Universiteit Utrecht

Faculteit Bètawetenschappen

Determining the solvability of colorless tasks using combinatorial topology

Bachelor Thesis

Max Meijer

Wiskunde

Supervisor:

Dr. F.L.M. Meier
Mathematical Institute

28 May 2019

Contents

1 Introduction

In this thesis, we will see how topology can be applied within the context of distributed computing. Distributed computing is the study of how concurrent computing systems can work together so that the system as a whole displays the proper behaviour. An example where this is particularly important is banking applications that have servers all over the world that communicate through the internet. Communication over the internet is not the only application. At the micro level, most computers provide ways to run multiple processes concurrently. It is also important to consider what happens if a subset of the computing systems fail. Preferable, we would have a system that keeps working correctly even if other systems fail.

To model these complex systems, we can use abstract simplicial complexes, a combinatorial tool, that has an important link to a topological version, namely geometric simplicial complexes. Establishing this connection between distributed computing and topology means that results from topology may have applications in distributed systems in addition to being useful in topology itself. Although any result we find in distributed computing could theoretically be proven without using topology, the fact that topological results can be translated might avoid double work from being done.

A good overview of the link between distributed computing and topology is provided in [4]. We will focus on one particular area, namely the solvability of colorless tasks in systems where crashes are undetectable. In some ways, colorless tasks are the most fundamental tasks in distributed computing. The consensus problem, one of the most important problems in distributed computing, can be modeled as a colorless task. Other examples include k-set agreement, approximate agreement and commit-adopt.

In Section 2, we will introduce some concepts from distributed computing and we will discuss how we can model them using functions. In section 3 we will define abstract simplicial complexes, which we will be able to use in section 4 in defining a more powerful distributed computing model. In section 5, geometric simplicial complexes are introduced together with how they connect abstract simplicial complexes to topology. In section 6, we will provide some examples of colorless tasks. Finally, we will show in section 7 that the solvability of colorless tasks is linked to the existence of continuous functions between topological spaces.

2 Colorless tasks

2.1 Programs, crashes, and colorless tasks

In this section, the concepts of multi-process programs and colorless tasks are introduced. We start by defining what we mean by processes and programs. For a complete operational model that fits these definitions, see [4], Chapter 4.

Let $p \geqslant 0$ and $n \geqslant 1$ be integers. An n-process **program** consists of n processes. A **process** is given inputs from a set I and after its execution it returns output values in a set O. Note that a process does not need to be deterministic. That is, it might return different output values in different executions even if it is given the same input values during those executions.

A process may **crash** during its execution (due to external factors, like hardware failure). After crashing it will not be able to communicate with other processes and it will not return any output value. We assume a crashed process to be indistinguishable from a process that is slow to respond so that a crash cannot be detected by other processes.

At the start of an **execution** of a program, the processes are each given input values in I. During an execution, the processes may communicate with each other. During the execution of a p-**resilient** program, at most p processes crash and the non-crashing processes return an output value. An execution of a program thus induces both an input value and an output value for each non-crashing process.

For an execution of the program P where the *set* of input values is given by $\sigma \subseteq I$ and the *set* of output values by $\tau \subseteq O$, we say that it is an execution of P with **colorless input** σ and **colorless output** τ. Note that when we speak of a *colorless* input or output we only look at the *set* of input values or the *set* of output values, not at the way the input values are distributed among the processes or at which processes return which output values.

The property of programs that we will be most interested in is described by the following definition in which we write 2^S for the set of all subsets of a set S.

Definition 1 (Colorless execution map)**.** Let I be a set of input values and O a set of output values. Assume that $\mathcal{I}$ is a non-empty family of finite subsets of I and that $\mathcal{O}$ is a non-empty family of finite subsets of O. Let P be a program for which all executions have colorless inputs from $\mathcal{I}$ and colorless outputs in $\mathcal{O}$. The **colorless execution map** of P is a function $\mathbf{P} : \mathcal{I} \to 2^{\mathcal{O}}$, mapping each colorless input σ to the set of colorless outputs that can be the result of an execution of P with colorless input σ.

We are interested in whether specific problems are solvable by programs. That is, we specify what a program needs to do and then check whether this can be achieved within the operational constraints. Defining these problems is possible through the notion of a colorless task, which is based on [4], Definition 4.1.2.

Definition 2 (Colorless task)**.** Let I be a set of input values and O a set of output values. A **colorless task** is a tuple $(\mathcal{I}, \mathcal{O}, \Delta)$, where:

(i) $\mathcal{I}$ is a non-empty family of finite subsets of I

(ii) $\mathcal{O}$ is a non-empty family of finite subsets of O.

(iii) $\Delta : \mathcal{I} \to 2^{\mathcal{O}}$ is a function that maps each colorless input in $\mathcal{I}$ to a corresponding non-empty set of allowed colorless outputs as a subset of $\mathcal{O}$.

A program P with input values from I, output values in O and execution map $\mathbf{P}$ is said to solve the colorless task $(\mathcal{I}, \mathcal{O}, \Delta)$ if and only if $\mathbf{P}(\sigma) \subseteq \Delta(\sigma)$ for all $\sigma \in \mathcal{I}$. In other words, all executions of P with colorless input $\sigma \in \mathcal{I}$ have a colorless output that is an element of $\Delta(\sigma)$.

Note that we don't specify beforehand which process gets which input value from the colorless input. The program must return a correct output for all the possible ways that the input values can be distributed over the processes. On the other hand, it doesn't matter which process returns which output value as long as the set of output values is allowed by Δ for the given colorless input.

We will now turn to an example of a colorless task that is considered a fundamental problem in distributed computing.

Example 3 (Binary consensus problem)**.** Suppose we have two processes A and B that accept input values from $\{0, 1\}$ and return output values in $\{0, 1\}$. In the consensus problem we want A and B to both output one of the values that they received as input values although we set one important additional restriction: both processes must return the same output value. That is, they must achieve consensus. So if both processes receive 0 as an input value then they should both return 0 and if they both receive 1 as an input value then they should both return 1. If they receive different input values (that is, one of them gets 0 and the other gets 1), then it doesn't matter whether they return 0 or 1 as long as they return the *same* output value.

We can represent this problem as a colorless task. Let $I = \{0, 1\}$ and $O = \{0, 1\}$. Define $\mathcal{I} = \{\{0\}, \{1\}, \{0, 1\}\}$ and $\mathcal{O} = \{\{0\}, \{1\}\}$. Note that $\{0, 1\}$ is not an element of $\mathcal{O}$. This represents the restriction that both processes must return the same output value. The map $\Delta : \mathcal{I} \to \mathcal{O}$ is given by $\Delta(\{0\}) = \{\{0\}\}$, $\Delta(\{1\}) = \{\{1\}\}$ and $\Delta(\{0, 1\}) = \{\{0\}, \{1\}\} = \mathcal{O}$. Note that the definitions of $\Delta(\{0\})$ and $\Delta(\{1\})$ imply that the processes can only return values that at least one process has received as an input.

In this thesis, we will focus on n-process programs that are $(n-1)$-resilient. An n-process program that is $(n-1)$-resilient is also called **wait-free**. This stems from the fact that $(n-1)$-resilient programs can never wait for a message from another process because the others might all have crashed (recall that we assume that there is no way for a process to distinguish between a crashed and a slow process).

2.2 States and protocols

In this section, we will assume that processes take input values in I, return output values in O and that programs are given colorless inputs in $\mathcal{I}$ and colorless outputs in $\mathcal{O}$.

At every point in the execution of a process, a process has a certain **state**. This state contains all the information that the process has about the outside world and in particular what it knows other processes have sent it. At the start of its execution, a process only knows what input was given to it so the set of possible initial states is isomorphic to I.

In practice, a process might have to discard some potentially useful state information during an execution. For example, there might not be enough memory to store the information, or interpreting it might take too much time. In this thesis, however, we will assume that the processes do not forget anything. That is, each state contains more information than each state before it.

There are various factors that may cause a program to run differently even if it is given the same input. The processes might rely on (pseudo-)random values, communication between processes might get delayed or be dropped entirely, or processes might crash. Because we are discussing

colorless tasks, some additional 'indeterminism' is caused by how the input values from the colorless input are distributed across the processes.

At the end of an execution, a process has to return an output value. The state just before it returns an output value is called its **final state** and it contains all the information from the previous states. We let S represent the set of final states of the process. For each input value in I, a certain set of final states is possible for a process. This can be represented by a map $f : I \to 2^S$. A process's output value can be determined from its final state. This can be represented by a map $g : S \to O$. Note that some information may get lost in this step because multiple different final states might result in the same output value. Essentially we have split the execution of a process into two phases: the gathering of information and the loss of information. The gathering of information in distributed computing is mainly done by communicating with other processes and the loss of information is deciding on which output value to return based on what happened during the first phase.

We can colorlessly extend our reasoning about processes to programs consisting of multiple processes. The **colorless state** of a program at a point in time consists of the set of states of its processes. Note that in a colorless state it is not specified which process is assigned which state. It is only specified what the *set* of the states of all the processes in the program is. We will assume that different processes in the same program will behave the same when they are in the same state. In particular, this means that the program's colorless output is only dependent on its final colorless state.

Let $\mathcal{P}$ be the set of possible final colorless states. Just like a process has a map from each possible input value to the set of possible final states, we can define $\Xi : \mathcal{I} \to 2^{\mathcal{P}}$ as a function that assigns to each colorless input the set of possible final colorless states. A program's colorless output can then be derived from its final colorless state, which gives us a function $\delta : \mathcal{P} \to \mathcal{O}$. Note that the composition $\delta \circ \Xi$ gives us the execution map $\mathbf{P} : \mathcal{I} \to 2^{\mathcal{O}}$. Operationally, the decomposition of the pseudocode can be represented by the following C++-like pseudocode:

```
Value executeProgram(Value input) {
    // the final state is reached by executing the protocol
    // the protocol receives the input as its initial state
    Value finalState = executeProtocol(input);

    // the decision is made based on the finalState
    // no communication with other processes is made here
    return makeDecision(finalState);
}
```

In distributed computing, the information gathering part of the execution usually consists of the communication between processes. This is why we refer to this part as the **protocol** of the program and why $\mathcal{P}$ is called the **protocol complex**.

Definition 4. A **colorless protocol** is a tuple $(\mathcal{I}, \mathcal{P}, \Xi)$ where:

- $\mathcal{I}$ is the set of possible colorless inputs
- $\mathcal{P}$ is the set of possible final colorless states
- $\Xi : \mathcal{I} \to 2^{\mathcal{P}}$ maps each colorless input σ to the set of final colorless states of executions with colorless input σ

A **colorless decision map** is a function $\delta : \mathcal{P} \to \mathcal{O}$ that maps each colorless final state to a colorless output. A program with protocol $(\mathcal{I}, \mathcal{P}, \Xi)$, and colorless decision map δ has a colorless execution map $\mathbf{P} : \mathcal{I} \to \mathcal{O}$ where for each input $\sigma \in \mathcal{I}$, we have that $\mathbf{P}(\sigma) = \delta(\Xi(\sigma))$.

We say that a colorless protocol $(\mathcal{I}, \mathcal{P}, \Xi)$ **solves** a colorless task $(\mathcal{I}, \mathcal{O}, \Delta)$ if and only if there exists a program with colorless protocol $(\mathcal{I}, \mathcal{P}, \Xi)$ that solves $(\mathcal{I}, \mathcal{O}, \Delta)$. Equivalently, Ξ solves $(\mathcal{I}, \mathcal{O}, \Delta)$ if and only if there exists a colorless decision map δ such that $\delta(\Xi(\sigma)) \subseteq \Delta(\sigma)$ for all $\sigma \in \mathcal{I}$.

An **operational model** defines the way processes are executed and how they can communicate. In particular, this defines which protocols are possible and which are not. In case a protocol is possible in an operational model, we will say that it exists in that operational model. Usually, the operational model is clear from the context and we will just speak about whether the protocol exists. We will assume that there are no restrictions on the decision map. This makes sense as the information that is required for determining the output value should be contained within the final colorless state.

Example 5. Our discussion of operational models has been rather abstract so far. Various different operational models can be defined. This example illustrates a few different concrete possibilities that could be interesting in the context of distributed computing.

1. The processes are Turing machines where each action can take an arbitrarily large amount of time. These processes can access some public memory that is accessible to all processes and there is an unbounded amount of private memory for each process.

2. Each process is an application running on a separate computer. There are cables directly connecting each pair of computers. The computers can pass messages over these cables.

2.3 Colorless single-layer atomic snapshot protocols

In this section, we introduce the colorless single-layer atomic snapshot protocols. This is based on the single-layer atomic snapshot operational model. In this model there is some shared memory, which they can all read from. It is partitioned into parts such that each part is only writable by one process. Each process has a part that it can write to.

Atomic reads and writes If a process writes to the memory, then it does so in an atomic way. The process changes the entire contents of its layer at a single point in time. For example, if it were to change the value 00 to 11 and some other process were to read this memory at the same time then it would either read 00 or 11, but it would never read 01 or 10.

Atomic snapshots It is possible for processes to make an **atomic snapshot** of a layer. This is a private copy of the contents of the entire layer at a point in time during the execution of the snapshot. This remains true even if multiple other processes change the contents of their part of the layer during the taking of the snapshot.

It is well-known that it is in fact possible to implement atomic snapshots using atomic reads and atomic writes. An explanation of this can be found in [4], Chapter 14. Atomic snapshots are sometimes referred to as **scan**s. As the memory can only be written to by one process, an algorithm that takes an atomic snapshot is called a **single-writer multiple-reader algorithm**. An implementation of such an algorithm was published for the first time in [1].

