UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO
MAESTRÍA EN CIENCIA E INGENIERÍA DE LA COMPUTACIÓN

ESPECIFICACIÓN DE TAREAS CONTÍNUAS EN MODELOS ASÍNCRONOS DE MEMORIA COMPARTIDA Y LA TOPOLOGÍA DE ARGUMENTOS DE PARTICIÓN

TESIS
QUE PARA OPTAR POR EL GRADO DE
MAESTRO EN CIENCIAS E INGENIERÍA DE LA COMPUTACIÓN

PRESENTA:
HUGO RINCÓN GALEANA

TUTORES PRINCIPALES
DR. SERGIO RAJSBAUM GORODEZKY
INSTITUTO DE MATEMÁTICAS, UNAM

DR. ULRICH SCHMID
EMBEDDED COMPUTING SYSTEMS GROUP, TUWIEN

CIUDAD UNIVERSITARIA, CD. MX. MAYO 2019

UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

Abstract

Posgrado en Ciencia e Ingeniería en Computación
Instituto de Matemáticas

Maestro en Ciencia e Ingeniería de la Computación

Continuous Task Specification in Asynchronous Shared Memory Models And the Topology of Partitioning Arguments

by Hugo RINCÓN GALEANA

This thesis further explores the relationship between combinatorial topology and distributed systems. This relationship was first introduced in the ACM STOC 1993 conference, where combinatorial topology was presented as an useful framework to understand better distributed tasks solvability. Results such as the central Asynchronous Computability Theorem (ACT) of Herlihy and Shavit, which characterizes solvable tasks in read/write shared memory where any number of asynchronous processes can crash, have established combinatorial topology as an essential tool.

In this thesis, we present two results, which are both strongly connected to the topological framework for distributed computing.

We reformulate the ACT in terms of continuous tasks, a new notion we propose here. Continuous tasks are defined through continuous functions that preserve some of the chromatic structure.

A second result is a topological formulation for the BRS theorem, an impossibility result for k-set agreement that captures the essence of general partitioning arguments. We also show that in general, partitioning arguments are too weak to show set agreement in the asynchronous shared-memory model by giving an algorithm that solves set agreement for any run that admits a partitioning argument in the asynchronous shared-memory model.

Acknowledgements

I want to thank my advisors,Sergio Rajsbaum and Ulrich Schmid, and my reviewers, Marcelo Aguilar, Armando Castañeda and David Flores, for their invaluable support and guidance.

I also want to thank all of my teachers for everything that I could learn from them.

Contents

Abstract iii

Acknowledgements v

1 Introduction 1
- 1.1 Background . . . 1
- 1.2 Motivation . . . 2
- 1.3 Results . . . 2
- 1.4 Organization . . . 3

2 Puzzles and Simplicial Complexes 5
- 2.1 The muddy children problem . . . 5
- 2.2 Cheryl's birthday puzzle . . . 6

3 The Asynchronous Shared Memory Model 11
- 3.1 Model Specification . . . 11
- 3.2 States, actions, executions . . . 12
- 3.3 Tasks . . . 12

4 Combinatorial Topology 15
- 4.1 Basic Concepts . . . 15
- 4.2 Subdivisions and Communication . . . 20

5 The ACT and Simplicial Approximation 23
- 5.1 The Asynchronous Computability Theorem . . . 23
- 5.2 Simplicial approximation . . . 24

6 Continuous Tasks 29
- 6.1 Contributions . . . 31
- 6.2 The Asynchronous Wait-free Shared Memory Model . . . 31
- 6.3 Chromatic Functions and Continuous Tasks . . . 32
- 6.4 Proofs for lemmas . . . 37

7 Message Passing and BRS Theorem 39
- 7.1 Model and preliminary definitions . . . 39
- 7.2 Topological BRS Theorem . . . 45
- 7.3 Shared memory model . . . 48
- 7.4 Partition compatible runs . . . 49

8 Conclusions **53**
8.1 Summary 53
8.2 Future Work 53

To my family and friends.

The few and true who stayed until the end.

Chapter 1

Introduction

1.1 Background

Distributed tasks can be defined through a set of valid inputs, a set of valid outputs and a mapping that specifies which output configurations are valid with respect to an input configuration. In distributed computing the main difficulty for solving tasks is usually not the processing power of each process (as opposed to parallel or sequential computing), but it is more frequently related to communication failures, either in the interconnecting network or through a process failure by itself.

Since communication is assumed to be unreliable (with some model restrictions), protocols cannot fully depend on the global state of the system, but should depend only on the partial information that is acquired by each process. Since protocols should decide with incomplete information, they should take advantage of the model's communication assumptions. It has been discovered in 1993 [2] that there is a deep connection between distributed computability and algebraic topology. An overview of this area appears in [33].

The topological objects known as chromatic simplicial complexes allow us to model the incomplete information observed by each process. A chromatic simplicial complex consists of a set of colored vertices and a descending set of faces. Each vertex corresponds to the "point of view" of a process, while a face consists of a valid configuration of such points of view. The color for each vertex corresponds to the process id for such a point of view.

Given a set of valid inputs, we can build a simplicial complex that captures the valid input configurations. Symmetrically, another simplicial complex can be built for the output configurations. These complexes are known as the Input Complex and the Output Complex (denoted by $\mathcal{I}$ and $\mathcal{O}$ respectively). In this way, we can state a task in terms of two simplicial complexes and a map between the faces of $\mathcal{I}$ and the power set of faces of $\mathcal{O}$. A protocol complex $\mathcal{P}$ can also be built in the same way by using the final states for each process and their valid configurations instead of input or output values. Notice that communication and model conditions define the way that the input complex $\mathcal{I}$ is transformed into the protocol complex $\mathcal{P}$ [33].

Although these complexes can be formulated for any distributed computing model [10], we will focus on the asynchronous shared memory model with crash failures. In this model, each process communicates by writing into a segment of a shared memory. Any process can read any segment of the shared memory, but it may only write on its reserved segment. A crashing process may stop at any time of its execution and will never recover, thus it is not required to decide. Since we are considering an asynchronous model, each process runs at its own speed and may be delayed by an arbitrary amount of time. Therefore in this model, a crashed process is indistinguishable from a sufficiently delayed process.

Communication rounds of this model subdivide the Input Complex $\mathcal{I}$. Therefore, the Protocol Complex $\mathcal{P}$ corresponds to a chromatic subdivision of $\mathcal{I}$. This allows task solvability for this model to be characterized in terms of the Input Complex. This characterization is given by the Asynchronous Computability Theorem(ACT) [2].

1.2 Motivation

Our main goal is to study task solvability for distributed systems. In particular we will focus on the asynchronous shared memory model, since the protocol complex for this model corresponds to a subdivision of the Input Complex $\mathcal{I}$.

Since our focus is task solvability, rather than message or time complexity, we will consider that our computing model is the iterated immediate snapshot model. This model is equivalent to the general asynchronous wait-free shared memory model [20], but it is simpler since each process only has two operations and the protocol complex is given by the standard k-th chromatic subdivision, where k is the number of communication rounds defined for the protocol.

Since both the ACT [2] and the Simplicial Approximation Theorem can be used in conjunction to characterize topologically colorless tasks [33], our goal is now to extend this characterization to include colored tasks. Recall that in colorless tasks, a valid output is given only by the set of decision values, while in a colored task, it is also important to consider which process decides which value. The ACT [3] provides a characterization of the wait-free read/write memory solvability of colored tasks. One of our goals is to provide an alternate proof (of "only if" direction, the hardest one), which we believe is simpler. This extension is given in terms of chromatic functions, which are continuous functions that preserve some of the color structure of the input complex.

Another goal was to give a topological formulation for general partitioning arguments for the k-set agreement impossibility. Partitioning arguments show that k-set agreement is impossible for a non empty set of runs where communication is segmented and processes can be partitioned into sets with different information.

A first contribution towards this goal is a topological statement of a general partitioning argument known as the BRS Theorem [4], by generalizing this result, we could show that a partitioning argument is not always suitable for proving impossibilities in the Iterated Immediate Snapshot model.

1.3 Results

In this thesis we present two results

We reformulate the ACT [3] in terms of continuous tasks, a new notion we propose here. Continuous tasks are defined through continuous functions that preserve some of the chromatic structure. Such continuous functions are called chromatic functions. Any chromatic function $f : |\mathcal{I}| \to |\mathcal{O}|$ has a chromatic simplicial map $\mu : Sub(\mathcal{I}) \to \mathcal{O}$ that is sufficiently similar to f, but is defined over a chromatic subdivision of $\mathcal{I}$.

Such simplicial maps and subdivisions correspond to a protocol in the asynchronous shared memory model, according to the ACT. Reciprocally any chromatic simplicial map given by a protocol that solves a task induces a continuous chromatic function $|\mu|$. This correspondence allows us to formulate the general ACT in terms of continuous tasks.

A second result is a topological formulation for the BRS theorem [4], an impossibility result for k-set agreement that captures the essence of general partitioning arguments. The result is initially given in terms of indistinguishable runs. In order to characterize this result in terms of topology, we introduce the concept of D-view embeddings which is equivalent to run indistinguishability. We show that k-set agreement is impossible if the protocol complex D-view embeds a protocol complex that cannot solve k-set agreement.

We also show that, in general, partitioning arguments are not enough to show set-agreement impossibility in the asynchronous shared-memory model by giving an algorithm that solves set agreement for any run that admits a partitioning argument in the asynchronous shared-memory model. For this algorithm we use the fact that, in any run where a partitioning argument is valid, there exists some process that fails to receive some information at some point of the run. Therefore, the perfectly reliable communication configuration is not present, and we can then design a protocol that solves set-agreement.

1.4 Organization

The thesis chapter structure is the following:

We start with a puzzles Chapter 2, which uses simplicial complexes to model the individual points of view for each process as an example . In this chapter we also show how communication modifies the points of view for each process by giving the simplicial complex for the points of view at different stages. This illustrates very well that a protocol transforms the initial input complex from an uncertain state into a simplicial complex where it is safe to make a decision.

In Chapter 3 we give the formal definition for a distributed task, and the asynchronous shared-memory model. We also show the connection between this model and the combinatorial topology framework by stating the Asynchronous Computability Theorem. This theorem allows us to state task solvability for the asynchronous shared memory model only in terms of the input and output complexes.

The formal definitions and preliminary results about combinatorial topology are given in Chapter 4. The rigorous mathematical definition is given for a chromatic simplicial complex and for the topological spaces that these objects induce.

In Chapter 5 we elaborate on the relationship between the ACT and simplicial approximation. This is in a way an introduction to our first result, which extends the simplicial approximation theorem to a colored version of the theorem.

In Chapter 6 we give our first result, a chromatic version of the simplicial approximation theorem. This chromatic approximation allows us to state tasks in terms of continuous functions that preserve the basic chromatic structure of the input complex, i.e., *chromatic functions*. We prove that any solvable task is equivalent to a chromatic function between the input complex and the output complex. We also give some of the advantages of using a continuous task specification.

Our next results in Chapter 7 consist of giving a topological formulation for a generalized partition-based impossibility result. These results give a topological perspective of the partition-based BRS theorem. An interesting property about this result is that it proves k-set agreement impossibility for asynchronous message passing models without using the Sperner's Lemma or any topological framework. However, we also show that a partitioning argument (in particular the BRS theorem) is too weak to show set-agreement impossibility for the asynchronous shared memory model in general.

Finally in Chapter 8 we summarize briefly our results, future work, including related open questions.

Chapter 2

Puzzles and Simplicial Complexes

In this chapter we will model two puzzles with the simplicial complexes in order to illustrate the combinatorial topology framework. The model represents all possible configurations for a given puzzle. Vertices represent specific views for an agent, while edges and faces represent possible worlds.

It is worth noting that as communication happens, the possible worlds change, as each of the agents gain new information regarding their world. This is reflected as a transformation of the simplicial complex model. In particular for these puzzles, as each piece of information is gained, possible worlds are removed from the model.

This shows that puzzle solving strategies eventually transform the simplicial complex model into a trivial (one triangle or "simplex") complex. This trivial simplicial complex is the real configuration for the puzzle.

The puzzles that we are considering, is the classic muddy children problem, and a birthday guessing puzzle. While both can be solved by using logic and deduction, we solved them here with the topological model in order to familiarize the reader with simplicial complexes.

2.1 The muddy children problem

To further illustrate the topological approach, we use the following problem: Suppose that some children are playing in a muddy playground. At the end of their playtime, some of them have dirty foreheads and some are clean. Their teacher tells them publicly that at least one of them is dirty, and that, without communicating with each other, all muddy children must announce themselves on the hour. The solution is usually given in the following manner: If a child doesn't see any other muddy partners, it must anouce itself, because it can be certain that it is the muddy child. Inductively, if a child can see k muddy partners at time k, it must announce itself at time $k+1$. If it wasn't muddy, the rest of the muddy children would each see $k-1$ partners and announce themselves at time k.

