leq 2 \), it is decidable whether whether a colorless task has a wait-free protocol using read-write memory augmented by \( k \)-set agreement objects.

**Proof.** By Theorem 5.3.3, the colorless task \( (\mathcal{I}^*, \mathcal{O}^*, \Delta^*) \) has a protocol in this model if and only if there exists a continuous map \( f : |\text{skel}^{k-1}\mathcal{I}^*| \rightarrow |\mathcal{O}^*| \) carried by \( \Delta^* \).

When \( k = 1 \) this map exists if and only if \( \mathcal{P}(\bar{v}) \) is non-empty for each \( \bar{v} \in \mathcal{I}^* \). Given a complex, it is decidable whether it is non-empty.

When \( k = 2 \), this map exists if and only if \( \mathcal{P}(\sigma^*) \) is path-connected for each 2-simplex \( \sigma^* \). Given a complex, it is decidable whether it is path-connected.

**Applications** Given an adversary \( \mathcal{A} \) with minimum core size \( c \), whether \( \mathcal{A} \) permits a solution to an arbitrary colorless task is decidable if only if \( c \leq 2 \). Establishing the condition of Lemma ?? requires testing for path-connectivity if \( c \leq 2 \), which is decidable [3]. For \( c = 3 \), it requires testing for simple connectivity, which is undecidable [8, 10].

### 5.5 Chapter Notes

The atomic snapshot algorithm is due to Afek, Attiya, Dolev, Gafni, Merritt, and Shavit [1].

The pseudosphere construct is due to Herlihy, Rajsbaum, and Tuttle [11].

Gafni and Koutsoupias [8] were the first to note that 3-processor tasks are undecidable in the wait-free read-write model. This observation was generalized to other models by Herlihy and Rajsbaum [10]. Contractibility is undecidable because it reduces to the word problem for finitely-presented groups, (whether an expression reduces to the unit element). This problem was shown to be undecidable by S.P. Novikov in 1955, and the isomorphism
problem (whether two such groups are isomorphic) was shown to be undecidable by M.O. Rabin in 1958. Contractibility requires deciding whether a loop $\lambda$ in a finite simplicial complex $\mathcal{K}$ can be continuously deformed to a point. From a finite description of $\mathcal{K}$ one can construct a finite presentation of its fundamental group. Any finitely-presented group can be realized as the fundamental group of some complex. A loop $\lambda$ is contractible in $\mathcal{K}$ if and only if $\lambda$ represents the identity element of the fundamental group of $\mathcal{K}$. It follows that deciding contractibility is equivalent to deciding the word problem for finitely-presented groups, which is known to be undecidable. (For a more complete discussion of these problems, see Stillwell [21] or Sergeraert [19].)

5.6 Exercises

Exercise 5.1. Show that the complex constructed by independently assigning binary values to a set of $n + 1$ processes is homeomorphic to an $n$-sphere. (Hint: argue by induction on $n$.)

Exercise 5.2. Give an $(n + 1)$-process protocol for solving $\lceil \frac{n+1}{2} \rceil$-set agreement using read-write memory and test-and-set instructions.
Bibliography