In pseudocode, we represent the memory as an array of Values where each index i is only writable for process i. We will refer to the type of the data stored in the array by the keyword Value. A **colorless single-layer atomic snapshot protocol** (or **colorless single-layer protocol** for short when no confusion arrises) is a protocol that is run on the model described above, using the following C++-like pseudocode:

```
// this constant is the same for all the processes
const int numProcesses;

// this is different for each process
// it is a value between 0 and numProcesses - 1
const int currentProcessIndex;

// this value points to the same memory location
// for all processes
extern Value[numProcesses] memory;

// the colorless single-layer atomic snapshot protocol
Value executeColorlessSingleLayerProtocol ( Value input ) {
    // at the start, the only information we have is the input
    Value state = input;

    // write our view to our part of the memory
    update(memory[currentProcessIndex], state);

    // take an atomic snapshot of the memory
    Value[numProcesses] snapshot = scan(memory);

    // update our own view based on the snapshot
    // note that this contains our previous state
    // since we wrote it to memory before taking a snapshot
    // N.B. we only use the Set of Values
    state = setOfValues(snapshot);

    return state;
}
```

Colorless After the taking the snapshot, we only store the *set* of Values, not which value belongs to which process. This is why we call the protocol colorless. According to [4], Section 4.1.5, this does not result in any loss of generality for the solvability of colorless tasks.

2.4 Composing protocols

A natural thing that can be done with functions is composing two of them together to form a new one. We can do the same thing with protocols. The operational idea behind the composition of protocols is that each process starts by executing the first protocol and starts executing the second

protocol when it is done with the first. Here, the input given to the second protocol is the final state of the first protocol. The following is the combinatorial definition of the composition of protocol maps:

Definition 6 (Composition of protocol maps)**.** Let $(\mathcal{I}, \mathcal{P}_1, \Xi_1)$ and $(\mathcal{P}_1, \mathcal{P}_2, \Xi_2)$ be two colorless protocols. We define their **composition** $(\mathcal{I}, \mathcal{P}_2, (\Xi_2 \circ \Xi_1))$ as follows:

$$\Xi_2 \circ \Xi_1 : \mathcal{I}_1 \to 2^{\mathcal{P}_2}, \qquad (\Xi_2 \circ \Xi_1)(\sigma) = \bigcup_{\tau \in \Xi_1(\sigma)} \mathbf{P}_2(\tau)$$

An important thing to realize is that the composition of two protocols might not exist in an operational model, even if those two protocols do both exist in it. In particular, a composition of colorless single-layer protocols is not always a colorless single-layer protocol.

Note that, in this case, $\circ$ is not the normal composition of functions. The map $\Xi_2 : \mathcal{P}_1 \to 2^{\mathcal{P}_2}$ automatically induces the following map

$$\Xi_2 : 2^{\Xi_1} \to 2^{2^{\mathcal{P}_2}}, \qquad \Xi_2(\mathcal{A}) = \{\, \Xi_2(\sigma) \mid \sigma \in \mathcal{A} \,\}$$

If we would then use the normal function composition of Ξ_1 and Ξ_2, we would obtain a map of the form

$$\mathcal{I} \to 2^{2^{\mathcal{P}_2}}$$

In a way, the compositon of execution maps is normal composition composed with the following map that 'flattens' the image:

$$\bigcup : 2^{2^{\mathcal{P}_2}} \to 2^{\mathcal{P}_2}, \qquad f(A) = \bigcup A$$

Using the flattening map after the composition represents the fact that the program's intermediate state after executing the first protocol does not really matter. We are only interested in the final state of the program.

2.5 Colorless layered atomic snapshot protocol

The colorless layered protocol is a composition of colorless single-layered protocols. In particular, an m-layered protocol is the composition of m single-layer protocols, which we call the layers. Each layer needs to execute its code unhindered by the other layers even if other processes are executing other layers at the same time. A protocol that satisfies this property is called **communication-closed**, under the definition given in [3].

We need to modify the operational model, so that colorless layered protocols exist in it. We do this by separating the memory into layers. The first single-layer protocol only writes to the first layer, the second only to the second layer, etc. In pseudocode, we represent the layers as an array where each layer is an element of the array. Each layer is then itself an array where index i is only writable for process i. A **colorless layered atomic snapshot protocol** (or **colorless layered protocol** for short when it is clear from the context) is a protocol that is run on the model described above, using the following pseudocode when it has numLayers layers and numProcesses processes:

```
// these constants are the same for all the processes
const int numLayers;
```

```
const int numProcesses;

// this is different for each process
// it is a value between 0 and numProcesses - 1
const int currentProcessIndex;

// this value points to the same memory location
// for all processes
extern Value[numLayers][numProcesses] memory;

// the colorless layered atomic snapshot protocol
Value executeColorlessLayeredProtocol ( Value input ) {
    // at the start, the only information we have is the input
    Value state = input;

    // loop through all the layers, indexed from 0 to numLayers - 1
    for ( int layer = 0 ; layer < numLayers ; layer++ ) {
        // write our view to our part of the current layer
        update(memory[layer][currentProcessIndex], state);

        // take an atomic snapshot of the current layer
        Value[numProcesses] snapshot = scan(memory[layer]);

        // update our own view based on the snapshot
        // N.B. we only use the Set of Values
        state = setOfValues(snapshot);
    }
    return state;
}
```

The **colorless layered atomic snapshot protocol** defined above is based on the colorless immediate snapshot protocol defined in [4], Section 4.1.5. Instead of immediate snapshots, we use atomic snapshots. Immediate snapshots are a stronger version of atomic snapshots that we will not need in this thesis. The ability to take immediate snapshots instead of atomic snapshots does not affect the solvability of tasks, even though immediate snapshots are more powerful. This was shown in [2].

3 Abstract simplicial complexes

3.1 Abstract simplicial complexes

In this section, we introduce abstract simplicial complexes and some basic definitions based on these complexes. In Section 4.2 we will show that abstract simplicial complexes can be used in defining colorless tasks. Abstract simplicial complexes have been applied in various settings so that proofs about them can be useful for colorless tasks. Particularly interesting will be their connection to geometric simplicial complexes because that connection will enable us to apply results from topology in the context of colorless tasks.

Definition 7 (Abstract simplicial complex)**.** Let S be a set and let $\mathcal{A}$ be a non-empty family of finite subsets of S. Then $\mathcal{A}$ is an **abstract simplicial complex** if and only if the inclusion $2^X \subseteq \mathcal{A}$ holds for each $X \in \mathcal{A}$. In other words, if X is an element of $\mathcal{A}$ then all of the subsets of X are elements of $\mathcal{A}$. Elements of $\mathcal{A}$ are called **simplices** and elements of simplices are called **vertices**. The set of vertices of $\mathcal{A}$ is represented by $V(\mathcal{A}) := \bigcup \mathcal{A}$.

The following is an example of such an abstract simplicial complex.

Example 8. Let S be a finite set. Then 2^S is an abstract simplicial complex. Indeed, for any simplex $\sigma \in 2^S$ and $\tau \in 2^\sigma$, we have that $\tau \subseteq \sigma \subseteq S$ so that $\tau \in 2^S$, which means that $2^\sigma \subseteq 2^S$. Every subset of S is a simplex of 2^S and the vertices of 2^S are precisely the elements of $S = V(2^S)$.

In the next definition, we introduce some basic terminology around simplicial complexes.

Definition 9. Let $\mathcal{A}$ be an abstract simplicial complex and let $\sigma \in \mathcal{A}$. Then the **dimension** of the simplex σ is $\dim \sigma := \#\sigma - 1$. If the dimensions of the simplices of $\mathcal{A}$ have a maximum, we say that this maximum is the dimension of $\mathcal{A}$. Let $m \geqslant -1$ be an integer. Then the m-dimensional **skeleton** $\operatorname{skel}^m \mathcal{A}$ is the set of simplices of $\mathcal{A}$ that have a dimension of at most m. For a simplex σ we define $\operatorname{skel}^m \sigma := \operatorname{skel}^m 2^\sigma$.

Note that we defined the dimension of a simplex as one less than its cardinality as a set. This will prove useful later on when we define geometric simplicial complexes. It mainly has to do with the fact that we can visualize the vertices of an abstract simplicial complex as points and 1-dimensional simplices as line segments between these points.

The following example illustrates what the (-1)- and 0-dimensional skeletons generally are.

Example 10. Let $\mathcal{A}$ be an abstract simplicial complex. By definition it is non-empty, so there is a simplex $\sigma \in \mathcal{A}$. As $\varnothing \subseteq \sigma$, we have that $\varnothing \in \mathcal{A}$. The empty set $\varnothing$ has no elements so its dimension is -1. Therefore $\operatorname{skel}^{-1} \mathcal{A} = \{\varnothing\}$.

Let $v \in V(\mathcal{A})$. Then v is contained in some simplex $\sigma \in \mathcal{A}$, so $\{v\} \subseteq \sigma$, which means that $\{v\} \in \mathcal{A}$. It follows that $\operatorname{skel}^0 \mathcal{A} = \{\varnothing\} \cup \{\, \{v\} \mid v \in V(\mathcal{A}) \,\}$. Because there is a bijection between $v \in V(\mathcal{A})$ and $\{v\} \in \mathcal{A}$, sometimes the simplex $\{v\}$ is also called a vertex.

A subset (resp. proper subset) of a simplex is also called a **face** (resp. **proper face**) of a simplex. A simplex is called a **facet** of $\mathcal{A}$ if it is not the face of any simplex in $\mathcal{A}$. An abstract simplicial complex is called **pure** of dimension d if all of its facets are of dimension d. Subsets of $\mathcal{A}$ that are themselves abstract simplicial complexes are said to be **subcomplexes** of $\mathcal{A}$. The **boundary** $\delta\sigma$ of σ is the subcomplex that consists of proper faces of σ.

The morphisms between abstract simplicial complexes are called simplicial maps, which are defined in the following definition.

Definition 11. Let $\mathcal{A}$ and $\mathcal{B}$ be two abstract simplicial complexes. Any function $\phi : V(\mathcal{A}) \to V(\mathcal{B})$ is called a vertex map. The vertex map ϕ is called a **simplicial map** if for every simplex $\sigma \in \mathcal{A}$, we have that $\phi(\sigma) \in \mathcal{B}$.

The notation $f : A \to B$ indicates that f is a function from A to B, which means each element $a \in A$ is associated with an element $f(a) \in B$. If $C \subseteq A$ then we write $f(C) := \{ f(c) \mid c \in C \}$. In this way f can also be seen as a function from 2^A to 2^B (or even from 2^{2^A} to 2^{2^B}). In particular, a simplicial map ϕ is a function of the form $V(\mathcal{A}) \to V(\mathcal{B})$, but it also automatically extends to a map of the form $\mathcal{A} \to \mathcal{B}$, defined as follows:

$$\phi(\sigma) = \{ \phi(v) \mid v \in \sigma \} \tag{1}$$

If $\phi : \mathcal{A} \to \mathcal{B}$ is a bijection and if both ϕ and its inverse ϕ^{-1} are simplicial maps then we call ϕ and ϕ^{-1} **simplicial isomorphisms**. If a simplicial isomorphism exists between two abstract simplicial complexes, we call them **isomorphic**.

3.2 Carrier maps

Besides simplicial maps, there is another interesting kind of function between abstract simplicial complexes: the carrier map. Let $\mathcal{A}$ and $\mathcal{B}$ be abstract simplicial complexes. A simplicial map from $\mathcal{A}$ to $\mathcal{B}$ maps each simplex in $\mathcal{A}$ to a simplex in $\mathcal{B}$. A carrier map from $\mathcal{A}$ to $\mathcal{B}$ maps each simplex in $\mathcal{A}$ to a *subcomplex* of $\mathcal{B}$.

Definition 12. Let $\mathcal{A}$ and $\mathcal{B}$ be two abstract simplicial complexes. A **carrier map** between $\mathcal{A}$ and $\mathcal{B}$ is a function $\Phi : \mathcal{A} \to 2^{\mathcal{B}}$ such that $\Phi(\sigma)$ is an abstract simplicial complex for every $\sigma \in \mathcal{A}$ and such that the following condition holds for all σ, $\tau \in \mathcal{A}$:

$$\Phi(\sigma) \subseteq \Phi(\tau) \quad \text{if} \quad \sigma \subseteq \tau \tag{2}$$

For a subcomplex $\mathcal{K} \subseteq \mathcal{A}$, we write $\Phi(\mathcal{K}) := \bigcup_{\sigma \in \mathcal{K}} \Phi(\sigma) \subseteq \mathcal{B}$.

The following lemma provides us we a useful alternate definition.

Lemma 1. Let $\mathcal{A}$ and $\mathcal{B}$ be two abstract simplicial complexes and let $\Phi : \mathcal{A} \to 2^{\mathcal{B}}$ be a function. Then condition (2) is equivalent to the following **monotonicity condition**:

$$\Phi(\sigma \cap \tau) \subseteq \Phi(\sigma) \cap \Phi(\tau) \quad \text{for all} \quad \sigma, \tau \in \mathcal{A} \tag{3}$$

Proof. Assume that the carrier map satisfies the monotonicity condition. Pick $\sigma, \tau \in \mathcal{A}$ such that $\sigma \subseteq \tau$. Then, by the monotonicity condition, we find that $\Phi(\sigma) = \Phi(\sigma \cap \tau) \subseteq \Phi(\sigma) \cap \Phi(\tau) \subseteq \Phi(\tau)$, which is condition (2).

Assume that the carrier map satisfies condition (2). Pick any $\sigma, \tau \in \mathcal{A}$. Then $\Phi(\sigma \cap \tau) \subseteq \Phi(\sigma)$ follows from condition (2) since $\sigma \cap \tau \subseteq \sigma$. Similarly, $\Phi(\sigma \cap \tau) \subseteq \Phi(\tau)$ follows from condition (2) since $\sigma \cap \tau \subseteq \tau$. Hence, $\Phi(\sigma \cap \tau) \subseteq \Phi(\sigma) \cap \Phi(\tau)$, which is the monotonicity condition. □

Sometimes we need a carrier map to have additional properties. Two such properties are stated below. They are equivalent to the definitions given in [4], Definition 3.4.2.

Definition 13. Let $\mathcal{A}$ and $\mathcal{B}$ be two abstract simplicial complexes and let $\Phi : \mathcal{A} \to 2^{\mathcal{B}}$ be a carrier map. Then Φ is called **rigid** if for each simplex $\sigma \in \mathcal{A}$ of dimension d, the subcomplex $\Phi(\sigma)$ is pure of dimension d. If the equality $\Phi(\sigma \cap \tau) = \Phi(\sigma) \cap \Phi(\tau)$ holds for all $\sigma, \tau \in \mathcal{A}$ then we say that Φ is **strict**.