In the topological approach to this problem we represent the problem as an abstract simplicial complex, whose vertices are n dimensional vectors that represent the perspectives of each child. In vector representation, value 1 at coordinate i means that child i is muddy. Value 0 at coordinate i means that child i is clean, and $\perp$ represents ignorance about child i. Each possible configuration of this problem is represented by a simplex formed by n vertices that are consistent with each other. That is $v_1, \ldots, v_n$ form a simplex if $\forall \quad i, j \in \{1, \ldots, n\}, \forall \quad h \in \{1, \ldots, n\}, v_i(h) = v_j(h) \neq \perp$ or $v_i(h) = \perp$ or $v_j(h) = \perp$.

Notice that each vertex has exactly one coordinate with value $\perp$ (the coordinate corresponding to itself), since each child initially ignores its own status, but can see the rest of the children. Thus we can think of vertices as vectors of the following form $v = (x_1, \ldots, x_k = \perp, \ldots, x_n)$, assuming that $\perp = x_k$ for some $k \in \{1, \ldots, n\}$. This implies that each vertex belongs to exactly two simplices, the simplex where $\perp$ takes the value 1 and the simplex where $\perp$ takes value 0. Such simplices are the following, $\sigma_0(v) = \{z : z \text{ is consistent with } (x_1, \ldots, x_k = 1, \ldots, x_n)\}$ and $\sigma_1(v) = \{z : z \text{ is consistent with } (x_1, \ldots, x_k = 0, \ldots, x_n)\}$.

When the teacher announces that there is at least one muddy child the simplex $\sigma_0 = \{z : z \text{ is consistent with } (0, \ldots, 0)\}$ is removed from the abstract simplicial complex. That is, the possibility that every child is clean is discarded from our universe. Since every vertex belonged to two simplices, the vertices that belonged to σ_0 now only belong to one simplex. Since each of these vertices belong to only one simplex, they are certain of their status. This corresponds to the induction base on the first solution.

Now let's notice that, as time passes, our simplicial complex also evolves. For such purpose we define terminal simplices. We say that a simplex is terminal if it has at least one vertex that only belongs to this simplex. For each hour that has passed without announcements, the terminal simplices are removed. This is safe because any vertex that only belongs to one simplex is certain about its status. Thus, if no announcement is made, it's because these vertices perspective (which is now corresponding to its simplex) does not correspond to the outcome of the problem and can thus be removed.

Since at time 0 we have terminal simplices (caused by the teachers announcement), each vertex belongs to at most 2 simplices, each hour removes at least one terminal simplex, and because the whole abstract simplicial complex is connected at any time, eventually the simplex that represents the configuration matching the outcome of the problem becomes terminal, and an announcement will be made. If an announcement is made, then every other vertex on the simplex is also ready to announce itself, for there is only one simplex that makes every pair compatible.

Figure 2.1, taken from [33] illustrates this problem as an abstract simplicial complex for the case of 3 children and its evolution as time passes.

It is interesting to note that it suffices to guarantee that there is at least one terminal simplex at any given time. This is interesting because, we can change the initial statement by the teacher to any other that forbids any one or more simplices.

2.2 Cheryl's birthday puzzle

Now let's consider the following puzzle published by the New York Times [41]. Albert and Bernard are talking to Cheryl and decide to ask her birthday. She decides not

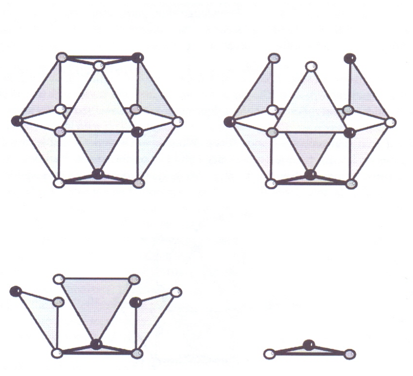

FIGURE 2.1: Muddy children problem evolution

to tell them directly but instead to give them some clues. First she lists 10 possible dates:

May		15	16			19
June				17	18	
July	14		16			
August	14	15		17		

After sharing this information with both of them, first she tells Albert the month of her birthday and next she tells Bernard the day of her birthday.

She asks Albert if he already knows her birthday, to which he responds: "I don't know, but I also know that Bernard doesn't know".

Upon hearing this Bernard says "I did not know her birthday, but now I know".

At last by hearing what Bernard said, Albert says "Now I also know her birthday".

Which date is Cheryl's birthday? To solve this question we will model the puzzle as an abstract simplicial complex. Each possible perspective from Albert and each possible perspective from Bernard form the vertices of the complex. Each possible birthday forms a simplex from the abstract simplicial complex. Since Cheryl tells Albert the month of her birthday, each possible perspective from Albert can be represented by each possible month. Likewise each possible perspective from Bernard can be represented by a day number between 14 and 19. Following these observations, we can notice that each simplex is formed by two vertices, the month and the day. Thus, our puzzle can be modeled by an undirected graph.

After Cheryl enumerates the list, Albert states that he doesn't know the date, and that Bernard also doesn't know the date. This means that Albert's perspective is ambiguous, and he also knows that Bernard's perspective is ambiguous. This statement

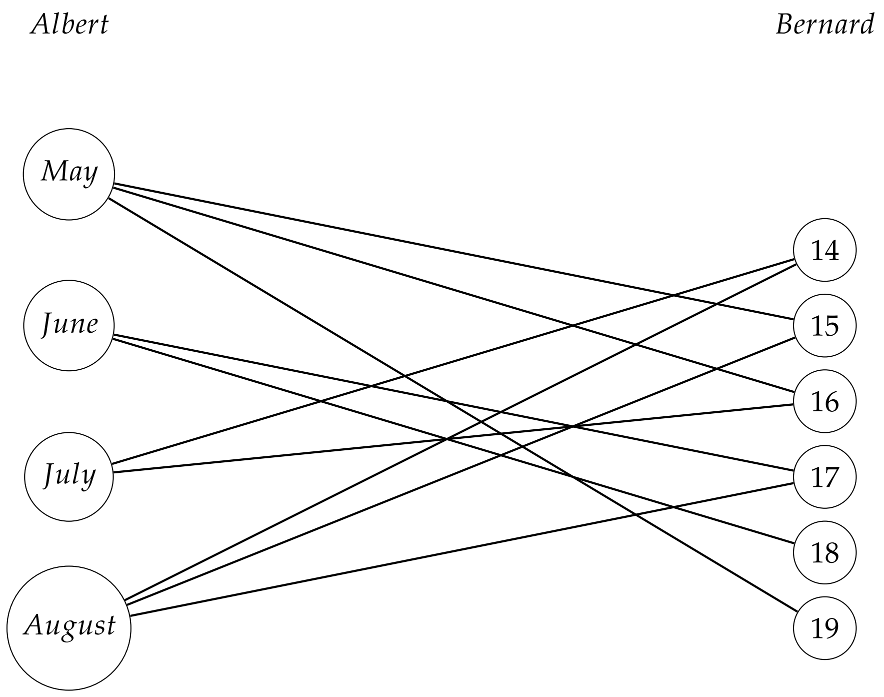

FIGURE 2.2: This initial undirected graph represents all possible birthdays after Cheryl enumerates a list

shows that all from Bernard's perspective, he has at least 2 possible months from which he can't decide with absoulte certainty. This means that vertices 18 and 19 are not part of the possibilities. Since Albert (who only knows the month) is aware of this, it should be because month June and May are not the month of the birthday, thus eliminating them from our graph.

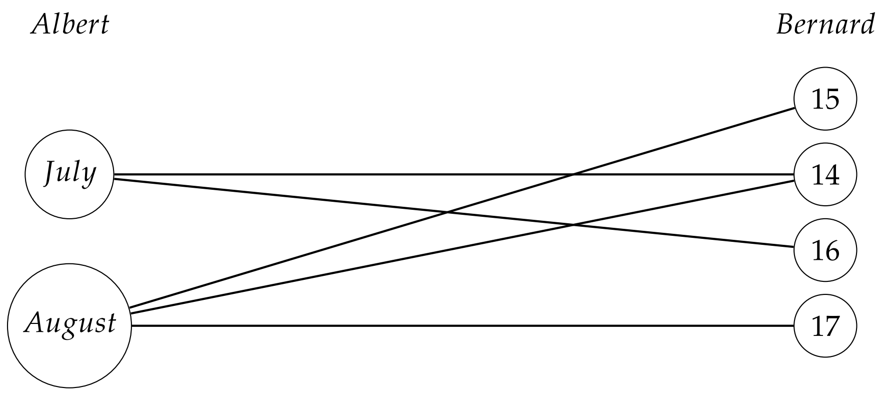

FIGURE 2.3: This graph remains after Albert's first statement

Next Bernard states that after hearing Albert's statement, he already knows the birthday. This removes vertex 14 from our graph, since it is the only ambiguous vertex (with degree at least 2) on Bernard's side.

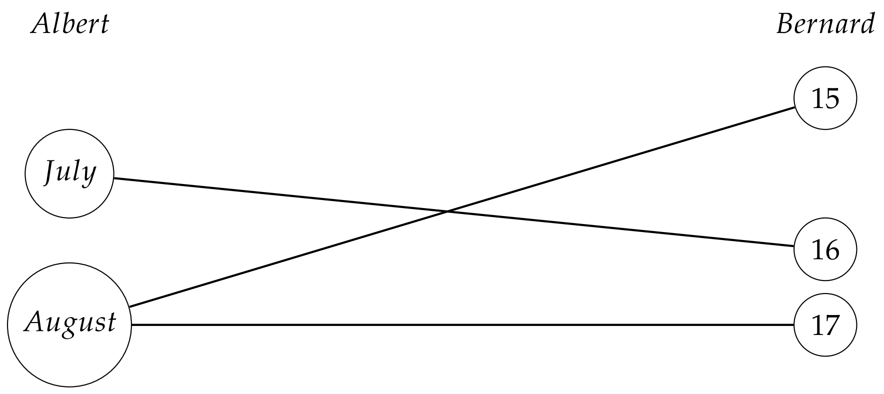

FIGURE 2.4: This graph remains after Bernard's statement

Finally since at last Albert knows the birthday after hearing what Bernard said, we remove all ambiguous vertices on Albert's side, leaving us only with July 16.

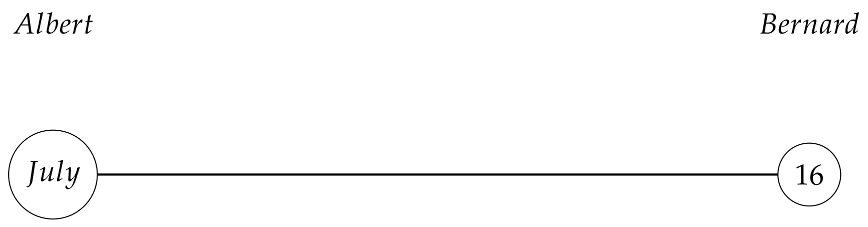

FIGURE 2.5: This graph remains after Albert's last statement

Chapter 3

The Asynchronous Shared Memory Model

In this chapter we give the basic formal definition for the model that we will use during the greater part of our work. We will also define tasks and task solvability.

This model is given by n processes that communicate through shared memory. Failure crashes are the only failures considered by our model. Since we are only concerned about computability, we will assume that the model is the same as the Iterated Immediate Snapshot Model. This model is equivalent in terms of computability power as the single write single read asynchronous shared memory model. The Iterated Immediate Snapshot Model, however provides us with a simpler way to reason about task solvability.

3.1 Model Specification

The asynchronous read/write model consists of $n+1$ processes denoted by $[n] = \{0, 1, \ldots, n\}$. These numbers are called ids.

A process is a deterministic state machine, that runs at an arbitrary speed independent of the speed of other processes (hence the asynchronous nature of the model).

Processes communicate using a shared memory array $mem[0 \ldots n]$ of $n+1$ atomic single writer-multi reader registers. Atomic operations $write(v)$ and $read(j), 0 \leq j \leq n$ are available to each process. Process i can use $write(v)$ to write value v to register i (single writer registers) and $read(j)$ reads register j ($mem[j]$).

The protocol D executed by the processes is given by their state machines, which are identical at every process except for the id i and some input value x_i. It is assumed that D is in canonical form, that is, it is deterministic, full-information and with a round structure. Full-information means that the entire local state of a process is available to other processes. That is, a process always writes its entire local state(view) to the shared memory. A round structure is defined as follows: A process executes a $Collect()$ operation if it reads all registers in an arbitrary order. We use the notation $WCollect(x)$ when a process executes a $write(x)$ operation followed by a $Collect()$ operation. A one round model consists of processes that execute once the $WCollect(x)$ operation. A multi round model consists of processes that execute N $WCollect(x)$ operations. By assuming that each process writes its whole local state (including past round states), and reads all the shared memory, we only need to concern ourselves with the execution of the rounds. The iterated immediate snapshot model simplifies the protocol specification, since each correct process executes exactly the same operations exactly the same amount of times. Since our main focus is computability power, this makes the iterated immediate snapshot model a useful choice for analyzing tasks.