Example 14. In this example we show that rigidity does not imply strictness or vice versa, as suggested by [4], Exercise 3.3.

First, we define a strict carrier map that is not rigid. Define $\mathcal{A} := 2^{\varnothing} = \{\varnothing\}$ and $\mathcal{B} := 2^{\{1\}} = \{\varnothing, \{1\}\}$. Note that they are both abstract simplicial complexes. Define:

$$\Phi : \mathcal{A} \to 2^{\mathcal{B}}, \qquad \Phi(\varnothing) = \mathcal{B}$$

Since $\mathcal{A}$ contains only one simplex, it easy to see that Φ is indeed a carrier map and that in addition, Φ is strict. Because $\{1\}$ is a facet of dimension 0, we conclude that $\Phi(\varnothing)$ is not pure of dimension -1. Hence Φ is not rigid.

For defining a rigid map that is not strict we need a slightly larger abstract simplicial complex. Define $\mathcal{A} := \{\varnothing, \{1\}, \{2\}\}$ and $\mathcal{B} := 2^{\{1\}} = \{\varnothing, \{1\}\}$. Note that they are both abstract simplicial complexes. We define $\Phi : \mathcal{A} \to 2^{\mathcal{B}}$ as follows for $\sigma \in \mathcal{A}$:

$$\Phi(\sigma) = \begin{cases} \{\varnothing\} & \text{if } \sigma = \varnothing \\ \mathcal{B} & \text{else} \end{cases}$$

Note that Φ is a carrier map because $\Phi(\varnothing) = \{\varnothing\} \subseteq \mathcal{B} = \Phi(\{1\}) = \Phi(\{2\})$. Also, Φ is rigid because $\Phi(\varnothing) = \{\varnothing\}$ and both $\{1\}$ and $\{2\}$ are of dimension 0 while $\Phi(\{1\})$ and $\Phi(\{2\})$ are pure of dimension 0. Note that $\Phi(\{1\} \cap \{2\}) = \Phi(\varnothing) = \varnothing$ and $\Phi(\{1\}) \cap \Phi(\{2\}) = \mathcal{B} \cap \mathcal{B} = \mathcal{B} \neq \varnothing$, so $\Phi(\{1\} \cap \{2\}) \neq \Phi(\{1\}) \cap \Phi(\{2\})$, which means that Φ is not strict.

An example of a strict carrier map is the skeleton operator, as suggested by Exercise 4.6.

Lemma 2. Let $\mathcal{A}$ be an abstract simplicial complex and let $m \geqslant -1$ be an integer. Then $\operatorname{skel}^m : \mathcal{A} \to 2^{\mathcal{A}}$ is a strict carrier map.

Proof. Pick $\sigma \in \mathcal{A}$. Note that $\operatorname{skel}^m \sigma$ is non-empty because $\varnothing \in \operatorname{skel}^m \sigma$. Let $\tau \in \operatorname{skel}^m \sigma \subseteq 2^{\sigma}$. Pick $\alpha \in 2^{\tau}$, which means that $\alpha \subseteq \tau$. Then we must have that $\#\alpha \leqslant \#\tau$, so $\dim \alpha \leqslant \dim \tau$. Because $\dim \tau \leqslant m$ (since $\tau \in \operatorname{skel}^m \sigma$), we also have that $\dim \alpha \leqslant m$. Since $\alpha \subseteq \tau \subseteq \sigma$, it then follows that $\alpha \in \operatorname{skel}^m \sigma$. Hence, $2^{\tau} \subseteq \operatorname{skel}^m \sigma$, which proves that $\operatorname{skel}^m \sigma$ is a simplicial complex.

Pick $\sigma, \tau \in \mathcal{A}$. The simplices of $\operatorname{skel}^m \sigma \cap \operatorname{skel}^m \tau$ are the simplices that are subsets of both σ and τ for which the dimension is at most m, which is equivalent to them being of dimension at most m and subsets of $\sigma \cap \tau$. Hence $\operatorname{skel}^m \sigma \cap \operatorname{skel}^m \tau = \operatorname{skel}^m \sigma \cap \tau$. It follows from Lemma 1 and Definition 13 that $\operatorname{skel}^m : \mathcal{A} \to 2^{\mathcal{A}}$ is a strict carrier map. □

The next lemma introduces the concept of the carrier of a simplex. The lemma is based on [4], Exercise 3.7 and the definition of the carrier is based on [4], the paragraph above Definition 3.4.3.

Lemma 3. Let $\mathcal{A}$ and $\mathcal{B}$ be abstract simplicial complexes and let $\Phi : \mathcal{A} \to 2^{\mathcal{B}}$ be a strict carrier map. Let $\tau \in \Phi(\mathcal{A})$ be a simplex. Then there exists a unique simplex σ in $\mathcal{A}$ of smallest dimension such that the subcomplex $\Phi(\sigma)$ contains τ. This simplex σ is called the **carrier** of τ and we write that $\sigma = \operatorname{Car}(\tau, \Phi)$.

Proof. Because $\tau \in \Phi(\mathcal{A})$, there must exist an $\alpha \in \mathcal{A}$ such that $\tau \in \Phi(\alpha)$. Because α is a finite set there must be a face $\sigma \subseteq \alpha$ of minimal dimension such that $\Phi(\sigma)$ still contains τ. This proves the existence of a carrier of τ, but we still need to prove its uniqueness to justify speaking of *the* carrier of τ.

Let σ_0 and σ_1 both be carriers of τ in $\mathcal{A}$. By definition, we have that $\tau \in \Phi(\sigma_0)$ and $\tau \in \Phi(\sigma_1)$, so τ is a simplex of the intersection $\Phi(\sigma_0) \cap \Phi(\sigma_1)$. Since Φ is strict, that intersection is equal to $\Phi(\sigma_0 \cap \sigma_1)$. Therefore, we have that $\tau \in \Phi(\sigma_0 \cap \sigma_1)$. Let $i \in \{0, 1\}$. If $\sigma_0 \cap \sigma_1$ were a proper face of σ_i, then its dimension would be smaller than σ_i. But then σ_i would not be the simplex of smallest dimension such that its image under Φ contains τ, which would be a contradiction. Therefore, $\sigma_0 \cap \sigma_1$ is not a proper face of σ_i so $\sigma_0 \cap \sigma_1 = \sigma_i$. As a result, we find that $\sigma_0 = \sigma_0 \cap \sigma_1 = \sigma_1$. This proves that there is indeed only one carrier of τ. □

The following definition is based on [4], Definition 3.4.3.

Definition 15. Let $\mathcal{A}$ and $\mathcal{B}$ be abstract simplicial complexes and let $\Phi, \Psi : \mathcal{A} \to 2^{\mathcal{B}}$ be carrier maps and let $\phi : \mathcal{A} \to \mathcal{B}$ be a simplicial map. Then we say that:

(i) the carrier map Φ is **carried** by Ψ if $\Phi(\sigma) \subseteq \Psi(\sigma)$ for all $\sigma \in \mathcal{A}$, denoted by $\Phi \subseteq \Psi$, and

(ii) the simplicial map ϕ is **carried** by Ψ if $\phi(\sigma) \in \Psi(\sigma)$ for all $\sigma \in \mathcal{A}$.

Assume we have two carrier maps $\Phi : \mathcal{A} \to 2^{\mathcal{B}}$ and $\Psi : \mathcal{B} \to 2^{\mathcal{C}}$. Because the codomain of Φ is given by $2^{\mathcal{B}}$ instead of $\mathcal{B}$, we cannot compose these two carrier maps with each other in the same way we compose normal functions. Therefore, we use the following definition.

Definition 16. Let $\mathcal{A}, \mathcal{B}, \mathcal{C}$ be abstract simplicial complexes and let $\Phi : \mathcal{A} \to 2^{\mathcal{B}}$ and $\Psi : \mathcal{B} \to 2^{\mathcal{C}}$ be carrier maps. Then we define their composition as the carrier map $\Psi \circ \Phi : \mathcal{A} \to 2^{C}$, $(\Psi \circ \Phi)(\sigma) := \bigcup_{\tau \in \Phi(\sigma)} \Psi(\tau)$.

Note that the definition above is similar to how we defined the composition of protocols in Definition 6. In Definition 17, we will see how protocols can be seen as carrier maps.

4 Colorless tasks as simplicial complexes

4.1 Monotonic colorless protocols

In Section 2.2, we split the execution map $\mathbf{P}$ into a protocol map $(\mathcal{I}, \mathcal{P}, \Xi)$ and a decision map δ. Recall that Ξ is a function from $\mathcal{I}$ to $2^{\mathcal{P}}$. If we require $\mathcal{I}$ and $\mathcal{P}$ to be abstract simplicial complexes, Ξ could be a carrier map between these two complexes. Hence, we arrive at the following defintion.

Definition 17. A **monotonic colorless protocol** is a tuple $(\mathcal{I}, \mathcal{P}, \Xi)$ where:

- the **input complex** $\mathcal{I}$ is an abstract simplicial complex with colorless inputs as its simplices
- the **protocol complex** $\mathcal{P}$ is an abstract simplicial complex with colorless final states as its simplices
- the map $\Xi : \mathcal{I} \to 2^{\mathcal{P}}$ is a carrier map where each input simplex $\sigma \in \mathcal{I}$ is mapped to the subcomplex of the protocol complex containing the possible colorless final states of executions with colorless input σ

An **output complex** $\mathcal{O}$ is an abstract simplicial complex that contains all possible colorless outputs of a program. A **monotonic colorless decision map** is a simplicial map $\delta : \mathcal{P} \to \mathcal{O}$ that maps each colorless final state to a colorless output.

Note that any monotonic colorless protocol is a colorless protocol and that any monotonic colorless decision map is a colorless decision map. It does not need to be true the other way around, however.

Let $(\mathcal{I}, \mathcal{P}_1, \Xi_1)$ and $(\mathcal{P}_1, \mathcal{P}_2, \Xi_2)$ be monotonic colorless protocols and let $(\mathcal{I}, \mathcal{P}_2, \Xi_3)$ be their composition as described in Definition 6. Then the map Ξ_3 is precisely the composition of Ξ_2 after Ξ_1 as carrier maps described in Definition 16, because the definitions are so similar.

Consider the scenario where a program has a final colorless state σ but where just before the end some processes crash, resulting in an output that is a subset of σ instead of the entirety of σ. This explains why we can assume that the output complex is a simplicial complex and why the map Ξ maps simplices to subcomplexes.

In an execution where processes crash right after starting, it will be like those processes weren't started at all to the rest of the processes. Therefore, it makes sense that a protocol should be able to run if only a subset of a certain colorless input is given to the program.

A program can also not detect whether processes have crashed. This is the justification for forcing Ξ to be a carrier map. For any $\sigma \subseteq \tau$, any result of an execution of σ could have been the result of an execution of τ, in which the processes with inputs in $\tau \backslash \sigma$ have crashed. Therefore $\Delta(\sigma) \subseteq \Delta(\tau)$ which is the condition required by the definition of a carrier map.

4.2 Monotonic colorless tasks

Using terminology from the previous sections, we discuss colorless tasks consisting of abstract simplicial complexes and carrier maps. The following definition is more restrictive than Definition 2 and is based on [4], Definition 4.2.1.

Definition 18. A **monotonic colorless task** is a colorless task $(\mathcal{I}, \mathcal{O}, \Delta)$ where:

- $\mathcal{I}$ is an abstract simplicial complex

- $\mathcal{O}$ is an abstract simplicial complex
- $\Delta : \mathcal{I} \to 2^{\mathcal{O}}$ is a carrier map

Note that $\varnothing \in \mathcal{I}$ because $\mathcal{I}$ is an abstract simplicial complex. Whether or not $\varnothing$ is in $\mathcal{I}$ or even what the value of $\Delta(\varnothing)$ is does not matter for whether a program solves the task. Indeed, there are no executions of programs where the colorless input is given by $\varnothing$. As a convention, we always set $\Delta(\varnothing) = \{\varnothing\}$.

As to why it is reasonable to require a colorless task to be monotonic, the same reasoning as we used for colorless protocols applies. We want it to let it be possible that the task can be solved by programs that satisfy those same properties.

5 Geometric simplicial complexes

5.1 Geometric simplicial complexes

Abstract simplicial complexes can be given geometric realizations. Not only will this let us visualize simplicial complexes, it will also associate the abstract simplicial complex with a topology, allowing us to use topological results for reasoning about colorless tasks. Before we can define a geometric simplicial complex though, we need to define geometric simplices.

Let $n \geqslant 0$. We denote the n-dimensional real space with $\mathbb{R}^n$. We write 0 for the origin of $\mathbb{R}^n$ (and not just for the origin in $\mathbb{R}$).

Definition 19. Let $-1 \leqslant n \leqslant m$ be integers with $m \geqslant 0$ and let $v_0, \ldots, v_n$ be $n+1$ affinely independent vertices in $\mathbb{R}^m$. The non-negative coefficients $(t_0, \ldots, t_n)$ are called **barycentric coordinates** if their sum $\sum_{i=0}^{n} t_i$ is equal to 1. The n-dimensional **geometric simplex** $\Delta = \operatorname{conv}(\{v_0, \ldots, v_n\})$ spanned by the vertices $v_0, \ldots, v_n$ is the set of points given by:

$$\sum_{i=0}^{n} t_i v_i \quad \text{where} \quad (t_0, \ldots, t_n) \text{ are barycentric coordinates} \tag{4}$$

For each point $p \in \Delta$, there exists precisely one tuple of barycentric coordinates $(t_0, \ldots, t_n)$ such that $p = \sum_{i=0}^{n} t_i v_i$.

The following example illustrates what the geometric simplices are for some low values of n.

Example 20. There is only one (-1)-dimensional geometric simplex: the empty set $\varnothing$. The 0-dimensional simplices are the points in $\mathbb{R}^m$. The 1-dimensional simplices are line segments with non-zero length in $\mathbb{R}^m$. The barycentric coordinates of the vertex at one end of the simplex are given by $(1, 0)$ and those of the vertex at the other end are given by $(0, 1)$. The barycentric coordinates $(\frac{1}{2}, \frac{1}{2})$ represent the point in the middle of the line segment. The 2-dimensional simplices are triangles with non-zero area. The 3-dimensional simplices are tetrahedrons with non-zero volume.

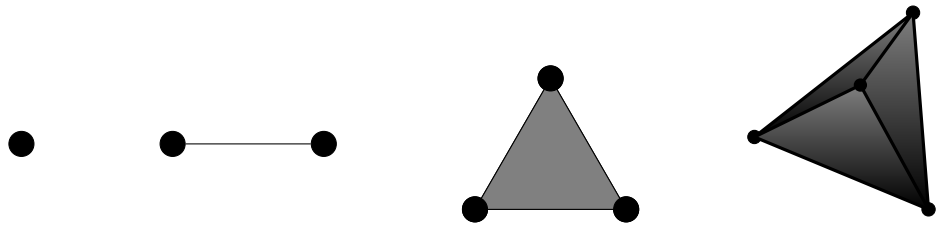

Figure 1: Some geometric simplices in increasing order of dimension

Note that any two simplices Δ_1, Δ_2 with the same dimension are homeomorphic, because we can construct a homeomorphism by taking each point in Δ_1 to its barycentric coordinates and then finding the point in Δ_2 with those same barycentric coordinates.

Given a geometric simplex $\Delta = \operatorname{conv} V$, for each subset $W \subseteq V$, we say that $\operatorname{conv} W$ is a **face** of Δ. For the set of faces of Δ, we write $\operatorname{Faces}(\Delta) := \operatorname{conv} 2^W$. Because all of the vertices in V are independent, none of these faces are the same. Note that any face of Δ is a subset of Δ. The following lemma illustrates how many faces there are.

Lemma 4. Let $-1 \leqslant m \leqslant n$ be integers. If Δ is an n-dimensional geometric simplex then it has 2^n faces and $\binom{n+1}{m+1}$ of these faces are of dimension m.

Proof. The faces of the Δ correspond bijectively to the subsets of the vertices of Δ. Since Δ has dimension n, it has $n+1$ vertices. From set theory, we know that the power set of a set containing $n+1$ elements has 2^{n+1} elements. Hence, there are 2^{n+1} faces of Δ.

There are $\binom{n+1}{m+1}$ subsets containing $m+1$ vertices (because we choose a set of $m+1$ vertices out of $n+1$ possible vertices). Since all of these subsets of vertices span different geometric simplices, we find that there are indeed $\binom{n+1}{m+1}$ geometric simplices of dimension $m+1$. □

Example 21. Any 3-dimensional geometric simplex has $2^4 = 16$ faces:

- 1 simplex of dimension -1 (the empty set)
- 4 simplices of dimension 0 (the vertices)
- 6 simplices of dimension 1 (the edges)
- 4 simplices of dimension 2 (the triangles)
- 1 simplex of dimension 3 (the tetrahedron)

In topology, the interior of a subspace with respect to some topological space is often a useful concept. For geometric simplices, we have a concept of interior that is independent of the surrounding topological space the simplex is embedded in.

Definition 22. Let Δ be a geometric simplex. The **interior** of Δ is defined as the part of the simplex that is not contained in any of its proper faces, so we write $\operatorname{Int}\Delta := \Delta \setminus \bigcup(\operatorname{Faces}(\Delta)\setminus\{\Delta\})$ for the interior of Δ.

Note that the definition of the interior we provided above does not always coincide with the topological interior of the geometric simplex with respect to some topological it is embedded in. For a geometric simplex of dimension n, the interior is the topological interior of the geometric simplex with respect to $\mathbb{R}^n$.

We can now define geometric simplicial complexes based on the definition of simplices.

Definition 23. Let $\mathcal{K}$ be a collection of geometric simplices. Then $\mathcal{K}$ is a **geometric simplicial complex** if and only if the following conditions hold:

(i) $\operatorname{Faces}(\sigma) \subseteq \mathcal{K}$ for all $\sigma \in \mathcal{K}$

(ii) $\sigma \cap \tau \in \operatorname{Faces}(\sigma) \cap \operatorname{Faces}(\tau)$ for all $\sigma, \tau \in \mathcal{K}$

We write $|\mathcal{K}| := \bigcup \mathcal{K}$ where $|\mathcal{K}|$ is equipped with the induced topology as subset of $\mathbb{R}^m$. We write $C(\mathcal{K})$ for the abstract simplicial complex given by $\{\, C(\sigma) \mid \sigma \in \mathcal{K} \,\}$. Note that C forms a bijection between the abstract simplicial complex and its **geometric realization** $\mathcal{K}$.

Note that the first condition of Definition 23 is very similar to the condition in the definition of an abstract simplicial complex, namely that $2^\sigma \subseteq \mathcal{A}$ for all $\sigma \in \mathcal{A}$. It follows from this similarity that $C(\mathcal{K})$ is a simplicial complex. Any geometric simplex that has dimension n as subspace of $\mathbb{R}^m$ gets mapped to an n-dimensional abstract simplex. This is the reason for defining the dimension as one less than the cardinality.

In the same way that the power set of any finite set is an abstract simplicial complex, we have that the faces of any geometric simplex form a geometric simplicial complex. It immediately follows from the definition that $\bigl|\operatorname{Faces}(\sigma)\bigr| = \sigma$ for any geometric simplex σ since all faces of σ are subsets of σ.

The concepts of simplicial maps, carrier maps, skeletons, boundaries, facets and subcomplexes defined on abstract simplicial complexes are all extended to geometric simplicial complexes by applying them through the bijection C. For example, given two geometric simplicial complexes $\mathcal{K}, \mathcal{L}$, a map $\phi : \mathcal{K} \to \mathcal{L}$ is a simplicial map if and only if the map $C_{\mathcal{L}}^{-1} \circ \phi \circ C_{\mathcal{K}}$ is simplicial.

An important property of geometric complexes is given by the following lemma.

Lemma 5. Let $\mathcal{K}$ be a geometric simplicial complex. Then for any $p \in |\mathcal{K}|$, there is precisely one simplex $\sigma \in \mathcal{K}$ such that $p \in \operatorname{Int}(\sigma)$. In other words, $\{\operatorname{Int}(\sigma) \mid \sigma \in \mathcal{K}\}$ is a partition of $|\mathcal{K}|$.

Proof. Choose any $p \in |\mathcal{K}|$. Then there exists a simplex $\tau = \operatorname{conv}\{v_0, \ldots, v_n\} \in \mathcal{K}$ of smallest dimension such that $p \in \tau$. We know that p cannot be inside one of the proper faces of τ because those faces have smaller dimension. Therefore p is in the interior of τ.

Now let σ, τ be any two simplices such that $p \in \operatorname{Int}(\sigma)$ and $p \in \operatorname{Int}(\tau)$. Then we have that $p \in \sigma \cap \tau$ and it follows from Definition 23 that $\sigma \cap \tau \in \operatorname{Faces}(\sigma) \cap \operatorname{Faces}(\tau)$. If $\sigma \cap \tau$ were a proper face of σ, then p would, by definition, not be in $\operatorname{Int}(\sigma)$. Hence, $\sigma = \sigma \cap \tau$. The same reasoning applies for τ so we have that $\tau = \sigma \cap \tau = \sigma$. This proves that $\sigma = \tau$. □

Assume $\mathcal{K}$ is a finite geometric simplicial complex with $V(C(\mathcal{K})) = \{v_0, \ldots, v_n\}$. Let p be a point in $|\mathcal{A}|$. Because of the lemma above there is a unique simplex σ such that $p \in \operatorname{Int}(\sigma)$. We can then define its **barycentric coordinates** $(t_0, \ldots, t_n)$ by setting $t_i = 0$ if $v_i \notin C(\sigma)$ and setting the other coordinates equal to the barycentric coordinates in the geometric simplex σ. Note that we then have that $p = \sum_{i=0}^{n} t_i v_i$.

Definition 24. Let $\mathcal{A}$ and $\mathcal{B}$ be two geometric simplicial complexes and let $\phi : \mathcal{A} \to \mathcal{B}$ be a simplicial map. Then we define the map $|\phi| : |\mathcal{A}| \to |\mathcal{B}|$ on the interior $\operatorname{Int}(\sigma)$ with $\sigma = \{v_0, \ldots, v_n\}$ by setting $|\phi|(p) = \sum_{i=0}^{n} t_i \phi(v_i)$ where $(t_0, \ldots, t_n)$ are the barycentric coordinates of p in σ.

Lemma 6. Let $\mathcal{A}$ be a finite abstract simplicial complex. Then there exists a geometric simplicial complex $\mathcal{K}$ such that $C(\mathcal{K})$ and $\mathcal{A}$ are isomorphic as abstract simplicial complexes.

Proof. Because $\mathcal{A}$ is finite, we can write $V(\mathcal{A}) = \{v_0, v_1, \ldots, v_n\}$. Define $\mathcal{B} = \{0, e_1, \ldots, e_n\}$, where where $0 \in \mathbb{R}^n$ and $e_1, \ldots, e_n$ are the unit vectors in $\mathbb{R}^n$. We can construct a simplicial isomorphism $\phi : V(\mathcal{A}) \to V(\mathcal{B})$ by setting $\phi(v_0) = 0$ and $\phi(v_i) = e_i$ for $i > 0$. These vertices are all affinely independent, so all the simplices of the form $\phi(\sigma)$ are faces of the geometric simplex spanned by the vertices $\phi(v_i)$. Hence, $\mathcal{K} := \operatorname{conv}(\mathcal{B})$ is a geometric simplicial complex with $C(()\mathcal{K}) = \mathcal{B}$, which is isomorphic to $\mathcal{A}$. □

Note that the geometric simplicial complex constructed is quite large. It is often possible to find embeddings in $\mathbb{R}^n$ for much lower values of n, but for us the existence is enough, because we have the following lemma:

Lemma 7. Let $\mathcal{K}, \mathcal{L}$ be two finite geometric simplical complexes such that $C(\mathcal{K})$ and $C(\mathcal{L})$ are isomorphic as abstract simplicial complexes. Then $|\mathcal{K}|$ and $|\mathcal{L}|$ are homeomorphic.

In the lemma above we assume that $\mathcal{A}$ is finite. A proof of can be found in [5], Lemma 2.8 (the topology used for geometric simplicial complexes in [5]is different than the subset topology induced by $\mathbb{R}^n$, but it is the same for finite complexes).

Using the two previous lemmas we conclude that the following construction is well-defined (up to homeomorphism):

Definition 25. Let $\mathcal{A}$ be a finite abstract simplicial complex and choose a geometric simplicial complex $\mathcal{K}$ such that $C(\mathcal{K}) \cong \mathcal{A}$. Then we say that the **geometric realization** of $\mathcal{A}$ is the topological space given by $|\mathcal{K}|$.

We also have the following result for the map $|\phi|$:

Lemma 8. Let $\mathcal{A}$ and $\mathcal{B}$ be two geometric simplicial complexes and let $\phi : \mathcal{A} \to \mathcal{B}$ be a simplicial map. If $\mathcal{A}$ is finite then the map $|\phi|$ is continuous.

This is proven in [5], Lemma 2.7.

Lemma 9. Let Δ be an n-dimensional geometric simplex. Then there is a homeomorphism from Δ to D^n such that $\Delta \setminus \operatorname{Int}(\Delta)$ is mapped to S^{n-1}.

This lemma is proven in [5], Lemma 1.1b. This automatically means that the same applies to the geometric realization of a simplex of an abstract simplicial complex.

5.2 The star construction

The star of a simplex is a construction that can be useful when stating or proving local properties in a simplicial complex. This section is based on [4], Section 3.3.1.

Definition 26. Let $\mathcal{A}$ be an abstract simplicial complex and let $\sigma \in \mathcal{A}$ be a simplex. Then the **star** of σ in $\mathcal{A}$ is defined as $\operatorname{St}(\sigma, \mathcal{A}) := \{\tau \in \mathcal{A} \mid \sigma \subseteq \tau\}$. If $\mathcal{A}$ is clear from the context, we also write $\operatorname{St}(\sigma)$.

If $\mathcal{K} = (\mathcal{A}, G)$ is a geometric simplicial complex, then the **open star** of σ is defined as $\operatorname{St}^{\circ}(\sigma, \mathcal{K}) := \sqcup_{\tau \in \operatorname{St}(\sigma, \mathcal{K})} \operatorname{Int} |\tau|$. The fact that this union is indeed disjoint follows from Lemma 5.

For vertices v we write $\operatorname{St}(v, \mathcal{A}) := \operatorname{St}(\{v\}, \mathcal{A})$ and $\operatorname{St}^{\circ}(v, \mathcal{K}) := \operatorname{St}^{\circ}(\{v\}, \mathcal{K})$

Note that the open star is open in the topology of the simplicial complex. In some ways, the star is like a simplicial equivalent of the the balls of radius ϵ in a metric space as demonstrated by the following lemma, which is based on a remark made in [4], Section 3.3.1.

Lemma 10. Let $\mathcal{K}$ be a geometric simplicial complex. Then $(\operatorname{St}^{\circ}(v, \mathcal{K}))_{v \in V(\mathcal{A})}$ is an open cover of $|\mathcal{K}|$.

Proof. Let $p \in |\mathcal{B}|$. Because of Lemma 5, p is contained in $\operatorname{Int} |\sigma|$ for some simplex σ, so for any $v \in \sigma$ we have that $p \in \operatorname{St}^{\circ} v$. Therefore, the collection $(\operatorname{St}^{\circ} v)_{v \in V(\mathcal{B}))}$ of open stars around vertices in $\mathcal{B}$ forms an open cover of $|\mathcal{B}|$. □

5.3 The join

The join is a useful construction that is defined as follows on abstract simplicial complexes.

Definition 27. Let $\mathcal{A}, \mathcal{B}$ be two abstract simplicial complexes. Then the **join** of $\mathcal{A}$ and $\mathcal{B}$ is given by $\mathcal{A} * \mathcal{B} := \{ \sigma \cup \tau \mid \sigma \in \mathcal{A}, \tau \in \mathcal{B} \}$.

Note that $\mathcal{A} \subseteq \mathcal{A} * \mathcal{B}$ and $\mathcal{B} \subseteq \mathcal{A} * \mathcal{B}$ since $\varnothing \in \mathcal{B}$ and $\varnothing \in \mathcal{A}$. It is easy to see that $\mathcal{A} * \mathcal{B}$ is an abstract simplicial complex.