3.2 States, actions, executions

We will add to the set of processes $[n]$, an environment register e that models shared memory and scheduling operations of the processes.
For each process i (including the environment register e) , we have a set $\mathcal{L}_i$ representing the local states of i. We use the notation L_i to indicate that i is at local state $L_i \in \mathcal{L}_i$, and $\mathcal{L}_i^0$ to denote the set of $i's$ initial states.. The set of global states is given by $\mathscr{G} = \mathcal{L}_e \times \mathcal{L}_0 \times \ldots \times \mathcal{L}_n$, with $\mathcal{G}^0$ describing the set of initial global states. That is, the vector of the local states defines a global state. This is an important property that will become relevant for the topological model of the protocol.

In the model described previously, given a protocol D and a global state G that includes the shared memory and the environment, the scheduling of the processes uniquely determines the next state of the system. A scheduling action is a set $Sched \subseteq \{0, \ldots, n\}$ of the processes that are scheduled to move next. A run of a deterministic protocol D can be represented by $G^0 \odot sa_1 \odot sa_2 \odot \ldots$, where G^0 is an initial global state and sa_i are scheduling actions. An execution is a subinterval of a run that starts and ends in a state. The sequence of scheduling actions $sa_1, sa_2, \ldots,$ is called a composed scheduling action. Given an execution R and a composed scheduling action sa, $R \odot sa$ denotes the execution given by extending R through the composed scheduling action sa (first execute R, then execute the composed scheduling actions, starting with the final state of R).

We will analyze N-round protocols. Let Sch be the set of all composed scheduling actions where every correct process executes its N rounds. Since the correct processes in the model that we are considering, each executes exactly N rounds, then we can consider all valid sequences of scheduling actions as permutations or subsequences of permutations. All such permutations are allowed when $t = n$ processes can crash. When $t < n$, it is possible to define a subset of the permutations that represent the model where task solvability is concerned. Let $S_N(D, M, \mathcal{G}^0)$ be the set of all executions of protocol D in model M that start in the initial states $\mathcal{G}^0$, and each process executes N rounds, according to a schedule in Sch.

3.3 Tasks

Processes have input values; in the initial states of the system, $\mathcal{G}^0$, all processes are in the same initial state, differing only in their ids and input values, the shared memory is empty. Each combination of input values assigned to each process is represented as a vector $(x_0, \ldots, x_n)$. Analogously, the output of a task can be defined by a set of vectors $(y_0, \ldots, y_n)$ which represent the combination of processes output values.

A task is defined by a set of input vectors, a set of output vectors, and a relation Δ that specifies for each input vector x, a set of valid output vectors denoted by $y = \Delta(x)$.

Definition 1. *A* ***task*** *is a triple $\langle I, O, \Delta \rangle$ where I and O are sets of vectors, and Δ is a map from I into 2^O [33], [13].*

In a given model with schedules *Sch*, a protocol D solves the task in N rounds if after executing any schedule in *Sch*, each process i with input value x_i, decides an output value y_i, such that $y = (y_i) \in \Delta(x)$ for $x = (x_i)$.

Consider for example the binary consensus task. Processes start with input value 0 or 1, and all correct processes have to decide the same value, which must be the input of one of the processes. This task is unsolvable, even if only one process may crash [1].

More generally, in the k-set agreement task [13], processes start with inputs from a set of at least $k+1$ values, and have to decide on at most k different inputs. This task is solvable if and only if $t < k$, where t is the number of processes allowed to crash.

Another task that is similar to consensus is the approximate agreement task. Correct processes must agree on values close to each other. The approximate agreement task is solvable in the asynchronous model even when $t = n$ [5]. The rounds of the protocol needed depend on how close the values should be.

In the next section we will provide a topological framework for analyzing the asynchronous model.

Chapter 4

Combinatorial Topology

In this chapter we will give the preliminary combinatorial topology concepts that we need for our results. It is intended that this work be self-contained, although a basic knowledge of topology is required for further results.

First we define the basic objects known as simplicial complexes. Simplicial complexes will be used to define spaces that represent universes of valid inputs, valid outputs or minimal final configurations. It is interesting to note that we provide these universes with a topological structure instead of only considering the isolated inputs or outputs.

This topological structure gives rise to interesting properties that can be translated to distributed computing properties.

4.1 Basic Concepts

First we define the basic object in combinatorial topology, the simplicial complex. Simplicial complexes are interesting because they are combinatorial objects that capture interesting topological properties. In a way, they are a bridge between combinatorics and topology.

A simplicial complex $\mathcal{S}$ is a pair $\langle V, S \rangle$, such that V is a set, called the vertices of $\mathcal{S}$, and $S \subseteq 2^V$ is a collection of finite non-empty subsets of V such that if $\sigma \in S$ and $\sigma' \subseteq \sigma$, then $\sigma' \in S$; and for each $v \in V$, $\{u\} \in S$.

We say that a simplicial complex $\mathcal{L} = \langle V', S' \rangle$ is a subcomplex of $\mathcal{K} = \langle V, S \rangle$ if $V' \subseteq V$ and $S' \subseteq S$. We denote it by $\mathcal{L} \subseteq \mathcal{K}$.

A simplicial complex $\mathcal{S} = \langle V, S \rangle$ is called a simplex if $S = 2^V$, when V is finite. In this thesis we will restrict only to finite vertex sets. Abusing notation, we will also refer to elements of S as simplices (since elements of S correspond to subcomplexes that are simplices). We can think of simplices as the building blocks of a simplicial complex. We will also refer to simplices as faces of $\mathcal{S}$.

Also we will use the same name of the simplicial complex (S) to refer to the simplicial set, in order to avoid cumbersome notation. We will return to the tuple notation whenever there is a risk of ambiguity.

Let $\mathcal{S} = \langle V, S \rangle$ be a simplicial complex. $\sigma \in S$ is a facet of $\mathcal{S}$ if it is maximal in S, that is, σ is not a face of a larger simplex.

A simplex $\sigma \in S$, has dimension $n-1$ if $|\sigma| = n$. We say that the whole complex has dimension n if the maximum dimension of its simplices is n.

A simplicial complex C is pure if all its facets are of the same dimension n.

Let $\mathcal{K} = \langle V_1, S_1 \rangle$ and $\mathcal{L} = \langle V_2, S_2 \rangle$ be two simplicial complexes. We define the join of $\mathcal{K}$ and $\mathcal{L}$, denoted by $\mathcal{K} * \mathcal{L} = \langle V_1 \sqcup V_2, S' \rangle$, where $S' = S_1 \sqcup S_2 \sqcup \{\alpha \sqcup \beta \mid \alpha \in S_1,\ \beta \in S_2\}$. The join is a higher-dimensional case of the graph sum. Also the join of a simplicial complex with a vertex is the combinatorial analog to the coning operation.

Simplicial complexes can also be thought in a more geometric sense. Given a simplex σ of dimension n, we can consider S, a set of $n+1$ affine-indepent vertices in $\mathbb{R}^{n+1}$. The affine space generated by S is called the geometric realization of S, and is denoted by $|S|$. However, since defining a simplicial complex directly through cartesian coordinates may be troublesome, we will define it as a functional space, where each point is an affine coordinate function.

Now we define maps that are structure preserving. Let $\mathcal{K} = \langle V, S \rangle$ and $\mathcal{L} = \langle V', S' \rangle$ be two simplicial complexes. A map $\mu : V \to V'$ is a simplicial map between $\mathcal{K}$ and $\mathcal{L}$ if for all $\sigma \in S$, $\mu(\sigma) \in S'$. Given $\{v_0, \ldots, v_n\} \in S$, then $\{\mu(v_0), \ldots, \mu(v_n)\} \in S'$.

We will also be interested in maps that are not vertex-vertex maps. In particular, we will also consider maps that send simplices into subcomplexes. We also want those maps to preserve somehow the topological structure. If $\mathcal{K} = \langle V, S \rangle$ and $\mathcal{L} = \langle V', S' \rangle$ are simplicial complexes, then $\varphi : S \to 2^{S'}$ is a carrier map if it is monotonic, that is, for all $\sigma, \tau \in S$, $\sigma \subseteq \tau$ implies that $\varphi(\sigma) \subseteq \varphi(\tau)$.

Let S be a simplicial complex with vertices V_S, we define a geometric simplicial complex, denoted by $|S| = \{\alpha : V_S \to [0,1] \subseteq \mathbb{R} \mid \alpha^{-1}(0,1] \in S, \sum\limits_{v \in V_S} \alpha(v) = 1\} \subseteq \mathbb{R}^{V_S}$.

If V_S is finite, this gives way to a canonical embedding into a real vector space, which is finite dimensional if S is finite, that does not depend on how the vertices are labeled. Simply notice that $|S| \subseteq \mathbb{R}^{V_S} \cong \mathbb{R}^k$ for some k if V_S is finite.

However, if $|V_S|$ is too large, there exists an embedding that depends only on the dimension of S, but is not canonical since it depends on the vertex labeling. Lets assume that $V_S = \{v_1, v_2, \ldots, v_m\}$ and $dim(S) = n$. Let $p_i = (i, i^2, i^3, \ldots, i^{2n+1}) \in \mathbb{R}^{2n+1}$ for $1 \leq i \leq m$. It follows that $f : |S| \hookrightarrow \mathbb{R}^{2n+1}$, $f(\alpha) = \sum\limits_{i=1}^{m} \alpha(v_i) p_i$ is an affine embedding into R^{2n+1}.

This construction allows us to connect the combinatorial point of view with a topological point of view by using affine combinations.

Now that we have a geometric interpretation of simplicial complexes, we can now define subdivisions of simplicial complexes.

Let $\mathcal{A}$ and $\mathcal{B}$ be simplicial complexes. We say that $\mathcal{B}$ subdivides (is a subdivision of) $\mathcal{A}$ if there exists an homeomorphism $h : |\mathcal{A}| \to |\mathcal{B}|$ and a carrier map $\Phi : \mathcal{A} \to 2^{\mathcal{B}}$ such that, for every simplex $\sigma \in \mathcal{A}$, the restriction $h|_{|\sigma|}$ is a homeomorphism between $|\sigma|$ and $|\Phi(\sigma)|$ [33].

Another equivalent definition of subdivision is given as follows : $\mathcal{B}$ is a subdivision of $\mathcal{A}$ if the following conditions are met:

1. $V_B \overset{\iota}{\hookrightarrow} |A|$.
2. Given $\{\alpha_0, \dots, \alpha_q\} \in \mathcal{B}$, there exists $s \in \mathcal{A}$ such that $\{\alpha_0, \dots, \alpha_q\} \in |s|$.
3. The induced map $\hat{\iota} : |\mathcal{B}| \to |\mathcal{A}|$ is a homeomorphism, where $\hat{\iota}$ is defined as follows: let $\beta = \sum\limits_{w \in V_{\mathcal{B}}} \beta(w)w$, then $\hat{\iota}(\beta) = \sum\limits_{w \in V_{\mathcal{B}}} \beta(w)\iota(w)$.

As we can see, the definition allows for many different kinds of subdivision, however we are particularly interested in the standard barycentric subdivision, and its chromatic generalization, the standard chromatic subdivision.

Let $\mathcal{A} = \langle V, S \rangle$ be a simplicial complex. The barycentric subdivision of $\mathcal{A}$ denoted by $\text{Bary}\mathcal{A} = \langle \text{Bary}(V), \text{Bary}(S) \rangle$ is defined as follows:

- The set of vertices of the barycentric subdivision of $\mathcal{A}$, corresponds to the set of simplices of $\mathcal{A}$. $\text{Bary}(V) = S$.
- A collection of simplices of $\mathcal{A}$, $\{\sigma_0, \dots, \sigma_k\}$ belongs to $\text{Bary}(S)$ if $\{\sigma_0 \subsetneq \dots \subsetneq \sigma_k\}$. Notice that the vertices σ_i of $\text{Bary}(V)$ correspond geometrically to the barycenter of σ_i.

In a more explicit way, we can define the inclusion map $Bary(V) \overset{\iota}{\hookrightarrow} |S|$ as follows: let $\sigma \in S = Bary(V) = S$ be a q-simplex, then $\iota(\sigma)(v) = \begin{cases} \frac{1}{q+1} & \text{if } v \in \sigma \\ 0 & \text{if } v \notin \sigma \end{cases}$.

This definition corresponds to $\iota(\sigma)$ being the barycenter of $|\sigma|$.

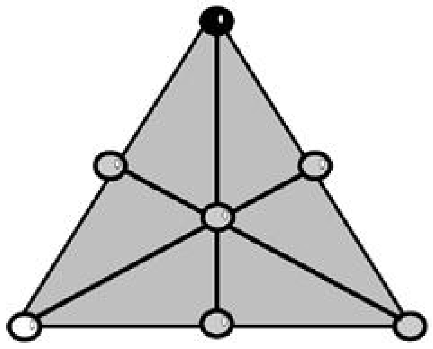

FIGURE 4.1: Barycentric subdivision of a triangle [33]

Since we are interested in distinguishing between processes, we also color the vertices of a simplicial complex. If $\mathcal{K} = \langle V, S \rangle$ is a simplicial complex, a coloring of $\mathcal{K}$ is a map from the vertices of the complex to a finite set of colors C, $\chi : V \to C$ such that for every $\sigma \in S$, $\chi_{|\sigma}$ is injective (it does not repeat colors in any simplex).

A chromatic simplicial complex is the object given by a simplicial complex and a coloring for the simplicial complex. We denote a chromatic simplicial complex as a tuple $\langle V, S, \chi, C \rangle$, where V and S are as defined for simplicial complexes, C is a finite color set, and χ is a coloring for $\langle V, S \rangle$. Note that every simplicial complex admits a coloring. Take for example the trivial coloring $\chi : V \to V$, where $\chi = I_V$ is the identity map.

Now, we can define a map that also preserves the color structure. A chromatic simplicial map between chromatic simplices $\mathcal{K} = \langle V, S, \chi, C \rangle$ and $\mathcal{L} = \langle V', S', \chi', C \rangle$ is a map $\mu : V \to V'$ such that it is a simplicial map and $\chi(v) = \chi(\mu(v))$.

We can also define a chromatic join of two chromatic simplicial complexes $\mathcal{K}$ and $\mathcal{L}$ if we either restrict to disjoint color sets or by making them disjoint from the definition of the join. To simplify notation, we will assume that $\mathcal{K}$ and $\mathcal{L}$ have different chromatic sets.

Let $\mathcal{A} = \langle V, S, \chi, C \rangle$ and $\mathcal{B} = \langle V', S', \chi', C \rangle$ be chromatic simplicial complexes. We say that $\mathcal{B}$ subdivides chromatically (is a chromatic subdivision of) $\mathcal{A}$ if $\langle V', S' \rangle$ is a subdivision of $\langle V, S \rangle$ and the subdivision carrier map $\Phi : \mathcal{A} \to 2^{\mathcal{B}}$ is chromatic, meaning that it preserves color, that is, a vertex is mapped of a color c is mapped to a set of vertices necessarily of color c. [33].

We also define a chromatic generalization of the barycentric subdivision called the standard chromatic subdivision. Let $\mathcal{A} = \langle V, S, \chi, C \rangle$ be a chromatic simplicial complex. We define the standard chromatic subdivision of $\mathcal{A}$, denoted by $\text{Ch}\mathcal{A} = \langle \text{Ch}V, \text{Ch}S, \chi', C \rangle$ as follows: The vertices of the standard chromatic subdivision are pairs (c, σ) such that $c \in C$, $\sigma \in S$, and $c \in \chi(\sigma)$. A collection of pairs $\{(c_0, \sigma_0), \ldots, (c_k, \sigma_k)\}$ is a k-simplex in the standard chromatic subdivision if$\{\sigma_0 \subseteq \ldots \subseteq \sigma_k\}$, and if $c_i \in \chi(\sigma_j)$ then $\sigma_i \subseteq \sigma_j$. In the standard chromatic subdivision of a simplex σ, it is particularly useful to define the center simplex of the subdivision. The center simplex consists of the vertices (c_i, σ) for each color $c_i \in \chi(\sigma)$. Figure 4.2 shows an example of a standard chromatic subdivision.

We use chromatic simplicial complexes to model the possible views for each process. A color corresponds to a process and a vertex to a possible view for such process. In the previous section we noted that a global configuration is determined by the local configurations of the processes. It is natural then to see that a global configuration under this model corresponds to a simplex: Possible global states form a simplicial complex in which vertices are local views of a process, simplices are global configurations, and an implicit undistinguishability relation is given by vertices that belong to several simplices.

We use such a model to represent the inputs, the outputs, and the different stages of execution of a protocol. In our asynchronous model of computation, a communication

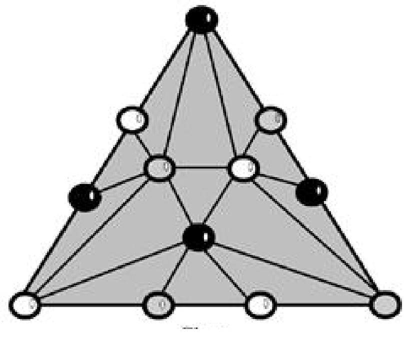

FIGURE 4.2: Chromatic subdivision of a triangle [33]

round can be described by applying an iterated operation to the input complex.

More formally, we can model a task $\langle I, O, \Delta \rangle$ using the chromatic simplicial complexes in the following way. As discussed earlier, I and O are sets of vectors that represent configurations for each of the processes. We now use them to build an input complex $\mathcal{I} = \langle V_I, S_I \rangle$ and an output complex $\mathcal{O} = \langle V_O, S_O \rangle$ for the task.

For the vertices of the input complex we consider the input values of the processes: $V_I = \{(p,x) |\ p$ is a process and x is a valid input for $p\}$. A simplex $s = \{(p_1, x_1), \ldots, (p_k, x_k)\} \in S_I$ if there exists a vector $x \in I$ such that s induces a subvector of s, i.e. that s forms a valid input configuration. For $(p, x_p) \in V_I$ then we define the coloring of the input complex as $\chi(p, x_p) = p$.

We define the output complex in the same way as the input complex. The only difference is that I and O are different vector sets.

For example, let's consider 2-process binary consensus. In that case $I = \{(0,0), (0,1), (1,0), (1,1)\}$ and $O = \{(0,0), (1,1)\}$.

For the input complex we have the following vertices and simplices:
$V_I = \{(p_1,0), (p_1,1), (p_2,0), (p_2,1)\}$.
$S_I = \{\{(p_1,0)\}, \{(p_1,1)\}, \{(p_2,0)\}, \{(p_2,1)\}, \{(p_1,0), (p_2,0)\}, \{(p_1,0), (p_2,1)\}, \{(p_1,1), (p_2,0)\}, \{(p_1,1), (p_2,1)\}\}$.

The output complex has the following vertices and simplices:
$V_O = \{(p_1,0), (p_1,1), (p_2,0), (p_2,1)\}$
$S_O = \{\{(p_1,0)\}, \{(p_1,1)\}, \{(p_2,0)\}, \{(p_2,1)\}, \{(p_1,0), (p_2,0)\}, \{(p_1,1), (p_2,1)\}\}$.

Figure 4.3 illustrates the topology of the input and output complexes.

It is worth noting that the input complex for consensus is connected and the output complex for consensus is disconnected.

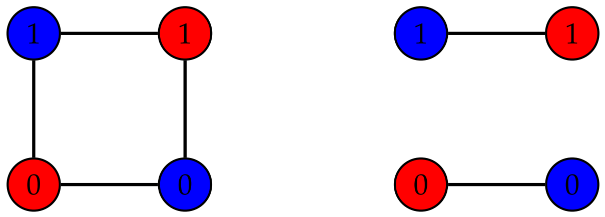

FIGURE 4.3: Input(left) and output(right) complexes for 2 process binary consensus

Given the input and output complexes for the task $T = \langle I, O, \Delta \rangle$, we now translate Δ into a map from the input complex into the output complex. Note that Δ is not a simplicial map, since a single input simplex may have more than one acceptable output. Taking consensus for example, we find $\Delta(\{(p_1,0),(p_2,1)\}) = \{\{(p_1,0),(p_2,0)\},\{(p_1,1),(p_2,1)\}\}$. However Δ induces a carrier map from the input complex to the output complex. More specifically, for V_I, let v be a vertex of $\mathcal{I}$, we define $S_v = \{\sigma \in S_I \mid v \in \sigma$ and σ is a facet of the input complex $\}$.

We define $\tilde{\Delta}(\sigma)$ as the subcomplex of $\mathcal{O}$ given by $\Delta(\bigcap\limits_{v \in \sigma} S_v)$. By definition it is clear that $\tilde{\Delta}$ is a carrier map, and that it extends Δ.

4.2 Subdivisions and Communication

Lets recall from a previous section that since the asynchronous model is a full-information protocol, we only need to be concerned about the scheduling of the process rounds.

We can use our topological framework to analyze the effects of communication rounds on the input complex. We define a protocol complex $\mathcal{P} = \langle V_P, S_P \rangle$ in the following way.

$V_P = \{(p,s) \mid p$ is a process and s is an n-round final view of $p\}$. This means that for each process and each of its possible final views there is a vertex in V_P. Note that a final view is a local state of a process p such that the protocol is able to decide an output value for p.

$S_P = \{\sigma \mid$ for all $(p_1,s_1),(p_2,s_2) \in \sigma$, s_1 and s_2 are compatible n-round final process views $\}$. Note that s_1 and s_2 are compatible if there exists a run where s_2 is a final view for p_1 and s_2 is a final view for p_2

Herlihy and Shavit, showed that an n-round protocol complex corresponds to the n^{th} chromatic subdivision of the input complex [3]. Even more, they characterized asynchronous solvability in terms of the input and output complexes using the correspondence between the n-round protocol complex and the standard n^{th} chromatic subdivision of the input complex.

Let's consider a scenario with two processes p_1 and p_2 that have initial input values 1 and 0 respectively.

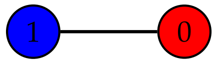

FIGURE 4.4: p_1 and p_2 (blue and red) have initial values 1 and 0 respectively

The following possibilities are valid observations for p_1:
p_1 wrote its view and read before p_2 was able to write its view. We denote this by $(p_1, \perp)$.
p_1 wrote its view and read after p_2 wrote its view. We denote this by $(p_1, 0)$.

The valid observations for p_2 are:
p_2 wrote its view and read before p_1 was able to write its view. We denote this by $(p_2, \perp)$.
p_2 wrote its view and read after p_1 wrote its view. We denote this by $(p_2, 1)$.

The valid protocol runs are:

1. $\{(p_1, \perp), (p_2, 1)\}$ which corresponds to p_1 reading before p_2 wrote its view. It results from the following process schedule $(p_1.\mathit{write}, p_1.\mathit{collect}, p_2.\mathit{write}, p_2.\mathit{collect})$.
2. $\{(p_2, 1), (p_1, 0)\}$ which corresponds to both p_1 and p_2 being able to read each others view. It results from the process schedules
$\{(p_1.\mathit{write}, p_2.\mathit{write}, p_1.\mathit{collect}, p_2.\mathit{collect}), (p_2.\mathit{write}, p_1.\mathit{write}, p_1.\mathit{collect}, p_2.\mathit{collect}),$
$(p_1.\mathit{write}, p_2.\mathit{write}, p_2.\mathit{collect}, p_1.\mathit{collect}), (p_2.\mathit{write}, p_1.\mathit{write}, p_2.\mathit{collect}, p_1.\mathit{collect})\}$.
3. $\{(p_1, 0), (p_2, \perp)\}$ which corresponds to p_2 reading before p_1 is able to write its view.
It results from the following process schedule $(p_2.\mathit{write}, p_2.\mathit{collect}, p_1.\mathit{write}, p_1.\mathit{collect})$.

Figure 4.5 illustrates the protocol complex for this case.

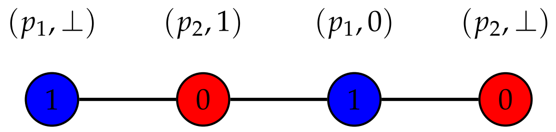

FIGURE 4.5: 1-round protocol complex

Note that we can get an $s+k$ round protocol by composing an s-round protocol and a k-round protocol. By using this observation, we can consider the final view of a process in an s round protocol as the input value for the k round protocol that will come after. This implies that an n-round protocol complex is obtained by the n-iterated chromatic subdivision of the input complex, since a 1 round protocol induces a 1-chromatic subdivision, and an n-round protocol can be obtained by iterating a 1-round protocol.

Chapter 5

The ACT and Simplicial Approximation

We have already given the preliminary concepts and definitions regarding combinatorial topology and the distributed computing model.

In this chapter we introduce the Asynchronous Computability Theorem (ACT) and the Simplicial Approximation Theorem. These results can be used to produce a fully topological characterization of colorless tasks. This characterization serves as the basis for our first result which extends the Simplicial Approximation Theorem into a chromatic version.

In order to approach the ACT, we show the relationship between communication rounds and the standard chromatic subdivision of the input complex.

5.1 The Asynchronous Computability Theorem

Now that we have defined simplicial complexes for both the inputs and outputs and defined a carrier map between them that corresponds to Δ we can move to the Asynchronous Computability Theorem. This theorem was stated by Herlihy and Shavit [3], and characterizes asynchronous task solvability in terms of combinatorial topology, using the tools and framework introduced in the previous section.