Now we try to extend this concept to geometric simplicial complexes. Let $\mathcal{K}$ and $\mathcal{L}$ be two geometric simplicial complexes where $\mathcal{K}, \mathcal{L} \subseteq \mathbb{R}^n$ for some $n \in \mathbb{N}$. If every simplex in $C(\mathcal{K}) * C(\mathcal{L})$ consists of affinely independent vertices, then we can construct the geometric join like this:

$$\mathcal{K} * \mathcal{L} := \{ \operatorname{conv}(\alpha) \mid \alpha \in C(\mathcal{K}) * C(\mathcal{L}) \} = \{ \operatorname{conv}(C(\sigma) \cup C(\tau)) \mid \sigma \in \mathcal{K}, \tau \in \mathcal{L} \}$$

If $\mathcal{K}$ is pure of dimension d and v is a vertex then $v * \mathcal{K}$ is called the **cone** over $\mathcal{K}$ with **apex** v.

5.4 Connectedness

For $n \geqslant 0$, the n-disk D^n is the subset of $\mathbb{R}^n$ for which the distance to the origin is at most 1. The $(n-1)$-sphere S^{n-1} is the subset of D^n for which the distance to the origin is precisely 1. Here, $\mathbb{R}^0$ is defined as the set containing only the origin, which we write as 0. In particular, this means that $D^0 = \{0\}$ and that $S^{-1} = \varnothing$. We will use the following topological definition of k-connectedness.

Definition 28. Let $n \geqslant 1$ and $k \geqslant -1$ be integers, Let $X \subseteq \mathbb{R}^n$. Then $\mathcal{K}$ is said to be k-connected if and only if any continuous map $f : S^l \to |\mathcal{K}|$ with $-1 \leqslant l \leqslant k$ can be extended to a continuous map $F : D^{l+1} \to |\mathcal{K}|$ such that $F|_{S^l} = f$. If $\mathcal{K}$ is 0-connected, then it is also called **path-connected**.

The following lemma provides a condition that is equivalent to (-1)-connectedness.

Lemma 11. Let $\mathcal{K}$ be a geometric simplicial complex or an abstract simplicial complex. Then $\mathcal{K}$ is (-1)-connected if and only if $|\mathcal{K}|$ is non-empty.

Proof. Since $D^0 = \{0\}$ and $S^{-1} = \varnothing$, we find that $\mathcal{K}$ is (-1)-connected if and only if any continuous map $f : \varnothing \to |\mathcal{K}|$ can be extended to a continuous map $F : \{0\} \to |\mathcal{K}|$ such that $F|_{\varnothing} = f$. Note that there is precisely one function of the form $f : \varnothing \to |\mathcal{K}|$ so that $F|_{\varnothing} = f$ for any function $F : \{0\} \to |\mathcal{K}|$. If $|\mathcal{K}|$ is empty, then there is no such F, which means that $\mathcal{K}$ is not (-1)-connected. Assume that $|\mathcal{K}|$ is non-empty. Pick some $p \in |\mathcal{K}|$ and set $F(0) = p$. Since this function is constant, it is automatically continuous. Hence, it follows from the definition that $|\mathcal{K}|$ is (-1)-connected. □

5.5 Subdivisions

The idea behind a subdivision is that we split up each simplex into multiple simplices in such a way that the topology remains the same.

Definition 29. Let $\mathcal{A}$ be an abstract simplicial complex. A **subdivision** of $\mathcal{A}$ is an abstract simplicial complex $\operatorname{Div}_{\mathcal{A}} \mathcal{A}$ together with a carrier map $\operatorname{Div}_{\mathcal{A}} : \mathcal{A} \to 2^{\operatorname{Div}_{\mathcal{A}} \mathcal{A}}$ such that the following conditions hold:

(i) $\operatorname{Div}_{\mathcal{A}} \sigma$ is finite

(ii) $|\sigma| = |\operatorname{Div}_{\mathcal{A}} \sigma|$ for all $\sigma \in \mathcal{A}$

(iii) $\operatorname{Div}_{\mathcal{A}} \mathcal{A} = \bigcup_{\sigma \in \mathcal{A}} \mathcal{A}$

5.5.1 The barycentric subdivision

The barycentric subdivision is a standard subdivision that can be applied to any simplicial complex. To define it, we need the geometric concept of a barycenter.

Definition 30. Let $\Delta = \operatorname{conv}\{v_0, \dots, v_n\}$ be a geometric simplex. Then the **barycenter** of Δ is the point given by $\sum_{i=0}^{n} \frac{1}{n+1} v_n$. The barycenter can also be seen as the average of the vertices or as the point with barycentric coordinates $(\frac{1}{n+1}, \dots, \frac{1}{n+1})$.

We can use this to define the barycentric subdivision of simplices and simplicial complexes.

Definition 31. Let σ be a geometric simplex with barycenter p. We write $\operatorname{Bary}\sigma$ for its **barycentric subdivision**. It is defined by induction on d.

If $d \leqslant 0$, then $\operatorname{Bary}\sigma = \sigma$.

If $d > 0$, then $\operatorname{Bary}\sigma$ is the cone over $\operatorname{skel}^{d-1}\mathcal{A}$ with apex p.

Lemma 12. Let $\mathcal{A}$ be a geometric simplicial complex. Then the function $\operatorname{Bary} : \mathcal{A} \to \operatorname{Im}(\operatorname{Bary})$ (where $\operatorname{Bary}\sigma$ is as defined above) is a subdivision of $\mathcal{A}$.

A proof of the lemma above can be found in [4], Section 3.6.2. The subdivision $\operatorname{Bary} : \mathcal{A} \to \operatorname{Im}(\operatorname{Bary})$ is also called the barycentric subdivision of $\mathcal{A}$.

Since Bary is defined on any abstract simplicial complex and in particular on $\operatorname{Bary}\mathcal{A}$ for any abstract simplicial complex $\mathcal{A}$, we can repeatedly apply Bary, we denote this by Bary^N for some natural N. Note that Bary^N is a subdivision and a composition of carrier maps as in Definition 16.

If we keep applying the barycentric subdivision the diameters of the simplices get smaller and smaller. This is formalized in the following lemma.

Lemma 13. Let $\mathcal{A}$ be a geometric simplicial complex. For every $\epsilon > 0$ there is an $N > 0$ such that all simplices in $\operatorname{Bary}\mathcal{A}$ have diameter less than ϵ.

There is also an equivalent abstract definition of the barycentric subdivision:

Definition 32. Let $\mathcal{A}$ be a geometric simplex and let $\sigma \in \mathcal{A}$. Then the vertices of $\operatorname{Bary}\sigma$ are the non-empty faces of σ and a collection of vertices in $\operatorname{Bary}\sigma$ given by $\sigma_0, \sigma_1, \dots, \sigma_k$ forms a face of $\operatorname{Bary}\sigma$ if and only if they can be reindexed so that $\sigma_0 \subseteq \sigma_1 \subseteq \cdots \subseteq \sigma_k$.

5.6 Simplicial approximations

It is clear that each simplicial map between two simplicial complexes can be tranformed into a map between the geometric realizations of those simplicial complexes. This geometric realization of a simplicial map turns out to be continuous.

The other way around however, continuous maps between geometric realizations aren't always geometric realizations of simplicial maps. While a simplicial map maps simplices to simplices, most continuous maps don't map simplices to simplices. We want to allow the continuous map to be different from geometric realizations of simplicial maps but not so different that it is unclear which simplicial map it corresponds to. This leads us to the following definition.

Definition 33. A simplicial map $\phi : \mathcal{A} \to \mathcal{B}$ is a simplicial approximation to $f : |\mathcal{A}| \to |\mathcal{B}|$ if for every simplex $\sigma \in \mathcal{A}$, the following condition holds:

$$f(\operatorname{Int}|\sigma|) \subseteq \operatorname{St}^{\circ}(\phi(\sigma), \mathcal{B}) \tag{5}$$

Informally, simplicial approximations are simplicial maps for which the image under f of each simplex σ happens to be so close to $\phi(\sigma)$ that it can be 'shifted' towards $\phi(\sigma)$ without intersecting any simplices. The following lemma will be useful when proving that a simplicial approximation exists for a particular continuous map.

Lemma 14. A vertex map $\phi : V(\mathcal{A}) \to V(\mathcal{B})$ is a simplicial approximation to $f : |A| \to |B|$ if and only if the following condition holds for every simplex $\sigma \in \mathcal{A}$:

$$f(\operatorname{Int}|\sigma|) \subseteq \bigcap_{v \in \sigma} \operatorname{St}^{\mathrm{o}}(\phi(v), \mathcal{B}) \tag{6}$$

Proof. Let $\sigma \in \mathcal{A}$. Let $\tau \in \bigcap_{v\in\sigma} \operatorname{St}(\phi(v), \mathcal{B})$. Then τ must contain $\phi(v)$ for each $v \in \sigma$. In other words, $\phi(\sigma) \subseteq \tau$, so $\tau \in \operatorname{St}(\phi(\sigma), \mathcal{B})$. Hence $\bigcap_{v\in\sigma} \operatorname{St}(\phi(v), \mathcal{B}) \subseteq \operatorname{St}(\phi(\sigma), \mathcal{B})$. Let $\tau \in \operatorname{St}(\phi(\sigma), \mathcal{B})$. By definition, $\phi(\sigma) \subseteq \tau$. Pick any $v \in \sigma$. Clearly, $v \in \tau$, so by definition, $\tau \in \operatorname{St}(\phi(v), \mathcal{B}) \subseteq \bigcap_{v\in\sigma} \operatorname{St}(\phi(v), \mathcal{B})$. This proves that $\bigcap_{v\in\sigma} \operatorname{St}(\phi(v), \mathcal{B}) = \operatorname{St}(\phi(v), \mathcal{B})$. A consequence is that $\bigcap_{v\in\sigma} \operatorname{St}^{\mathrm{o}}(\phi(v), \mathcal{B}) = \operatorname{St}^{\mathrm{o}}(\phi(\sigma), \mathcal{B})$.

Assume that condition (6) holds. We still need to prove that ϕ is a simplicial map. Let $\sigma \in \mathcal{A}$. If σ is empty then $\phi(\sigma) = \varnothing \in \mathcal{B}$. Assume σ is non-empty and let $v \in \sigma$. Then $\varnothing \neq f(\operatorname{Int}|\sigma|) \subseteq \bigcap_{v\in\sigma} \operatorname{St}^{\mathrm{o}}(\phi(v), \mathcal{B})$. It follows that $\bigcap_{v\in\sigma} \operatorname{St}(\phi(v), \mathcal{B})$ is non-empty so there is a simplex in $\mathcal{B}$ containing $\phi(\sigma)$. From Definition 7 it then follows that $\phi(\sigma)$ is a simplex in $\mathcal{B}$. Since σ was chosen arbitrarily in $\mathcal{A}$ this implies that ϕ is indeed a simplicial map. □

The following theorem is a very important result that establishes the connection between continuous and simplicial maps. Using this theorem, we can transform topological results about the existence of continuous maps into combinatorial results about the existence of simplicial maps.

Theorem 34. *Let $\mathcal{A}$ and $\mathcal{B}$ be finite abstract simplicial complexes. Let $f : |\mathcal{A}| \to |\mathcal{B}|$ be a continuous map between their geometric realizations. Then there exists an $N > 0$ and a simplicial map $\phi : \operatorname{Bary}^N \mathcal{A} \to \mathcal{B}$ such that ϕ is a simplicial approximation of f.*

Proof. Note that $|\operatorname{Bary}^N \mathcal{A}| = |\mathcal{A}|$ so the statement that $\phi : \operatorname{Bary}^N \mathcal{A} \to \mathcal{B}$ is a simplicial approximation of f is well-defined.

By taking the inverse image of each of the open sets, we obtain $(f^{-1}(\operatorname{St}^{\mathrm{o}} w))_{w\in V(\mathcal{B})}$, which is then an open covering of $|\mathcal{A}|$, because f is continuous. Because $\mathcal{A}$ is finite, its geometric realization is compact, so by Lesbesgue's number lemma it must have a Lebesgue number $\rho > 0$ such that any closed set of diameter less than ρ is contained in a set $f^{-1}(\operatorname{St}^{\mathrm{o}} w)$ for some $w \in V(\mathcal{B})$.

Now take any vertex $v \in V(\mathcal{A})$. Note that there is an N such that $\operatorname{diam}(|\operatorname{St}(v, \operatorname{Bary}^N \mathcal{A})|) < \rho$, because of Lemma 13. Because there are a finite number of vertices in $V(\mathcal{A})$, we can choose an N such that $\operatorname{diam}(|\operatorname{St}(v, \operatorname{Bary}^N \mathcal{A})|) < \rho$ for all $v \in V(\mathcal{A})$. Since $|\operatorname{St}(v, \operatorname{Bary}^N \mathcal{A})|$ is closed in $|\mathcal{A}|$ with diameter less than ρ, there should be a vertex $w \in V(\mathcal{B})$ such that $\operatorname{St}^{\mathrm{o}}(v, \operatorname{Bary}^N \mathcal{A}) \subseteq |\operatorname{St}(v, \operatorname{Bary}^N \mathcal{A})| \subseteq f^{-1}(\operatorname{St}^{\mathrm{o}} w)$, meaning that $f(\operatorname{St}^{\mathrm{o}}(v, \operatorname{Bary}^N \mathcal{A})) \subseteq \operatorname{St}^{\mathrm{o}} w$. Because $\mathcal{A}$ is finite, we can use this to construct a vertex map $\phi : V(\operatorname{Bary}^N \mathcal{A}) \to V(\mathcal{B})$ where each vertex $v \in V(\operatorname{Bary}^N \mathcal{A})$ is mapped to a vertex $\phi(v) \in V(\mathcal{B})$ such that the following condition holds:

$$f(\operatorname{St}^{\mathrm{o}}(v, \operatorname{Bary}^N \mathcal{A})) \subseteq \operatorname{St}^{\mathrm{o}}(\phi(v), \mathcal{B})$$

Now, let σ be any simplex of $\operatorname{Bary}^N \mathcal{A}$ and let v be a vertex of σ. Clearly, $\sigma \in \operatorname{St}(v, \operatorname{Bary}^N \mathcal{A})$, so $\operatorname{Int}|\sigma| \subseteq \operatorname{St}^{\mathrm{o}}(v, \operatorname{Bary}^N \mathcal{A})$. Then $f(\operatorname{Int}|\sigma|) \subseteq f(\operatorname{St}^{\mathrm{o}}(v, \operatorname{Bary}^N \mathcal{A})) \subseteq \operatorname{St}^{\mathrm{o}}(\phi(v), \mathcal{B})$, which implies that $f(\operatorname{Int}|\sigma|) \subseteq \operatorname{St}^{\mathrm{o}}(\phi(v), \mathcal{B})$. Because this is true for all $v \in \sigma$, we find that $f(\operatorname{Int}|\sigma|) \subseteq \bigcap_{v\in\sigma} \operatorname{St}^{\mathrm{o}}(\phi(v), \mathcal{B})$. It follows from Lemma 14 that ϕ is a simplicial approximation to f. □

5.7 Approximations of carrier maps

In the same fashion as in the last section, we now try to approximate carrier maps. We will discuss two different kinds of approximations of carrier maps: continuous approximations and simplicial approximations. Theorem 36 and Corollary 38 and their proofs are based on [4], Theorem 3.7.7.

We start with continuous approximations.

Definition 35. Let $\mathcal{A}, \mathcal{B}$ be finite abstract simplicial complexes. A function $f : |\mathcal{A}| \to |\mathcal{B}|$ is called a **continuous approximation** of a carrier map $\Phi : \mathcal{A} \to 2^{\mathcal{B}}$ if and only if the following condition holds for every simplex $\alpha \in \mathcal{A}$:

$$f(|\alpha|) \subseteq |\Phi(\alpha)|$$

A continuous approximation of Φ is also said to be **carried by** Φ.

Note that this definition does not put a lot of requirements on the map f. The image $\Phi(\alpha)$ is a subcomplex of $\mathcal{B}$, but if f maps $|\alpha|$ to the geometric realization of just a single simplex of $\mathcal{B}$, then it still classifies as a continuous approximation. Therefore, it might be the case that a continuous approximation 'looks' much simpler than the carrier map it approximates.

Still, not all carrier maps have a continuous approximation, although the following theorem shows that, for carrier maps that satisfy some additional connectivity condition, we can always construct a continuous approximation.

Theorem 36. *Let $\mathcal{A}$ and $\mathcal{B}$ be finite abstract simplicial complexes and let $\Phi : \mathcal{A} \to 2^{\mathcal{B}}$ be a carrier map. Assume that the subcomplex $\Phi(\sigma)$ is $(\dim(\sigma) - 1)$-connected for every simplex $\sigma \in \mathcal{A}$. Then there exists a continuous approximation $f : |\mathcal{A}| \to |\mathcal{B}|$ of Φ.*

Proof. For proving this theorem we will start with a continuous approximation of Φ restricted to the vertices and we will then use $(\dim(\sigma) - 1)$-connectedness to inductively construct continuous approximations on higher-dimensional skeletons.

For each $v \in V(\mathcal{A})$, we have that $|\Phi(v)|$ is non-empty because $\Phi(v)$ is (-1)-connected, so it is possible to construct a function $f_0 : |\operatorname{skel}^0 \mathcal{A}| \to |\mathcal{B}|$ by setting $|v|$ to some element of $|\Phi(v)|$. This is well-defined because any point in $|\operatorname{skel}^0 \mathcal{A}|$ is a geometric realization of a 0-dimensional simplex in $\mathcal{A}$. Furthermore, it is always possible to construct this map, because $\mathcal{A}$ is finite. Note that this function f_0 is a continuous approximation of $\Phi|_{\operatorname{skel}^0 \mathcal{A}}$.

For our induction hypothesis, assume that we have constructed a continuous approximation f_d of $\Phi|_{\operatorname{skel}^d \mathcal{A}}$. Let σ be a $(d+1)$-dimensional simplex in $\mathcal{A}$. Clearly any proper face τ of σ is in the d-dimensional skeleton $\operatorname{skel}^d \mathcal{A}$, so $f(|\tau|) \subseteq |\Phi(\tau)|$ because of the induction hypothesis. Since Φ is a carrier map, we know that $\Phi(\tau) \subseteq \Phi(\sigma)$ so $f(|\tau|) \subseteq |\Phi(\tau)| \subseteq |\Phi(\sigma)|$.

Note that $\Phi(\sigma)$ is d-connected (because σ is of dimension $d+1$ and we assumed that $\Phi(\sigma)$ is $(\dim \sigma - 1)$-connected) and that there is a homeomorphism g between $|\sigma|$ and the $(d+1)$-disk such that $g(|\delta\sigma|)$ is the d-sphere. Using the definition of d-connectedness we then find that $f_d|_{|\delta\sigma|}$ can be extended to a continuous map $f_\sigma : |\sigma| \to |\Phi(\sigma)|$.

If we perform this construction for all $\sigma \in \operatorname{skel}^{d+1} \mathcal{A} \setminus \operatorname{skel}^d \mathcal{A}$, we can construct a map $f_{d+1} : |\operatorname{skel}^{d+1} \mathcal{A}| \to |\mathcal{B}|$ such that $f_{d+1}|_{|\sigma|} = f_\sigma$. This map is well-defined and continuous because all maps f_σ agree on $\operatorname{skel}^d \mathcal{A}$. □

We now prove the following lemma, which will make proving Theorem 37 easier. It is based on [4], Proposition 3.7.4, but it is worded slightly differently so that it is easier to use.

Lemma 15. Let $\mathcal{A}$ and $\mathcal{B}$ be finite abstract simplicial complexes. Assume that $f : |\mathcal{A}| \to |\mathcal{B}|$ is a continuous map and that ϕ is a simplicial approximation of f. Let α be a simplex of $\mathcal{A}$ and let $\mathcal{C}$ be any subcomplex of $\mathcal{B}$ such that $f(|\alpha|) \subseteq |\mathcal{C}|$. Then $\phi(\alpha)$ is a simplex in $\mathcal{C}$.

Proof. Let $x \in \operatorname{Int}(\alpha)$. Because of Lemma 5, there exists a unique $\beta \in \mathcal{B}$ such that $f(x) \in \operatorname{Int}\beta$.

By the definition of a simplicial approximation, we have that $f(x) \in \operatorname{St}^{\circ}(\phi(\alpha))$, which means by definition that there is a simplex in the $\operatorname{St}\phi(\alpha)$ such that $f(x)$ is contained in the interior of that simplex. Note that that simplex must be β. Hence, $\beta \in \operatorname{St}\phi(\alpha)$.

Because of Lemma 5, we find that there is a $\tau \in \mathcal{C}$ such that $f(x) \in \operatorname{Int}\tau$. Since $\mathcal{C}$ is a subcomplex of $\mathcal{B}$, we have that $\tau \in \mathcal{B}$. Because $\tau \in \mathcal{B}$ and $f(x) \in \operatorname{Int}\tau$, we find that $\tau = \beta$. Therefore $\beta \in \mathcal{C}$.

Combining the results from the previous paragraphs, we have that $\beta \in \operatorname{St}\phi(\alpha) \cap \mathcal{C}$. Using the definition of a star we then find that $\phi(\alpha)$ is a face of τ, which means by Definition 23 that $\phi(\alpha) \in \mathcal{C}$ because it is the face of a simplex in $\mathcal{C}$. □

A **simplicial approximation** of a carrier map $\Phi : \mathcal{A} \to 2^{\mathcal{B}}$ is a map $\phi : \operatorname{Bary}^N \mathcal{A} \to \mathcal{B}$ such that $\phi(\operatorname{Bary}^N \alpha)$ is a subcomplex of $\Phi(\alpha)$ for all $\alpha \in \mathcal{A}$. Note that ϕ is a simplicial approximation of Φ if and only if the carrier map $\phi \circ \operatorname{Bary}^N$ is carried by Φ.

The following theorem shows that a simplicial approximation of a continuous approximation of a carrier map is a simplicial approximation of that carrier map. It resembles [4], Lemma 3.7.8 but it is worded in a more rigorous way. Its proof is based on the last part of the proof of [4], Theorem 3.7.7.

Theorem 37. *Let $\mathcal{A}$ and $\mathcal{B}$ be finite abstract simplicial complexes, let $\Phi : \mathcal{A} \to 2^{\mathcal{B}}$ be a carrier map, and let $f : |\mathcal{A}| \to |\mathcal{B}|$ be a continuous approximation of Φ. If $\phi : \operatorname{Bary}^N \mathcal{A} \to \mathcal{B}$ is a simplicial approximation of f, then it is a simplicial approximation of Φ.*

Proof. Let ϕ be a simplicial approximation of f. Let $\alpha \in \mathcal{A}$ be a simplex. Pick a simplex $\sigma \in \operatorname{Bary}^N \alpha$. Because $\Phi(\alpha)$ is a subcomplex of $\mathcal{B}$ and since we have that $f(|\sigma|) \subseteq |\Phi(\alpha)|$, we can use Lemma 15 to conclude that $\phi(\sigma)$ is not just a simplex of $\mathcal{B}$, but also a simplex of $\Phi(\alpha)$. Since we had chosen σ arbitrarily in $\operatorname{Bary}^N \alpha$, we can conclude that $\phi(\operatorname{Bary}^N \alpha)$ is a subcomplex of $\Phi(\alpha)$. Therefore, ϕ is a simplicial approximation of Φ. □

We can combine Theorem 36 and Theorem 37 into the following statement:

Corollary 38. *Let $\mathcal{A}$ and $\mathcal{B}$ be finite abstract simplicial complexes and let $\Phi : \mathcal{A} \to 2^{\mathcal{B}}$ be a carrier map. Assume that the subcomplex $\Phi(\sigma)$ is $(\dim(\sigma) - 1)$-connected for every simplex $\sigma \in \mathcal{A}$. Then there exists a simplicial approximation $\phi : \operatorname{Bary}^N \mathcal{A} \to \mathcal{B}$ of Φ.*

Proof. From Theorem 36 we know that there exists a continuous approximation $f : |\mathcal{A}| \to |\mathcal{B}|$. Using Theorem 34, we can find a simplicial approximation $\phi : \operatorname{Bary}^N \mathcal{A} \to \mathcal{B}$ of f. From Theorem 37, it then follows that ϕ is a simplicial approximation to Φ as well. □

6 Examples of colorless tasks

In this section, we will discuss several examples of colorless tasks.

6.1 Barycentric agreement

The barycentric agreement task is an important type of task in distributed computing. It is particularly interesting because of Lemma 16, which proves a result about operational models where the iterated barycentric agreement task can be solved.

Definition 39. Let $\mathcal{I}$ be an input complex. The **barycentric agreement task** is the task $(\mathcal{I}, \operatorname{Bary}\mathcal{I}, \operatorname{Bary})$. For any integer $N > 0$ the **iterated barycentric agreement task** is the task $(\mathcal{I}, \operatorname{Bary}^N \mathcal{I}, \operatorname{Bary}^N)$.

6.2 Consensus

In Example 3 we saw how the binary consensus task is a colorless task. In this section we will generalize the binary consensus task in the form of a monotonic colorless task.

We first describe the consensus task operationally. Let S be a finite set. The processes are each given an input value from S and they must all output the same value but they can only return a value that was given to at least one of the processes as input.

The possible colorless inputs are the subsets of S, so we have that $\mathcal{I} = 2^S$. Choose an input simplex $\sigma \in \mathcal{I}$. Then the possible colorless outputs are those with at most one element from σ, so $\Delta(\sigma) = \operatorname{skel}^0 \sigma$. This means that the consensus task can be represented as $(\mathcal{I}, \operatorname{skel}^0 \mathcal{I}, \operatorname{skel}^0)$.

If $\#S = c$, this task is also referred to as c-consensus.

6.3 Set agreement

We can generalize the consensus task even further by talking about k-set agreement. While in the consensus task there can be at most one distinct output value from the different processes, in the k-set agreement task we allow k distinct output values from the different processes. Naturally, 1-set agreement then corresponds to consensus.

The combinatorial definition of k-set agreement is as follows:

$$(\mathcal{I}, \operatorname{skel}^{k-1} \mathcal{I}, \operatorname{skel}^{k-1})$$

Note that, unlike in the consensus task, we allow $\mathcal{I}$ to be any complex and not just the power set of some finite set S.

A k-set agreement task is trivial to solve by an n-process program if $n < k$, because they can all return their input value.

6.4 Approximate agreement

This example is inspired by [4], Exercise 4.4.

Another way to change the consensus task is to allow the processes to return distinct output values as long as they are sufficiently close. Let $\epsilon > 0$ be a real number and let $n > 0$ be an integer. Let $\mathcal{K}$ be a finite geometric simplicial complex. Define $\mathcal{I} := C(\mathcal{K})$.

We define the n-dimensional ϵ-approximate agreement task as follows: given an input simplex $\sigma \in \mathcal{I}$, all processes must return points in the convex hull of σ and the Euclidean distance between any pair of these points must be smaller than ϵ. Let R_ϵ be the collection of finite subsets of $\mathbb{R}^n$ for which the distance between any pair of points is less than ϵ. We can define this problem as a colorless task in the following way:

$$\begin{aligned} \mathcal{O} :=& \{\, B \in R_\epsilon \mid B \subseteq \operatorname{conv} A \text{ for some } A \in \mathcal{I} \,\} \\ \Delta : \mathcal{I} \to 2^{\mathcal{O}}, \qquad \Delta(A) =& \{\, B \in R_\epsilon \mid B \subseteq \operatorname{conv} A \,\} \end{aligned}$$

6.5 Earth agreement

This example is based on [4], Exercise 4.5.

Suppose there are some robots which communicate through a colorless layered protocol. We have a set with 4 elements given by $\tau^3 = \{0, 1, 2, 3\}$. Each robot is assigned to a location in τ^3. In other words, the input complex is $\mathcal{I} := 2^{\tau^3}$. Note that τ^3 is a 3-simplex of $\mathcal{I}$. In the context of this task, the complex $\mathcal{I}$ is sometimes called Earth since the geometric realization of $\mathcal{I}$ is homeomorphic to the closed ball, which resembles Earth.