Theorem 1 (Asynchronous Computability Theorem [3]). *Let $T = \langle I, O, \Delta \rangle$ be a task. T has a wait-free, layered immediate snapshot protocol if and only if the input complex $\mathcal{I}$ has a chromatic subdivision $Div\mathcal{I}$ and a chromatic simplicial map $\mu : Div\mathcal{I} \to \mathcal{O}$ into the output complex such that μ preserves Δ.*

The goal of this work is to establish sufficient topological conditions on the simplicial complexes of the theorem such that a chromatic simplicial map exists between a chromatic subdivision of the input complex $\mathcal{I}$ and the output complex $\mathcal{O}$.

A very important tool for connecting simplicial complexes to topological properties is simplicial approximation. A simplicial approximation is a simplicial map that is homotopic (can be deformed continuously) to a given continuous function. It allows us to represent a continuous function in combinatorial terms, in other words, it extracts set topological properties and carries it to combinatorics.

A very useful result for finding simplicial approximations is the simplicial approximation theorem, which roughly states that every continuous function between simplicial complexes has a simplicial approximation. The formal definition of simplicial approximation and the statement for the simplicial approximation theorem will be given in detail in the following section. An original proof for an appropriate chromatic approximation theorem will also be included.

It is worth noting that this theorem does not yield a *chromatic* simplicial approximation. In this chapter we will show that this is possible under certain conditions for the output complex $\mathcal{O}$.

5.2 Simplicial approximation

(Following notation from http://suess.sdf-eu.org/website/lang/en/algtop/notes5.pdf)

Suppose that $f : |\mathcal{K}| \to |\mathcal{L}|$ is a continuous function between the geometric realizations of two simplicial complexes $\mathcal{K}$ and $\mathcal{L}$. Notice that, for each point $y \in |\mathcal{L}|$, y is an interior point of a unique simplex $\sigma \in \mathcal{L}$. This simplex is called the *carrier* of y and is denoted $\text{carr}_{\mathcal{L}}(y)$ (with respect to $\mathcal{L}$)

A simplicial map $\phi : K \to L$ is a *simplicial approximation* to f (with respect to $\mathcal{K}$ and $\mathcal{L}$) if, for each point $x \in |\mathcal{K}|$, the image of its geometric realization $|\phi|(x)$ lies in $carr_{\mathcal{L}}(f(x))$.

For a given simplicial complex K we define the *star* of a vertex v by

$$star(v) := \bigcup_{v\in\sigma} \sigma^o.$$

Where σ^o is the interior of the geometric realization of σ.

An alternate, but equivalent and useful definition for open star of a vertex is the following:

Let $v \in V_K$, we define $st(v) = \{\alpha \in |K| \mid \alpha(v) \neq 0\}$. From this definition, it clearly follows that $st(v)$ is an open subset of $|K|$.

The following definition is also equivalent:

Let $f : |K| \to |L|$ continuous, a simplicial map $\phi : K \to L$ is a simplicial approximation to f if $f(\alpha) \in |l| \Rightarrow |\phi|(\alpha) \in |l|$, for $\alpha \in |K|$ and $l \in L$. In particular if $v \in V_k$ such that $f(v) \in V_l$ then $|\phi|(v) = f(v)$. Each $v \in V_k$ defines an element $v \in |K|$ given by $v(v') = \begin{cases} 1 & \text{if} v' = v \\ 0 & \text{if} v' \neq v \end{cases}$

Notice that when K is finite, we give $|K|$ the metric $d(\alpha,\beta) = \sqrt{\sum_{v\in V_K} [\alpha(v) - \beta(v)]^2}$. If $V_K = \{v_1, \dots, v_m\}$ then $|K| \subseteq \mathbb{R}^{V_K} \cong \mathbb{R}^m$, so that $|K|$ has the canonical euclidean metric.

Lemma 1. *If $\phi : K \to L$ is a simplicial approximation to $f : |K| \to |L|$, then $|\phi| \cong f$.*

Let X be a triangulable space, and let $\mathscr{U}$ be an open covering of X. A triangulation $f : |K| \to X$ is finer than $\mathscr{U}$ if for each $v \in V_K$, there exists $U \in \mathscr{U}$ such that $f(st(v)) \subseteq U$.

Another important technical concept for simplicial approximation is the mesh of the geometric realization of a simplicial complex. Lets recall that the geometric realization of an n-dimensional simplicial complex is an Euclidean space (embedded in $\mathbb{R}^{2n+1}$). Therefore, it can be given a metric. Since we are considering finite simplicial complexes, we can define the maximum diameter of every simplex, as a measure of the coarseness of the complex. This measure will help us to decide whether a subdivision is fine enough for simplicial approximation.

More formally we define the mesh of a simplicial complex $\mathcal{K}$ as

$$\text{mesh}(\mathcal{K}) = \max\{\text{diam}(\sigma) | \sigma \in \mathcal{K}\}$$

Lemma 2. *For any vertex $v \in |\sigma|$ in the geometric realization of a simplex σ of dimension n, with barycenter $\hat{\sigma}$ and diameter $diam(\sigma) = d$, $d(v, \hat{\sigma}) \leq \frac{n}{n+1} \cdot d$. Therefore the barycentric subdivision is a mesh shrinking operation.*

Proof. Consider $\sigma = \{v_0, \ldots, v_n\}$. Since $v \in |\sigma|$, it is an affine combination of $v_0, \ldots v_n$, that is, $v = \sum_{i=0}^{n} \lambda_i \cdot v_i$. Note that the barycenter $\hat{\sigma}$ is also an affine combination $\hat{\sigma} = \sum_{i=0}^{n} \frac{1}{n+1} \cdot v_i$.

First consider the distance between the barycenter and the rest of the vertices:

$$|\hat{\sigma} - v_j| = \left| \left(\sum_{j \neq i = 0}^{n} \frac{1}{n+1} \cdot v_i \right) - \frac{n}{n+1} \cdot v_j \right| = \frac{1}{n+1} \left| \sum_{i=0}^{n} (v_i - v_j) \right| \leq \frac{n}{n+1} \cdot d$$

.

Note that since v is an affine combination, then $\hat{\sigma} = \sum_{i=0}^{n} \lambda_i \cdot \hat{\sigma}$. Therefore

$$|v - \hat{\sigma}| = |\sum_{i=0}^{n} \lambda_i \cdot (v_i - \hat{\sigma})| \leq \sum_{i=0}^{n} \lambda_i |(v_i - \hat{\sigma})| \leq \sum_{i=0}^{n} \lambda_i \cdot (\frac{n}{n+1} \cdot d) = \frac{n}{n+1} \cdot d$$

.

□

Corollary 1. *If $\mathcal{K}$ is an n-dimensional simplicial complex of mesh d then, $Bary^k(\mathcal{K})$ has a mesh upper bound of $(\frac{n}{n+1})^k \cdot d$.*

Corollary 2. *For every n-dimensional simplicial complex $\mathcal{K}$ and every $\epsilon > 0$, there exists $k \in \mathbb{N}$ such that $mesh(Bary^k \mathcal{K}) < \epsilon$.*

We will now show that if we choose carefully the center simplex for standard chromatic subdivision, the mesh upper bound still holds: Lets assume that for a given simplex σ of diameter d and dimension n, and a k-dimensional face κ, we chose the subdivision simplex of κ to have diameter at most $\frac{k}{k+1} \cdot d$. Since the subdivision simplex of κ has diameter at most $\frac{k}{k+1} \cdot d$, adjacent new vertices are at a distance at most $\frac{k}{k+1} \cdot d$. Since every other edge lies inside a face of the barycentric subdivision, its length is bounded by $\frac{n}{n+1} \cdot d$. Therefore, all edges of the standard chromatic subdivision for σ are bounded by $\frac{n}{n+1} \cdot d$.

This observation shows that the previous results hold for the standard chromatic subdivision as well.

Corollary 3. *If $\mathcal{K}$ is an n-dimensional chromatic simplicial complex with mesh d, then $Ch^k(\mathcal{K})$ has a mesh upper bound of $(\frac{n}{n+1})^k \cdot d$.*

Corollary 4. *For every n-dimensional chromatic simplicial complex $\mathcal{K}$ and every $\epsilon > 0$, there exists $k \in \mathbb{N}$ such that $mesh(Ch^k\mathcal{K}) < \epsilon$.*

Lemma 3. *$\sigma = \{v_0, \ldots, v_n\}$ is a simplex in $\mathcal{K}$ iff $\bigcap_{i=0}^{n} star(v_i) \neq \varnothing$.*

Proof. If σ is a simplex in $\mathcal{K}$ then $\sigma^o \neq \varnothing$. Let $x \in \sigma^o$. For each v_i, $i \in \{0, \ldots, n\}$, $v_i \in \sigma$. Then $\sigma^o \subseteq star(v_i) = \bigcup_{v_i \in \mu} \mu^o$. Therefore $\varnothing \neq \sigma^o \subseteq \bigcap_{i=0}^{n} star(v_i)$.

Now lets assume that $\bigcap_{i=0}^{n} star(v_i) \neq \varnothing$. Let $x \in \bigcap_{i=0}^{n} star(v_i)$ and $\mu = \text{carr}_{\mathcal{K}}(x)$. Since $x \in star(v_i)$, $\text{carr}_{\mathcal{K}}(x) = \mu \subseteq star(v_i)$. It follows from the definition of $star(v_i)$ and from $\mu \subseteq star(v_i)$ that $v_i \in \mu$ for $i \in \{0, \ldots, n\}$.

Since μ is a simplex of $\mathcal{K}$ and $\sigma \subseteq \mu$ therefore $\sigma = \{v_0, \ldots, v_n\}$ is also a simplex of $\mathcal{K}$. □

Lemma 4. *Let $v \in \bigcap_{i=0}^{n} star(v_i)$, then $v_i \in carr_{\mathcal{K}}(v)$ for each $i \in \{0, \ldots, n\}$.*

Proof. Since $v \in star(v_i)$, then $\text{carr}_{\mathcal{K}}(v) = \mu \subseteq star(v_i)$. Therefore $v_i \in \mu$ for each i. □

Lemma 5. *Let $f : |\mathcal{K}| \to |\mathcal{L}|$ be a continuous function between the underlying spaces of the simplicial complexes $\mathcal{K}$ and $\mathcal{L}$. Assume there is a vertex map ϕ, such that for each vertex v of $\mathcal{K}$,*

$$f(star^o(v)) \subseteq star^o(\phi(v)).$$

Then ϕ is a simplicial approximation to f

Proof. To show that ϕ is a simplicial approximation of f, we need to show that it is a simplicial map and that for each point $x \in |\mathcal{K}|$, $|\phi(x)| \in \text{carr}_{\mathcal{L}}(f(x))$.

Let $x \in |\mathcal{K}|$. Consider $\text{carr}_{\mathcal{K}}(x) = \{v_0, \ldots, v_k\}$, then $x \in carr_{\mathcal{L}}(x)$ and $x \in \bigcap_{i=0}^{k} star(v_i)$.

By hypothesis we have that $f(star(v_i)) \subseteq star(\phi(v_i))$. This implies that $f(\bigcap_{i=0}^{k} star(v_i)) \subseteq$

$\bigcap_{i=0}^{k} f(star(v_i)) \subseteq \bigcap_{i=0}^{k} star(\phi(v_i))$, and hence $f(x) \in \bigcap_{i=0}^{k} star(\phi(v_i))$. Therefore $\bigcap_{i=0}^{k} star(\phi(v_i)) \neq \emptyset$. By Lemma 2, $\{\phi(v_0), \dots, \phi(v_k)\}$ is a simplex in $\mathcal{L}$. This makes ϕ a simplicial map. Also, since $f(x) \in \bigcap_{i=0}^{k} star(\phi(v_i))$, Lemma 3 reveals that $|\phi(v_i)| \in \mathrm{carr}_{\mathcal{L}}(f(x))$. This shows that $\phi(\mathrm{carr}_{\mathcal{K}}(x)) \subseteq \mathrm{carr}_{\mathcal{L}}(f(x))$. Therefore $\phi(x) \in \mathrm{carr}_{\mathcal{L}}(f(x))$, showing that ϕ is a simplicial approximation to f. □

Lemma 6. *Let $\mathcal{K} = \langle V(\mathcal{K}), S(\mathcal{K}) \rangle$ be a simplicial complex, then $\mathscr{C} = \{star(v) | v \in V(\mathcal{K})\}$ is an open covering of $|\mathcal{K}|$.*

Proof. Let $x \in |\mathcal{K}|$, consider $\mathrm{carr}_{\mathcal{K}}(x) = \sigma = \{v_0, \dots, v_n\}$, then $\sigma \in$ star(v_i) for each i. By definition, each $star(v_i)$ is an open set. □

Lemma 7. *Let $\mathcal{K}$ be a simplicial complex, and $\mathscr{C}$ an open covering for $|\mathcal{K}|$. Then there exists $n \in \mathbb{N}$ such that for every vertex $v \in V(Bary^n \mathcal{K})$, $star(v) \subseteq S \in \mathscr{C}$.*