At the end of the convergence task the robots have to be in at most 3 different locations, but if they were assigned to 2 or fewer different locations at the start, they need to stay in the same position. Formally, the output complex is $\mathcal{O} := \operatorname{skel}^2 \mathcal{I}$ and the carrier map Δ is given by:

$$\Delta(\sigma) = \begin{cases} 2^\sigma & \text{if } \dim \sigma \leqslant 1 \\ \operatorname{skel}^2 \mathcal{I} & \text{if } \dim \sigma \geqslant 2 \end{cases}$$

Recall that the 3-set agreement task is defined as $(\mathcal{I}, \operatorname{skel}^2 \mathcal{I}, \operatorname{skel}^2)$. By distinguishing cases using the dimension of σ, just like we did for defining Δ, we obtain:

$$\operatorname{skel}^2 \sigma = \begin{cases} 2^\sigma & \text{if } \dim \sigma \leqslant 1 \\ 2^\sigma & \text{if } \dim \sigma = 2 \\ \operatorname{skel}^2 \mathcal{I} & \text{if } \dim \sigma = 3 \end{cases}$$

The only difference is in the value of $\Delta(\sigma)$ for 2-dimensional simplices σ. Let σ be a two dimensional simplex. Clearly, $\tau^3 \backslash \sigma$ is a simplex of $\operatorname{skel}^2 \mathcal{I}$ but not of 2^σ. Therefore, $\operatorname{skel}^2 \sigma \subsetneq \operatorname{skel}^2 \mathcal{I} = \Delta(\sigma)$. We conclude that any program that solves k-set agreement also solves Earth agreement but not vice versa. In the next section, we will see that while there is no colorless layered protocol that solves k-set agreement, Earth agreement can even be solved by a colorless single-layer protocol.

7 Solvability of colorless tasks

We want to determine the solvability of colorless tasks within operational models. In other words we want to determine for which colorless tasks there exists a protocol (within the operational model) that solves it.

7.1 Protocol complex lemma

The protocol complex lemma is an important result for proving solvability of colorless tasks in operational models where there exist protocols for all iterated barycentric agreement tasks. It is based on [4], Lemma 4.2.6.

Let $f : |\mathcal{P}| \to |\mathcal{O}|$ be a continous map and let $\Xi : \mathcal{I} \to 2^{\mathcal{P}}$ be a carrier map. Then we can compose them if we take geometric realizations of the subcomplexes of $\mathcal{P}$ that are a result of Ξ. We denote this **composition** by $f \circ \Xi$, which is then defined by:

$$(f \circ \Xi) : \mathcal{I} \to 2^{|\mathcal{O}|}, \qquad (f \circ \Xi)(\sigma) = f(|\Xi(\sigma)|)$$

Lemma 16. Suppose that, within some operational model, for any input complex $\mathcal{I}$ and any integer $N > 0$, there exists a protocol that solves the iterated barycentric agreement task, which is given by $(\mathcal{I}, \operatorname{Bary}^N \mathcal{I}, \operatorname{Bary}^N)$. Then a task $(\mathcal{I}, \mathcal{O}, \Delta)$ has a protocol (in that operational model) if and only if there exists a protocol $(\mathcal{I}, \mathcal{P}, \Xi)$ (in that operational model) with a continuous map $f : |\mathcal{P}| \to |\mathcal{O}|$ such that $f \circ \Xi$ is carried by Δ.

Proof. First, assume that there exists a protocol $(\mathcal{I}, \mathcal{P}, \Xi)$ that solves the task $(\mathcal{I}, \mathcal{O}, \Delta)$. Then by definition, there exists a simplicial decision map $\delta : \mathcal{P} \to \mathcal{O}$ such that $\delta \circ \Xi$ is carried by Δ. This then induces a continuous map $|\delta| : |\mathcal{P}| \to |\mathcal{O}|$ because of Lemma 8. Since $\delta \circ \Xi$ is carried by Δ, $|\delta| \circ \Xi$ is carried by Δ as well.

Now, assume there exists a protocol $(\mathcal{I}, \mathcal{P}, \Xi)$ with a continuous map $f : |\mathcal{P}| \to |\mathcal{O}|$ such that $f \circ \Xi$ carried by Δ. Because of Theorem 34, there must be a simplicial approximation $\phi : \operatorname{Bary}^N \mathcal{P} \to \mathcal{O}$ of f for some $N > 0$. By hypothesis, there is a protocol that solves the task $(\mathcal{P}, \operatorname{Bary}^N \mathcal{P}, \Xi)$. We can compose this with the protocol Ξ to obtain a protocol $(\mathcal{I}, \operatorname{Bary}^N \mathcal{P}, (\operatorname{Bary}^N \circ \Xi))$.

Note that $\phi \circ \operatorname{Bary}^N \circ \Xi$ is a simplicial appoximation of $f \circ \Xi$. Because $f \circ \phi$ is carried by Δ, it follows from Theorem 37 that $\phi \circ \operatorname{Bary}^N \circ \Xi$ is also carried by Δ. Therefore the protocol $(\mathcal{I}, \operatorname{Bary}^N \mathcal{P}, (\operatorname{Bary}^N \circ \Xi))$ solves the task $(\mathcal{I}, \mathcal{O}, \Delta)$. □

This also has the following consequence, which is a stronger version of [4], Lemma 4.2.7:

Lemma 17. Suppose that, within some operational model, for any input complex $\mathcal{I}$ and any integer $N > 0$, there exists a protocol that solves the iterated barycentric agreement task, which is given by $(\mathcal{I}, \operatorname{Bary}^N \mathcal{I}, \operatorname{Bary}^N)$. Let $(\mathcal{I}, \mathcal{O}, \Delta)$ be a colorless task and let Div be any subdivision of $\mathcal{O}$. Then $(\mathcal{I}, \operatorname{Div} \mathcal{O}, \operatorname{Div} \circ \Delta)$ has a protocol if and only if the task $(\mathcal{I}, \mathcal{O}, \Delta)$ has a protocol.

Proof. Let $(\mathcal{I}, \mathcal{P}, \Xi)$ be a protocol. Because $|\mathcal{O}| = |\operatorname{Div} \mathcal{O}|$, any continuous function of the form $|\mathcal{P}| \to |\operatorname{Div} \mathcal{O}|$ is also a continuous function of the form $|\mathcal{P}| \to |\mathcal{O}|$ and vice versa. Assume f is such a function.

Note that $|\tau| = |\operatorname{Div} \tau|$ for all $\tau \in \mathcal{O}$, hence $|\Delta(\sigma)| = |\operatorname{Div}(\Delta(\sigma))|$. By definition $f \circ \Xi$ as function of the form $\mathcal{I} \to |\mathcal{O}|$ is carried by Δ if and only if $f(|\Xi(\sigma)|) \subseteq |\Delta(\sigma)|$. Similarly, By definition $f \circ \Xi$ as function of the form $\mathcal{I} \to |\operatorname{Div} \mathcal{O}|$ is carried by $\operatorname{Div} \circ \Delta$ if and only if $f(|\Xi(\sigma)|) \subseteq |\operatorname{Div}(\Delta(\sigma))| = |\Delta(\sigma)|$, so $f \circ \Xi$ is carried by Δ if and only if $f \circ \Xi$ is carried by $\operatorname{Div} \circ \Delta$.

Hence, by Lemma 16, we have that $(\mathcal{I}, \operatorname{Div}\mathcal{O}, \operatorname{Div}\circ\Delta)$ has a protocol if and only if the task $(\mathcal{I}, \mathcal{O}, \Delta)$ has a protocol. □

7.2 Solvability by colorless layered protocols

Note that for an $(n+1)$-process protocol, there can be at most $n+1$ different input values in a colorless input. Hence, $\mathcal{I} = \operatorname{skel}^n \mathcal{I}$. This theorem is a modified version of [4], Theorem 4.2.8, inspired by [4], Exercise 4.14.

Theorem 40. *Let $(\mathcal{I}, \mathcal{P}, \Xi)$ be a colorless single-layer $(n+1)$-process atomic snapshot protocol. Then $\mathcal{P} = \operatorname{Bary}\mathcal{I}$ and $\Xi(\sigma) = \operatorname{Bary}\sigma$.*

Proof. In a single-layer execution, each process first writes its input value to its part of the shared memory and then takes a snapshot of the shared memory. So if τ is a final state of some process, then it is a set containing the values that were read from the shared memory. Because the processes only write input values to the shared memory, τ is a subset of the execution's colorless input simplex. This means that τ is an input simplex. Hence, the final colorless state (which, by definition, is the collection of final states) is a collection of input simplices.

Note that processes only write to their own memory once. In particular, once a value is written to the shared memory, it will stay there for the remainder of the execution. As a consequence, the set of input values written to the shared memory only gets larger over time.

Pick $\sigma \in \mathcal{I}$. We will now show that $\Xi(\sigma) = \operatorname{Bary}\sigma$.

Consider an execution of the protocol with colorless input σ. Pick any $0 \leqslant i, j \leqslant n$ such that P_i and P_j are both non-crashing processes, where P_i takes snapshot τ_i and P_j takes snapshot τ_j. Because the snapshots are atomic, either i takes the snapshot before j or the other way around or they take snapshots at the same time. In the first case, we have that $\tau_i \subseteq \tau_j$ and in the second case we have that $\tau_j \subseteq \tau_i$. When the snapshots are taken at the same time, they are equal.

The discussion above proves that the the snapshots in the final colorless state can be reordered into a chain of growing snapshots. Since the snapshots are subsets of σ, it follows that the final colorless state is a subset of the barycentric subdivision $\operatorname{Bary}\sigma$. In other words, $\Xi(\sigma) \subseteq \operatorname{Bary}\sigma$. We still need to prove the other inclusion.

Pick $\tau \in \operatorname{Bary}\sigma$. We will now show that there exists an execution of the protocol with colorless input σ and colorless final state τ. Because of the definition of the barycentric subdivision, we can write $\tau = \{\tau_0, \tau_1, \ldots, \tau_m\}$ where $\tau_0, \ldots, \tau_m \subseteq \sigma$ and $\varnothing \neq \tau_0 \subseteq \tau_1 \subseteq \cdots \subseteq \tau_m$. Now consider the execution with colorless input τ_m in which no processes crash and where the events unfold in the following order:

1. The processes that received input values in τ_0 write their input values to the shared memory.
2. The processes that received input values in τ_0 take a snapshot.
3. The processes that received input values in $\tau_1 \backslash \tau_0$ write their input values to the shared memory.
4. The processes that received input values in $\tau_1 \backslash \tau_0$ take a snapshot.

   ⋮

$2m+1$. The processes that received input values in $\tau_m \backslash \tau_{m-1}$ write their input values to the shared memory.

$2m+2$. The processes that received input values in $\tau_m \backslash \tau_{m-1}$ take a snapshot.

Pick any $1 \leqslant t \leqslant 2m+2$. Note that after step $2t+1$, the processes with input values in τ_t have all written their input values to memory. Since τ_t is a subset of τ, there exists a process with input value i for each input value i in τ_t. Therefore, any snapshot taken at step $2t+2$ is precisely τ_t.

We now prove by induction on t that τ_t is in the final colorless state for any $1 \leqslant t \leqslant 2m+2$. Note that τ_0 is non-empty by definition so there is a process that takes a snapshot at step 2, which means that τ_0 is in the final colorless state. Let $t > 0$ and assume that τ_{t-1} is in the final colorless state. If $\tau_t \backslash \tau_{t-1}$ is empty, then $\tau_t = \tau_{t-1}$ and by hypothesis, τ_{t-1} is in the final colorless state. If $\tau_t \backslash \tau_{t-1}$ is non-empty, then there is a process that takes a snapshot at step $2t+2$. This snapshot is then τ_t, which means that τ_t is in the final colorless state.

Since all $\tau_t \in \tau$ are in the final colorless state and since any snapshot that is taken is of the form τ_t, we have that τ is the final colorless state. Because τ was chosen arbitrarily, we have that $\operatorname{Bary}\sigma \subseteq \Xi(\sigma)$. Combining this with what we found earlier in the proof, we have that $\operatorname{Bary}\tau = \Xi(\sigma)$. This also means that $\mathcal{P} = \bigcup \Xi(\mathcal{I}) = \operatorname{Bary}\mathcal{I}$. □

As explained in Section 2.5, every colorless protocol can be seen as repeated single-layer protocols. Therefore we find the following result by induction on N:

Theorem 41. *Let $(\mathcal{I}, \mathcal{P}, \Xi)$ be a colorless N-layered $(n+1)$-process atomic snapshot protocol. Then $\mathcal{P} = \operatorname{Bary}^N \mathcal{I}$ and $\Xi(\sigma) = \operatorname{Bary}^N \sigma$.*

We then also have the following corollary:

Corollary 42. *Let $\mathcal{I}$ be an input complex of dimension n and let $N > 0$ be an integer. Then there is an $(n+1)$-process colorless layered protocol that solves the barycentric agreement task $(\mathcal{I}, \operatorname{Bary}^N \mathcal{I}, \operatorname{Bary}^N)$.*

This then leads us to arguably the most important result of this thesis:

Theorem 43. *Let $(\mathcal{I}, \mathcal{O}, \Delta)$ be a colorless task. Then there exists an $n+1$-process colorless layered protocol that solves $(\mathcal{I}, \mathcal{O}, \Delta)$ if and only if there exists a continous map $f : |\operatorname{skel}^n \mathcal{I}| \to |\mathcal{O}|$ carried by Δ.*

Proof. Note that any execution of an $(n+1)$-process program has at most $n+1$ distinct input values, so its colorless input simplex is at most of dimension n. This means that $(\mathcal{I}, \mathcal{O}, \Delta)$ is equivalent to $(\operatorname{skel}^n \mathcal{I}, \mathcal{O}, \Delta)$. Note that $\operatorname{skel}^n \mathcal{I}$ is of dimension n.