Proof. Since $|\mathcal{K}|$ is compact [28], for every open covering $\mathscr{C}$, there exists $\delta > 0$ (called the Lebesgue number for $\mathscr{C}$) such that for every A with $\mathrm{diam}(A) < \delta$ then $A \subseteq S \in \mathscr{C}$. Consider some n such that $\mathrm{Bary}^n \mathcal{K}$ has mesh less than $\delta/2$. For every $x, y \in star(v)$, we find $d(x,y) \leq d(x,v) + d(v,y) < \delta$, $star(v)$ has diameter less than δ. Therefore $star(v) \subseteq S \in \mathscr{C}$. □

Theorem 2 (Simplicial Approximation Theorem)**.** *Let $f : |\mathcal{K}| \to |\mathcal{L}|$ be a continuous function between the underlying spaces of simplicial complexes $\mathcal{K}$ and $\mathcal{L}$. Then, for sufficiently large m, there exists a simplicial approximation to f*

$$\phi : Bary^m \mathcal{K} \to \mathcal{L}$$

Proof. Consider the open covering $\mathscr{C} = \{star(w) | w \in V(\mathcal{L})\}$ for $|\mathcal{L}|$. Since f is continuous, $f^{-1}(\mathscr{C}) = \mathscr{C}'$ is an open covering of $|\mathcal{K}|$. It follows from Lemma 6 that there exists an m such that, for each vertex v of $\mathrm{Bary}^m \mathcal{K}$, $star(v) \subseteq S \in \mathscr{C}'$. This means that for each vertex v of $\mathrm{Bary}^m \mathcal{K}$, there is some $w \in V(\mathcal{L}$ with $star(v) \subseteq f^{-1}(star(w))$, hence $f(star(v)) \subseteq star(w)$. We can therefore define a vertex map $\phi : \mathrm{Bary}^m \mathcal{K} \to \mathcal{L}$ such that, for each vertex v of $\mathrm{Bary}^m \mathcal{K}$, $f(star(v)) \subseteq star(\phi(v))$.By Lemma 4, ϕ is a simplicial approximation to f.

□

This theorem characterizes solvability of colorless tasks, which are tasks that are defined only by their input and output values (colorless tasks do not take process ids into account), since their input and output complexes do not need a chromatic description. Also the protocol complex for colorless tasks induced by the rounds of communication is a barycentric subdivision of the input complex.

However, this is not sufficient for colored tasks in the general asynchronous model. For this more general problem we will extend Theorem 2 for chromatic simplicial complexes.

Note that the importance of the Simplicial Approximation Theorem goes beyond distributed computing applications. Since it can be shown that every geometric realization of every simplex is convex, every continuous function between triangulated spaces is homotopic to a simplicial map.

Chapter 6

Continuous Tasks

We will consider the model introduced in Chapter 3, which we briefly recall. We have a set of processes that communicate through a single-writer, multi-reader shared memory.

We consider the wait-free asynchronous model, where each process runs at its own speed and up to n processes may fail by crashing. A crashed process cannot further read nor write and is indistinguishable from a process with an arbitrarily large delay. Since all but one process may crash, and a crashed process is indistinguishable from a slow one, it is not safe for a process to wait for other processes.

There are several equivalent variants for this model, each with the same computability power. We will use the wait-free iterated immediate snapshot model, where each process executes the same number of communication rounds. Each communication rounds consists of executing an immediate snapshot operation. An immediate snapshot consists of writing the state of the process into its memory segment and immediately after reading the whole shared memory (snapshot). Although the immediate snapshot operation seems to be more powerful than single reads and writes, it has been shown in [33] that it has the same computability power.

Recall that the definition of a distributed task T is given by a tuple $\langle I, O, \Delta \rangle$ where I and O are sets, and $\Delta : I \to 2^O$ is monotonically non-decreasing [33]. The intuition behind the definition is that I corresponds to the set of valid global inputs, O corresponds to the valid set of outputs, and Δ is a map that associates which outputs are considered to be a valid solution of the task T for a given input.

For example, binary 2-process consensus specified as a task $T = \langle I, O, \Delta \rangle$, reads:

$$\begin{aligned}
I &= \{(0,0),(0,1),(1,0),(1,1)\} \\
O &= \{(0,0),(1,1)\} \\
\Delta(0,0) &= \{(0,0)\} \\
\Delta(0,1) &= \{(0,0),(1,1)\} \\
\Delta(1,0) &= \{(0,0),(1,1)\} \\
\Delta(1,1) &= \{(1,1)\}
\end{aligned}$$

While this definition is very general and comprises a correct definition for many different scenarios in distributed computing, it considers certain tasks as valid even though there does not exist a protocol that solves them. For example, let us consider the consensus task $T*$ for any non-trivial wait-free model. The classical FLP result [1] shows that no protocol for such a model exists that solves $T*$. It would hence be simpler not to consider $T*$ as a valid task within a non-trivial wait-free model.

In [2], Herlihy and Shavit used a topological framework to fully characterize asynchronous shared memory tasks. They described a standard way of building a simplicial complex with the input and the output sets. This construction is based on the local views of each process, rather than on descriptions based on global states of the system.

The input complex $\mathcal{I} = \langle V(\mathcal{I}), F(\mathcal{I}) \rangle$ is given by its set of vertices $V(\mathcal{I})$ and its set of faces $F(\mathcal{I})$.

- $V(\mathcal{I}) = \{(p_i, v_i) \mid p_i \in \Pi,\ v_i \in V_i\}$ where V_i is the set of valid inputs for p_i.
- $F(\mathcal{I}) = \{\sigma \subseteq V(\mathcal{I}) \mid \sigma \text{ is part of a valid input configuration } \}$

The output complex $\mathcal{O} = \langle V(\mathcal{O}), F(\mathcal{O}) \rangle$ is given by its set of vertices $V(\mathcal{O})$ and its set of faces $F(\mathcal{O})$. The output complex is given in the same way as the input complex.

- $V(\mathcal{O}) = \{(p_i, v_i) \mid p_i \in \Pi,\ v_i \in \hat{V}_i\}$ where $\hat{V}_i$ is the set of valid outputs for p_i.
- $F(\mathcal{O}) = \{\sigma \subseteq V(\mathcal{O}) \mid \sigma \text{ is part of a valid output configuration } \}$.

We also consider a chromatic map $\chi : V(\mathcal{I}) \cup V(\mathcal{O}) \rightarrow \{1, \ldots, n+1\}$ that maps each pair (p_i, v) to its process id $\chi(p_i, v) = i$.

The protocol complex $\mathcal{P}_{\mathcal{M}} = {}_{\mathcal{M}}\langle V(\mathcal{P}), F(\mathcal{P}) \rangle$ for a given protocol P and model $\mathcal{M}$ is defined as

- $V(\mathcal{P}) = \{(p_i, v_i) \mid p_i \in \Pi,\ v_i \in \overline{V}_i\}$ where $\overline{V}_i$ is the set of valid minimal final views for p_i in protocol P under a given model $\mathcal{M}$.
- $F(\mathcal{P}) = \{\sigma \subseteq V(\mathcal{P})\}$ where σ corresponds to a valid configuration of minimal final views of a run.
- The chromatic function $\chi : V(\mathcal{P}) \rightarrow \Pi$ is given by the id of each process, that is $\chi(p_i, v_i) = p_i$.
- The decision map for a protocol $\mu : V(\mathcal{P}) \rightarrow V(\mathcal{O})$ is a color-preserving vertex map that maps final views of a process to valid outputs for a task.

Task solvability in this general model is characterized in the following way: A task $T = \langle I, O, \Delta \rangle$ is solvable in a model M if and only if there exists a protocol P such that the decision map for P, $\mu : \mathcal{P} \rightarrow \mathcal{O}$ is a chromatic simplicial map that preserves Δ, that is $\mu(\sigma) \in \Delta(\sigma)$ for any simplex σ.

According to Chapter 4, in the iterated immediate snapshot model, any protocol complex corresponds to a chromatic subdivision of $\mathcal{I}$. Using this observation, task solvability in the iterated immediate snapshot model is characterized through Theorem 1 in Chapter 5, the Asynchronous Computability Theorem. This theorem states that the protocol task is solvable if and only if there exists a chromatic subdivision of the input complex, $Ch(\mathcal{I})$ and a chromatic simplicial map $\mu : Ch(\mathcal{I}) \to \mathcal{O}$ from the chromatic subdivision of the input complex to the output complex, such that μ preserves the decision map Δ.

This result implies that any protocol that solves a task in the wait-free shared memory model induces a continuous function between the geometric realization of the input complex and the output complex. We will show that for any continuous map $f : |\mathcal{I}| \to |\mathcal{O}|$ between the input and output complexes that satisfies certain color preserving conditions also induces a protocol whose decision map approximates f. It is this correspondence that allows us to state task specification in terms of a continuous function between geometric realizations of simplicial complexes.

6.1 Contributions

Our main contribution is a new way to state valid tasks, in an asynchronous wait-free shared memory model, that corresponds exactly to solvable tasks. For that purpose, we define a set of properties for continuous functions between geometric realizations of simplicial complexes $\mathcal{I}$ and $\mathcal{O}$. We say that $f : |\mathcal{I}| \to |\mathcal{O}|$ is chromatic if it satisfies said conditions, and will give the colouring correspondence between chromatic functions $f : |\mathcal{I}| \to |\mathcal{O}|$ and solvable tasks: For any chromatic function $f : |\mathcal{I}| \to |\mathcal{O}|$ there exists a chromatic subdivision $Ch(\mathcal{I})$ of $\mathcal{I}$, and a simplicial map $\mu : Ch(\mathcal{I}) \to \mathcal{O}$ that approximates f. Conversely any protocol that solves a task in the asynchronous wait-free shared memory model induces a chromatic function $f : |\mathcal{I}| \to |\mathcal{O}|$.

Our second contribution is that we characterize solvable tasks in terms of continuous functions, in a different way than it was proposed in [33]. This provides an easily applicable criterion for deciding whether or not a given task has a solution in the asynchronous wait-free memory model, without explicitly using more complex tools such as calculating homology groups.

A third contribution is that for any task given in terms of a chromatic function, we can construct a protocol in the iterated immediate snapshot that solves the given task.

6.2 The Asynchronous Wait-free Shared Memory Model

As mentioned before, we will use the iterated immediate snapshot model. Slightly generalizing the single layer model introduced in Chapter 3, the model consists of a set of processes $\Pi = \{p_0, \ldots, p_n\}$ that communicate using a layered single-write snapshot shared memory $M_{k\times(n+1)}$, where k is the number of layers of the shared memory. Each process executes exactly k asynchronous communication rounds, corresponding to the k layers. During a given round j, process p_i writes atomically its full state into $M[j][i]$, and immediately afterwards, it takes a snapshot of $M[j]$ (i.e. p_i instantaneously reads $M[j][k]$ for each $0 \leq k \leq n$). Note that immediate snapshots may be scheduled simultaneously for several processes, and that different protocols differ only in number of communication rounds and the decision for each process.

Consider the protocol complex $\mathcal{P}$ for this model. We consider the final view of a process p_i to be its k memory snapshots. Since we are considering a shared memory model, the scheduling of a round induces a partial ordering of the participating processes snapshots (a process that is scheduled later is able to read more local views). We have assured in Chapter 4. that shows that the protocol complex $\mathcal{P}$ for a k-round protocol in the model corresponds to the standard $k-th$ chromatic subdivision of $\mathcal{I}$, the input complex.

6.3 Chromatic Functions and Continuous Tasks

Consider a continuous function $f : |\mathcal{I}| \to |\mathcal{O}|$ between geometric simplicial complexes induced by chromatic simplicial complexes $\mathcal{I}$ and $\mathcal{O}$ of dimension n. We consider a generalized chromatic map $\bar{\chi} : V(\mathcal{I}) \cup V(\mathcal{O}) \to 2^C$ that sends vertices to singleton sets. We further extend it to include points in the geometric simplicial complexes that do not correspond to vertices of the simplicial complexes: We define the extended chromatic map $\chi : |\mathcal{I}| \cup |\mathcal{O}| \to 2^C$ as:

$$\chi(x) = \bar{\chi}(carr(x))$$

,

where $carr(x)$ is the minimal face (from $\mathcal{I}$ if $x \in \mathcal{I}$ or $\mathcal{O}$ respectively) that contains x as an interior vertex.

Definition 2 (Continuous Chromatic Function)**.** *We say that f is chromatic with respect to $\mathcal{I}$ and $\mathcal{O}$ if it satisfies the following conditions:*

(1) For every $x \in |I|$, $\chi(f(x)) \subseteq \chi(x)$

(2) For every $x \in |I|$, $\varepsilon > 0$ and $c \in \chi(x)$, there exists x_c such that $carr(x_c) = carr(x)$, $d(x_c, x) < \varepsilon$ and $c \in \chi(f(x_c))$.