Because of Corollary 42 and since compositions of colorless layered protocols are colorless layered protocols, we can then apply Lemma 16 to obtain that there exists a colorless layered protocol that solves $(\operatorname{skel}^n \mathcal{I}, \mathcal{O}, \Delta)$ if and only if there exists a colorless layered protcol $(\operatorname{skel}^n \mathcal{I}, \mathcal{P}, \Xi)$ with a continuous map $f : |\mathcal{P}| \to |\mathcal{O}|$ such that $f \circ \Xi$ is carried by Δ.

Because of Theorem 41, we have that $\mathcal{P} = \operatorname{Bary}^N \operatorname{skel}^n \mathcal{I}$ and $\Xi = \operatorname{Bary}^N \circ \operatorname{skel}^n$ for some $N > 0$. Hence, because Bary^N is a subdivision, it is the case that $|\mathcal{P}| = |\operatorname{Bary}^N \operatorname{skel}^n \mathcal{I}| = |\operatorname{skel}^n \mathcal{I}|$ and $(f \circ \Xi)(\sigma) = f(|\Xi(\sigma)|) = f(|\operatorname{Bary}^N \operatorname{skel}^n \sigma|) = f(|\operatorname{skel}^n \sigma|) = (f \circ \operatorname{skel}^n)(\sigma)$. Note that $(f \circ \operatorname{skel}^n)$ is carried by Δ if and only if f is carried by Δ.

We conclude that there exists an $(n+1)$-process colorless layered protocol that solves $(\mathcal{I}, \mathcal{O}, \Delta)$ if and only if there exists a continous map $f : |\operatorname{skel}^n \mathcal{I}| \to \mathcal{O}$ carried by Δ. □

Sometimes we want to prove that a task can be solved by a program with any number of processes. In that case, the following result may be useful:

Lemma 18. Let $(\mathcal{I}, \mathcal{O}, \Delta)$ be a colorless task where $\mathcal{I}$ is a finite. Then there exists an n-process colorless layered protocol that solves $(\mathcal{I}, \mathcal{O}, \Delta)$ for all $n > 0$ if and only if there exists a continous map $f : |\mathcal{I}| \to |\mathcal{O}|$ carried by Δ.

Proof. Suppose there exists a continuous map $f : |\mathcal{I}| \to |\mathcal{O}|$ carried by Δ. Pick any $n > 0$. Then, there the restriction $f|_{|\operatorname{skel}^{n-1}\mathcal{I}|} : |\operatorname{skel}^{n-1}\mathcal{I}| \to |\mathcal{O}|$ is a continuous map carried by Δ, so by Theorem 43, we have that there exists an n-process protocol that solves $(\mathcal{I}, \mathcal{O}, \Delta)$.

For the converse, suppose that there exists an $\#\mathcal{I}$-process colorless layered protocol that solves $(\mathcal{I}, \mathcal{O}, \Delta)$. Then, by Theorem 43, there must be a continuous map $f : |\operatorname{skel}^{\#\mathcal{I}}\mathcal{I}| \to |\mathcal{O}|$ carried by Δ. Since any simplex in $\mathcal{I}$ has a dimension less than $\#\mathcal{I}$, we have that $\operatorname{skel}^{\#\mathcal{I}}\mathcal{I} = \mathcal{I}$, so f is a continuous map from $|\mathcal{I}|$ to $|\mathcal{O}|$ and it is carried by Δ. □

7.3 Applications

7.3.1 Set agreement

Pick $k > 0$ and $n > 0$. Recall that k-set agreement is the task $(\mathcal{I}, \operatorname{skel}^{k-1}\mathcal{I}, \operatorname{skel}^{k-1})$. Because of Theorem 43, there exists an n-process colorless layered protocol that solves the task if and only if there exists a continuous map $f : \operatorname{skel}^n \mathcal{I} \to \operatorname{skel}^{k-1}\mathcal{I}$. If $k > n$, then $k - 1 \geqslant n$, so finding a continuous map $\operatorname{skel}^n \mathcal{I} \to \operatorname{skel}^{k-1}\mathcal{I}$ is easy because $\operatorname{skel}^n \mathcal{I} \subseteq \operatorname{skel}^{k-1}\mathcal{I}$, which means that we can just use the identity map.

To prove the solvability when $k \leqslant n$, we can make use [4], Fact 4.3.4, which states the following.

Lemma 19 (Sperner's Lemma)**.** There is no simplicial map ϕ from a subdivision $\operatorname{Div}\sigma$ to $\operatorname{skel}^{n-1}\sigma$ carried by the carrier map $\operatorname{Car}(\cdot, \operatorname{Div}\sigma)$.

Now, it follows from Theorem 34 that there exists an $N > 0$ and a simplicial map $\phi : \operatorname{Bary}^N \operatorname{skel}^n \mathcal{I} \to \operatorname{skel}^{k-1}\mathcal{I}$ if a continuous map $f : \operatorname{skel}^n \mathcal{I} \to \operatorname{skel}^{k-1}\mathcal{I}$ exists. For any k-dimensional simplex $\sigma \in \mathcal{I}$, we would then have that $\phi|_{2^\sigma}$ is a simplicial map carried by $\operatorname{skel}^{k-1}$, which satisfies the condition of Sperner's lemma. This is clearly a contradiction, so we conclude that a continuous map $f : \operatorname{skel}^n \mathcal{I} \to \operatorname{skel}^{k-1}\mathcal{I}$ cannot exist. As a result of Theorem 43, we find that there exists no colorless layered protocol that solves the task $(\mathcal{I}, \operatorname{skel}^{k-1}\mathcal{I}, \operatorname{skel}^{k-1})$ if $k \leqslant n$.

7.3.2 Approximate agreement

This section is a solution to Exercise 4.4. Recall that the approximate agreement task was defined as follows:

$$
\begin{aligned}
&\mathcal{I} := C(\mathcal{K}) \\
&\mathcal{O} := \{\, B \in R_\epsilon \mid B \subseteq \operatorname{conv} A \text{ for some } A \in \mathcal{I} \,\} \\
&\Delta : \mathcal{I} \to 2^{\mathcal{O}}, \qquad \Delta(A) = \{\, B \in R_\epsilon \mid B \subseteq \operatorname{conv} A \,\}
\end{aligned}
$$

where R_ϵ was the collection of finite sets in $\mathbb{R}^n$ with diameter less than ϵ and $\mathcal{K}$ was a finite geometric simplicial complex.

Note that there exists an $N > 0$ such that for any simplex $\sigma \in \operatorname{Bary}^N \mathcal{K}$, the geometric realization $|\sigma|$ has diameter less than $\frac{1}{2}\epsilon$. Note that $\operatorname{Bary}^N \circ C : \mathcal{I} \to \operatorname{Bary}^N \mathcal{K}$ is carried by Δ. Choose any decision map $\delta : \operatorname{Bary}^N \mathcal{K} \to \mathcal{O}$. Then $\delta \circ \operatorname{Bary}^N \mathcal{K} \circ C$ is carried by Δ. Hence there exists a colorless layered protocol that solves the approximate agreement task.

7.3.3 Earth agreement

In this section we will show that Earth agreement as defined in Section 6.5 can be solved by a colorless single-layer protocol. To do so, we must find a simplicial decision map $\delta : \operatorname{Bary}\mathcal{I} \to \mathcal{O}$ such that $\delta \circ \operatorname{Bary}$ is carried by Δ. Choose $\sigma \in \operatorname{Bary}\mathcal{I}$.

First we define $\delta : V(\operatorname{Bary}\mathcal{I}) \to V(\mathcal{O})$ as a vertex map. Note that, by definition, $V(\operatorname{Bary}\mathcal{I}) = \mathcal{I}\backslash\{\varnothing\}$ and $V(\mathcal{O}) = V(\operatorname{skel}^2\mathcal{I}) = V(\mathcal{I}) = \tau^3$. Choose $\sigma \in V(\operatorname{Bary}\mathcal{I}) = \mathcal{I}\backslash\{\varnothing\}$. Note that $\operatorname{Bary}\sigma$ is well-defined because σ is a simplex of $\mathcal{I}$. In particular, we have that $\{\sigma\} \in \operatorname{Bary}\sigma$. It follows that:

$$\{\delta(\sigma)\} = \delta(\{\sigma\}) \in \delta(\operatorname{Bary}\sigma) = (\delta \circ \operatorname{Bary})(\sigma)$$

Since we want to have that $(\delta \circ \operatorname{Bary})(\sigma) \subseteq \Delta(\sigma)$, we should ensure that $\delta(\sigma) \in V(\Delta(\sigma))$. If $\dim\sigma \leqslant 1$, that means that $\delta(\sigma) \in V(\Delta(\sigma)) = V(2^\sigma) = \sigma$. If $\dim\sigma > 1$, we just need to satisfy $\delta(\sigma) \in V(\Delta(\sigma)) = V(\operatorname{skel}^2\mathcal{I}) = V(\mathcal{I}) = V(\mathcal{O})$. Now, define $\delta : V(\operatorname{Bary}\mathcal{I}) \to V(\mathcal{O})$ as follows:

$$\delta(\sigma) = \begin{cases} \min\sigma & \text{if } \dim\sigma \leqslant 1 \\ 0 & \text{if } \dim\sigma > 1 \end{cases}$$

We will now prove that $\delta \circ \operatorname{Bary}$ is a simplicial map carried by Δ. Pick any simplex $\tau \in \mathcal{I}$. Pick a simplex in $\operatorname{Bary}\tau$. By definition, this simplex consists of input simplices $\sigma_0, \ldots, \sigma_m \subseteq \tau$ such that $\varnothing \neq \sigma_0 \subsetneq \cdots \subsetneq \sigma_m$. Note that $\dim\sigma_i \geqslant i$ for all i. We distinguish two cases: either $\dim\sigma_m \leqslant 1$ or $\dim\sigma_m > 1$.

First, assume $\dim\sigma_m \leqslant 1$. Then we have that:

$$\delta(\{\sigma_0, \ldots, \sigma_m\}) = \{\delta(\sigma_0), \ldots, \delta(\sigma_m)\} = \{\min\sigma_0, \ldots, \min\sigma_m\} \subseteq \bigcup_{0\leqslant i\leqslant m} \sigma_i \subseteq \tau$$

Hence, $\delta(\{\sigma_0, \ldots, \sigma_m\}) \in 2^\tau = \Delta(\tau)$.

Now, assume $\dim\sigma_m > 1$. Choose $-1 \leqslant j \leqslant m$ such that $\sigma_0, \ldots, \sigma_j$ have a dimension of at most 1 and such that $\sigma_{j+1}, \ldots, \sigma_m$ have a dimension greater than 1. Note that $j \leqslant 1$ and $m > j$, because the dimensions of the σ_i are strictly increasing. Since $\delta(\sigma) = \delta(\{\sigma_0, \ldots, \sigma_m\}) = \{\delta(\sigma_0), \ldots, \delta(\sigma_m)\} = \{\min\sigma_0, \ldots, \min\sigma_j\} \cup \{\min V(\mathcal{I})\}$, we find that:

$$\begin{aligned} \dim\delta(\sigma) &= \#\delta(\sigma) - 1 \\ &= \#(\{\min\sigma_0, \ldots, \min\sigma_j\} \cup \{\min V(\mathcal{I})\}) - 1 \\ &\leqslant \#(\{\min\sigma_0, \ldots, \min\sigma_j\}) + \#(\{\min V(\mathcal{I})\}) - 1 \\ &\leqslant 2 + 1 - 1 = 2 \end{aligned}$$

It follows from $\dim\delta(\sigma) \leqslant 2$ that $\delta(\sigma) \in \operatorname{skel}^2 2^{V(\mathcal{O})} = \operatorname{skel}^2\mathcal{I} = \Delta(\tau)$.

In both cases we have that $\delta(\sigma) \in \Delta(\tau)$, so we conclude that $(\delta \circ \operatorname{Bary})(\tau) = \bigcup_{\sigma\in\operatorname{Bary}\tau} \delta(\sigma) \subseteq \Delta(\tau)$, which makes δ a simplicial map carried by Δ. This means that the Earth agreement task can indeed be solved by a colorless single-layer protocol.

Operationally, the protocol and decision map above correspond to the following pseudocode:

```
const int numProcesses;
const int currentProcessIndex;
extern Value[numProcesses] memory; // all values start at -1
```

```
// input is 0, 1, 2 or 3
// return value is 0, 1, 2 or 3
Value solveEarthAgreementTask(Value input) {
    // colorless single-layer atomic snapshot protocol
    update(memory[currentProcessIndex], input);
    Value[numProcesses] snapshot = scan(memory);
    set<Value> locations = setOfValues(snapshot);

    // note that locations is non-empty
    // because we wrote our own input first

    // make decision for return value
    if (size(locations) < 2) {
        return min_element(locations);
    } else {
        // 0 is a valid location
        // any other valid location would also work
        // as long as it is the same constant for all processes
        return 0;
    }
}
```

This paragraph is dedicated to solving the remainder of [4], Exercise 4.5. Let Div be any subdivision of $\mathcal{O}$. From Lemma 17 it follows that $(\mathcal{I}, \mathrm{Div}\,\mathcal{O}, \mathrm{Div} \circ \Delta)$ is also solvable by a colorless layered protocol. From Lemma 17 it follows that $(\mathcal{I}, \mathrm{Div} \circ \mathrm{skel}^2\, O, \mathrm{Div} \circ \mathrm{skel}^2)$ is not solvable by a colorless layered protocol if the number of processes is greater than 3 since skel^2 is not solvable in that case.

References

[1] Yehuda Afek, Hagit Attiya, Danny Dolev, Eli Gafni, Michael Merritt, and Nir Shavit. Atomic snapshots of shared memory. *Journal of the ACM (JACM)*, 40(4):873–890, 1993.

[2] Elizabeth Borowsky and Eli Gafni. Immediate atomic snapshots and fast renaming. In *Proceedings of the twelfth annual ACM symposium on Principles of distributed computing*, pages 41–51. ACM, 1993.

[3] Tzilla Elrad and Nissim Francez. Decomposition of distributed programs into communication-closed layers. *Science of Computer Programming*, 2(3):155–173, 1982.

[4] Maurice Herlihy, Dmitry Kozlov, and Sergio Rajsbaum. *Distributed computing through combinatorial topology*. Morgan Kaufmann, 2014.

[5] James Munkres. *Elements of algebraic topology*. The Benjamin/Cummings Publishing company, 1984.

Index