This definition corresponds intuitively to the idea that f maps the color structure of $\mathcal{I}$ into $\mathcal{O}$. Since color corresponds to process ids in the distributed computing framework, it makes sense to be interested in functions that preserve the color structure. Informally, (1) corresponds to the notion that a valid decision on a given set of processes S should only depend on processes from S; Figure 6.1 shows an example of a function that violates this condition. In addition, (2) Ensures that any point in $|\mathcal{I}|$ is always close to points that decide with each of its colors. Figure 6.2 shows an example where this does not hold.

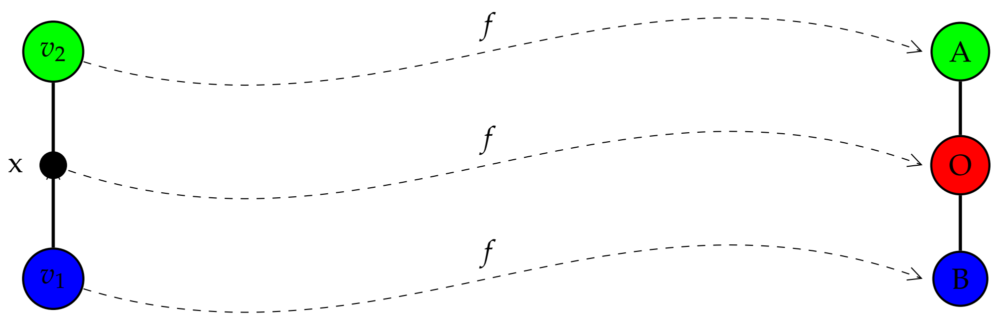

FIGURE 6.1: f is not chromatic since $f(x) = O$ and $\chi(O) \not\subset \chi(x)$

The following definition states formally what it means to be a chromatic approximation of a function. Although this definition is similar to the simplicial approximation concept used in algebraic topology [36], it is less restrictive in the topological sense, but has a new color preservation requirement. This will allow us to give a relationship between decision maps and continuous chromatic functions.

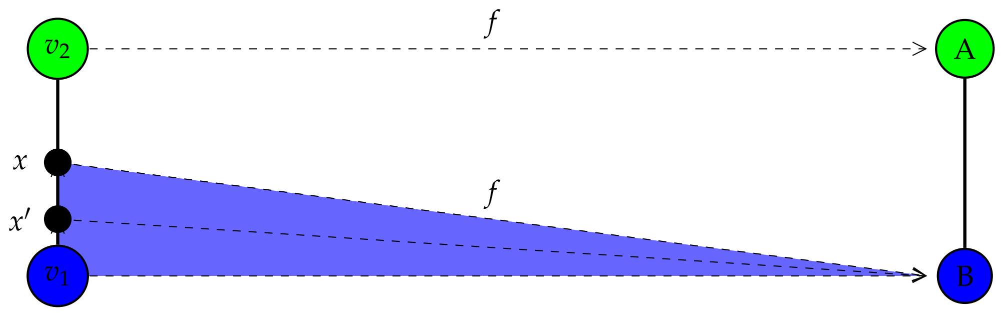

FIGURE 6.2: f is not chromatic since there is a region around x' that is mapped to blue, and $\chi(x') = \{blue, green\}$.

Definition 3. *Let $\mathcal{I}$ and $\mathcal{O}$ be chromatic simplicial complexes, and $f : |\mathcal{I}| \to |\mathcal{O}|$ be a continuous chromatic function. We say that $\mu : \mathcal{I} \to \mathcal{O}$ is a chromatic approximation to f if it is a chromatic simplicial map, and for every $v \in V(\mathcal{I})$, $carr(\mu(v)) \subseteq carr(f(v))$.*

Notice that chromatic approximations are different from the simplicial approximations defined in classical algebraic topology. While chromatic approximations may seem more permissive at first glance, it should be noted that it is required that they also preserve the chromatic structure.

We can proceed to define continuous tasks.

Definition 4 (Continuous Task)**.** *We say that $T = \langle \mathcal{I}, \mathcal{O}, f \rangle$ is a continuous task if $\mathcal{I}$ and $\mathcal{O}$ are pure chromatic simplicial complexes of the same dimension, and $f : |\mathcal{I}| \to |\mathcal{O}|$ is a continuous chromatic function.*

A continuous task uses a chromatic function in order to describe which outputs are valid for each input configuration. Later in this section, we will show that continuous tasks exactly coincide with the tasks that can be wait-free solved in asynchronous shared memory.

Definition 5. *We say that an algorithm A solves a continuous task $T = \langle \mathcal{I}, \mathcal{O}, f \rangle$, if A induces a subdivision of $\mathcal{I}$, $Sub(\mathcal{I})$, and a decision map $\mu : Sub(\mathcal{I}) \to \mathcal{O}$, such that for each $\sigma \in Sub(\mathcal{I})$, $|\mu(\sigma)| \subseteq f(|\sigma|)$.*

Definition 6. *We say that a continuous task T has a solution if there exists an algorithm A that solves T*

Note that a continuous task $T = \langle \mathcal{I}, \mathcal{O}, f \rangle$ induces a task $T' = \langle \mathcal{I}, \mathcal{O}, \Delta_f \rangle$, where $\Delta_f(\sigma) = \{carr_{\mathcal{O}}(f(x)) \mid x \in \mathcal{I} : \ carr_{\mathcal{I}}(x) = \sigma\}$. (Recall that a task is a tuple $\langle I, O, \Delta \rangle$ where Δ is a function that maps a valid input to a set of valid outputs).

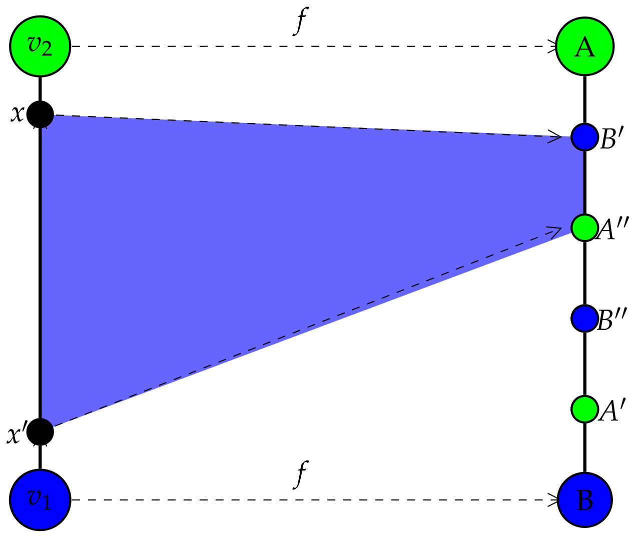

FIGURE 6.3: A continuous task is given by f. An input with values v_1 and v_2 is allowed to decide on any segment of AB but $B'A''$ is favored by f

.

Figure 6.3 shows a simple continuous task, similar to approximate agreement (can be extended to binary approximate agreement). The continuous task in Figure 3 induces the task $\langle \mathcal{I}, \mathcal{O}, \Delta_f \rangle$, where $\Delta_f(v_1, v_2) = \{AB', B'A'', A''B'', B''A', A'B\}$.

Lemma 8. *A continuous task $T = \langle \mathcal{I}, \mathcal{O}, f \rangle$ has a solution if and only if its induced task $T' = \langle \mathcal{I}, \mathcal{O}, \Delta_f \rangle$ has a solution.*

Theorem 3. *Let $T = \langle \mathcal{I}, \mathcal{O}, f \rangle$ be a continuous task, then T has a solution in the asynchronous shared memory model.*

This theorem is a direct consequence of Theorem 4, the proof of which relies on some technical lemmas.

Theorem 4. *Let $T = \langle \mathcal{I}, \mathcal{O}, \Delta \rangle$ be a task. T is solvable in the asynchronous wait-free shared memory model if and only if there exists a continuous task $T' = \langle \mathcal{I}, \mathcal{O}, f \rangle$, such that $f(|\sigma|) \subseteq |\Delta(\sigma)|$ for any input configuration σ.*

Proof.

First let's assume that $T = \langle \mathcal{I}, \mathcal{O}, \Delta \rangle$ is solvable. By the Asynchronous Computability Theorem, there exists k and $\mu : Ch^k(\mathcal{I}) \rightarrow \mathcal{O}$ such that μ is a chromatic simplicial map that preserves Δ. Consider $|\mu| : |\mathcal{I}| \rightarrow |\mathcal{O}|$ the continuous map induced by μ. By lemma 8, $|\mu|$ is chromatic since μ is a chromatic simplicial map. Therefore $\langle \mathcal{I}, \mathcal{O}, |\mu| \rangle$ is a continuous task. Notice that since $|\mu|$ is the affine map induced by μ, we have $|\mu|(\sigma) = |\mu(\sigma)|$, and since μ preserves Δ, also $\mu(\sigma) \subseteq \Delta(\sigma)$. It follows that $|\mu|(\sigma) \subseteq |\Delta(\sigma)|$.

Now lets assume that $\langle \mathcal{I}, \mathcal{O}, f \rangle$ is a continuous task such that $f(|\sigma|) \subseteq |\Delta(\sigma)|$. By Lemma 10, there exists a chromatic approximation of f, namely $\mu : Ch^k(\mathcal{I}) \rightarrow \mathcal{O}$. Let $\sigma' \in Ch^k(\mathcal{I})$ be a subdivided simplex of $\sigma \in \mathcal{I}$. Since μ is a chromatic approximation of f, we have $|\mu(\sigma')| \subseteq f(|\sigma'|) \subseteq f(|\sigma|) \subseteq |\Delta(\sigma)|$. Therefore μ is a chromatic simplicial map that preserves Δ, and by the Asynchronous Computability Theorem, task $T = \langle \mathcal{I}, \mathcal{O}, \Delta \rangle$ is solvable. □

Lemma 9. *Let $\mu : Ch^k(\mathcal{I}) \rightarrow \mathcal{O}$ be a chromatic simplicial map, then $|\mu| : |\mathcal{I}| \rightarrow |\mathcal{O}|$ is a chromatic continuous function.*

Lemma 10. *Lef $f : |\mathcal{I}| \rightarrow |\mathcal{O}|$ be a continuous chromatic function, there exists $k \in \mathbb{N}$ such that for any $\sigma \in Ch^k(\mathcal{I})$ (the k-th standard chromatic subdivision), $f(\sigma) \subseteq star(w)$ for some $w \in V(\mathcal{O})$.*

This property allows us to partially approximate f through a standard chromatic subdivision. Since we are interested in color preserving approximations, this is not enough, since we cannot necessarily map a whole simplex σ into a single vertex. However, this allows us to map a single vertex v of each facet σ into the center of a star w. We will now project the rest of the facet $\sigma \setminus \{v\}$ into the link of w. By composing the function with the projection over a single vertex w, we get a new continuous function. We can proceed iteratively until a trivial vertex map is left.

To summarize that this iterative subdivision, we give the following recursive definition.

Definition 7. *Let $\mathcal{I}$ and $\mathcal{O}$ be pure chromatic simplicial complexes of dimension k and $Sub(\mathcal{I})$ a chromatic subdivision of $\mathcal{I}$. We say that a continuous function $f : |\mathcal{I}| \rightarrow |\mathcal{O}|$ has the chromatic star property with respect to $Sub(\mathcal{I})$ if for each facet σ of $\mathcal{I}$, the following conditions hold:*

- *$f(|\sigma|) \subseteq star(w)$ for some $w \in \mathcal{O}$.*
- *If $k > 0$, for any $v \in \sigma$ such that $\chi(v) = \chi(w)$, $\pi_w \circ f : |\sigma \setminus v| \rightarrow |lk(w)|$ has the chromatic star property.*

Notice that both $\sigma \setminus v$ and $lk(w)$ have dimension $k - 1$, and that any chromatic vertex map has the chromatic star property.

Lemma 11. *Let $f : |\mathcal{I}| \to |\mathcal{O}|$ be a chromatic function and $Sub(\mathcal{I})$ a chromatic subdivision of $\mathcal{I}$ such that f has the chromatic star property with respect to $Sub(\mathcal{I})$. Then, there exists a chromatic simplicial map $\mu : Sub(\mathcal{I}) \to \mathcal{O}$ that is a chromatic approximation to f.*

Lemma 12. *Let $\mathcal{I}$ and $\mathcal{O}$ be pure chromatic simplicial complexes of dimension k and $f : |\mathcal{I}| \to |\mathcal{O}|$ a chromatic function. There exists a chromatic subdivision $Sub(\mathcal{I})$ such that f has the chromatic star property with respect to $Sub(\mathcal{I})$.*

Continuous tasks fully characterize solvable tasks in the asynchronous shared memory model. However, the chromatic function f also gives each valid output configurations a certain weight or value based on the volume of its preimage.

Consider the continuous task given by figure 3. Although any segment of AB is considered as a valid output, configuration $B'A''$ has a higher weight than any of the rest of valid configurations.

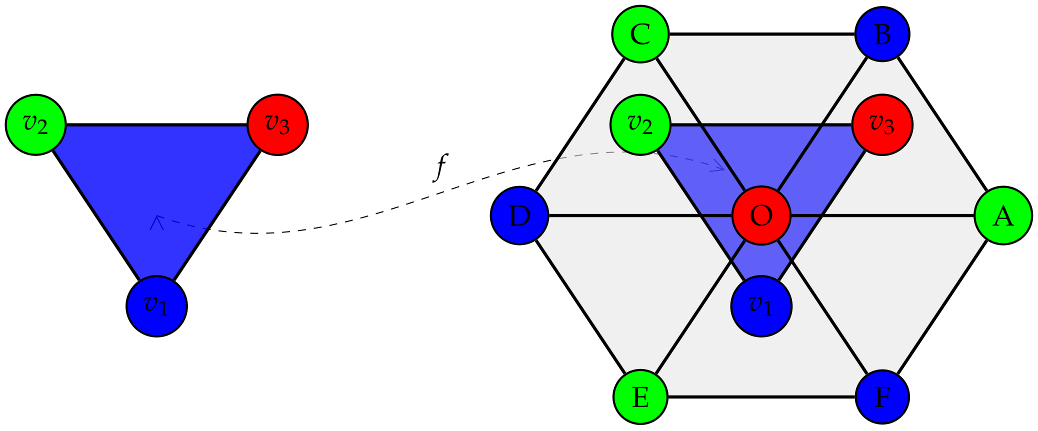

FIGURE 6.4: A simplex is mapped inside $star(o)$

6.4 Proofs for lemmas

In this section we include proofs for more technical lemmas.

Lemma 9. **Proof.** Since μ is a chromatic simplicial map, then for every $\sigma \in Ch^k(\mathcal{I})$, $\chi(\sigma) = \chi(\mu(\sigma))$. It follows then from the definition that $\chi(x) = \chi(|\mu|(x))$. Therefore $|\mu|$ is chromatic. □

Lemma 10. **Proof.** Consider $\mathcal{C} = \{star(w) \mid w \in V(\mathcal{O}\}$. $\mathcal{C}$ is an open covering of $\mathcal{O}$, therefore $f^{-1}(\mathcal{C})$ is an open covering of $\mathcal{I}$. Since $\mathcal{I}$ is compact, then $f^{-1}(\mathcal{C})$ has a Lebesgue number ε. Since the standard chromatic subdivision is a mesh-srhinking operation, there exists k such that $Ch^k(\mathcal{I})$ has mesh less than ε. Therefore, $\sigma \in f^{-1}(\mathcal{C})$ for any $\sigma \in Ch^k(\mathcal{I})$. Consequently, for any $\sigma \in Ch^k(\mathcal{I})$, $f(\sigma) \subseteq star(w)$ for some $w \in V(\mathcal{O})$. □

Lemma 11 **Proof.** Consider the vertex map $\mu : Sub(\mathcal{I}) \to \mathcal{O}$ given by $\mu(v) = w$ where $\chi(w) = \chi(v)$ and $w \in carr(f(v))$. μ is a chromatic vertex map, and from definition, it approximates f chromatically. It remains to see that it is also a simplicial map.

Let $\sigma \in Sub(\mathcal{I})$ be a facet. Since f has the chromatic star property with respect to $Sub(\mathcal{I})$, then $f(\sigma) \subseteq star(w)$ for some $w \in V(\mathcal{O})$. Since $\mathcal{I}$ and $\mathcal{O}$ are pure and of the same dimension, then there exists $v_1 \in \sigma$ such that $\chi(v_1) = \chi(w)$. From the definition of μ, it follows that $\mu(v_1) = w$. Notice that since μ is chromatic, then $\mu(\sigma \setminus v_1) \subseteq lk(w)$. Also notice that if κ is a simplex in $lk(w)$, then $\kappa \cup w$ is a simplex. Therefore it is enough to show that $\mu(\sigma \setminus v_1)$ is a simplex.

Since f has the chromatic star property, then $\pi_w \circ f$ also has the chromatic star property and there exists a w_2 such that $\pi_w \circ f(\sigma \setminus v_1) \subseteq star(w_2)$. Inductively, there exists w_i such that $\pi_{w_{i-1}} \circ \ldots \circ f(\sigma \setminus \{v_1, \ldots, v_{i-1}\}) \subseteq star(w_i)$.

Notice that $\mu(v_k)$ is a simplex, since it is only a vertex.
Since $\pi_{w_{k-2}} \circ \ldots \circ f(\sigma \setminus \{v_1, \ldots, v_{i-2}\}) \subseteq star(w_{k-1})$, then $\mu(v_k)$ and $\mu(v_{k-1})$ form a simplex. Inductively any $\{\mu(v_k), \mu(v_{k-1}), \ldots \mu(v_{k-i})\}$ form a simplex. In particular $\mu(\sigma)$ is a simplex.

Since μ is a chromatic vertex map and for any facet σ, $\mu(\sigma)$ is a simplex, then μ is a chromatic simplicial map. □

Chapter 7

Message Passing and BRS Theorem

In this chapter, we will analyze the topological structure of a general partition based impossibility result. This result is known as the BRS theorem, and it allows us to show k-set agreement impossibility for asynchronous message-passing models without the use of the Sperner's Lemma or other explicit topological tools.

We will also show that partition based arguments are not suitable for agreement impossibility in the shared memory model. We achieve this by showing that set-agreement is solvable in any shared memory model that allows a partition or that has a communication segmentation.

7.1 Model and preliminary definitions

We consider a set $\Pi = \{p_1, \ldots, p_n\}$ of processes each with its own unique identifier. We will primarily consider message passing protocols where each process has an individual message buffer for each of the processes in its local state. Valid messages are represented by a possibly infinite set M. Typically, we will consider deterministic full information protocols, where processes send messages that consist of the entire history of local states. We represent p_2 receives message $m \in M$ from p_1 by appending m to the message buffer of p_2 that corresponds to p_1. We will consider that the initial input is given as a message from a process p_i to itself, and that the decision value is also appended to the message buffer reserved to itself. A global state of the system is the vector of all local states.

We define a run of a given protocol as a valid infinite sequence of global states, in which eventually each non-crashing process reaches a final decision state. Messages that reach a process after a final decision state do not change the process' final decision. Note that the output of the process is defined by its local view at the first time it reached a decision state. We call such local views minimal final views. Notice that if run α and run β have the same minimal final views for each process, then they have the same decision outputs. Therefore we can restrict our attention to the equivalence classes of runs given by the minimal final views at each process.

We define a task $T_\Pi = \langle \mathcal{I}, \mathcal{O}, \Delta \rangle$ as a tuple where $\mathcal{I}$ and $\mathcal{O}$ are chromatic simplicial complexes that model the valid inputs and outputs for a set Π of processes and $\Delta : \mathcal{I} \to 2^{\mathcal{O}}$ is a valid decision function that maps valid input configurations to valid output configurations.

The input complex $\mathcal{I} = \langle V(\mathcal{I}), F(\mathcal{I})\rangle$ is given by its set of vertices $V(\mathcal{I})$ and its set of faces $F(\mathcal{I})$.

- $V(\mathcal{I}) = \{(p_i, v_i) \mid p_i \in \Pi,\ v_i \in V_i\}$ where V_i is the set of valid inputs for p_i.
- $F(\mathcal{I}) = \{\sigma \subseteq V(\mathcal{I}) \mid \sigma$ is part of a valid input configuration $\}$

The output complex $\mathcal{O} = \langle V(\mathcal{O}), F(\mathcal{O})\rangle$ is given by its set of vertices $V(\mathcal{O})$ and its set of faces $F(\mathcal{O})$. The output complex is given in the same way as the input complex.

- $V(\mathcal{O}) = \{(p_i, v_i) \mid p_i \in \Pi,\ v_i \in \hat{V}_i\}$ where $\hat{V}_i$ is the set of valid outputs for p_i.
- $F(\mathcal{O}) = \{\sigma \subseteq V(\mathcal{O}) \mid \sigma$ is part of a valid output configuration $\}$.

$\Delta : F(\mathcal{I}) \to F(\mathcal{O})$ is a function that satisfies:

- $\sigma \subseteq \mu \Rightarrow \Delta(\sigma) \subseteq \Delta(\mu)$.
- $\chi(\Delta(\sigma)) \subseteq \chi(\sigma)$.

We define the protocol complex $\mathcal{P}_{\mathcal{M}} = {}_{\mathcal{M}}\langle V(\mathcal{P}), F(\mathcal{P})\rangle$ for a given protocol P and model $\mathcal{M}$ as

- $V(\mathcal{P}) = \{(p_i, v_i) \mid p_i \in \Pi,\ v_i \in \overline{V}_i\}$ where $\overline{V}_i$ is the set of valid minimal final views for p_i in protocol P under a given model $\mathcal{M}$.
- $F(\mathcal{P}) = \{\sigma \subseteq V(\mathcal{P})\}$ where σ corresponds to a valid configuration of minimal final views of a run.
- The chromatic function $\chi : V(\mathcal{P}) \to \Pi$ is given by the id of each process, that is $\chi(p_i, v_i) = (p_i)$.
- The decision map for a protocol $\mu : V(\mathcal{P}) \to V(\mathcal{O})$ is a color-preserving vertex map that maps final views of a process to valid outputs for a task.

Notice that since the initial input values are self messages, each run is produced by a unique configuration of initial input values. Therefore, for each task T there exists a chromatic simplicial map $i_T : F(\mathcal{P}) \to F(\mathcal{I})$ such that $i(\sigma)$ is the initial input configuration for each process in σ.

A protocol P solves a task T_Π in model $\mathcal{M}$ if and only if the decision map for the protocol is a simplicial map that carries Δ. That is $\mu(\sigma) \subseteq \Delta(i_T(\sigma))$.

Notice that since the decision map μ needs to be chromatic, the decision map is determined by the mapping values at the facets. Therefore the facet decision map, $\hat{\mu} : \hat{F}(\mathcal{P}) \to 2^{V(\mathcal{O})}$, determines μ.

Definition 8 (Restriction of an Algorithm)**.** *Let A be an algorithm for a model $\mathcal{M} = \langle \Pi \rangle$, and $D \subseteq \Pi$ a nonempty set of processes. Consider a restricted model $\mathcal{M}' = \langle D \rangle$. The restriction of the model only requires that algorithm A is computationally compatible with $\mathcal{M}'$, i.e, A can be executed in $\mathcal{M}'$. The restricted model $\mathcal{M}'$ may consist of a smaller set of processes, and no assumptions are made about the synchrony of message passing in $\mathcal{M}'$. Therefore the runs in $\mathcal{M}'$ do not necessarily correspond to runs in $\mathcal{M}$ and vice versa. To restrict algorithm A for model $\mathcal{M}$ to an algorithm for model $\mathcal{M}'$, we just drop all messages sent from D to the outside. We call the restricted algorithm $A_{|D} = B$.*

Let $\mathcal{B}$ be the protocol complex for B executed in $\mathcal{M}'$. Notice that even if protocol B corresponds to a restriction of A to D, since message buffers for processes not in D are not present in $\mathcal{B}$, the protocol complex $\mathcal{B}$ is strictly different from any protocol complex that includes Π in its set of processes. Therefore, we need to define a way to extend the views in $\mathcal{B}$ in a way that could possibly match a protocol complex with processes Π executed in $\mathcal{M}$. The natural way of doing this is by adding empty message buffers denoted by $\perp$ for any process not in D.

We can define the extended complex of B with respect to Π, $\mathcal{A}_{\mathcal{D}}$, as follows:

- *$V(\mathcal{A}_{\mathcal{D}}) = \{(p, w, \perp, \ldots, \perp) \mid (p, w) \in V(\mathcal{B})\}$ and $\perp$ represents empty message buffers for processes in $\Pi \backslash D$.*
- *$F(\mathcal{A}_{\mathcal{D}}) = \{\sigma \subseteq V(\mathcal{A}_{\mathcal{D}}) \ : \ \exists \hat{\sigma} \in F(\mathcal{B}), (p, w, \perp, \ldots, \perp) \in \sigma \Rightarrow (p, w) \in \hat{\sigma}\}$. Notice that this definition induces an extended view copy of $F(\mathcal{B})$ in $\mathcal{A}_{\mathcal{D}}$.*

We will show that $\mathcal{A}_{\mathcal{D}}$ is isomorphic to $\mathcal{B}$.

Claim 1. *Let $\mathcal{A}_{\mathcal{D}}$ and $\mathcal{B}$ be as defined above. Then there exists a chromatic bijective simplicial map $\mu : \mathcal{A}_{\mathcal{D}} \to \mathcal{B}$.*

Proof.

Consider the following vertex map $\mu : V(\mathcal{A}_{\mathcal{D}}) \to V(\mathcal{B}), (p, s_w, m_1, \ldots, m_d, \bot, \ldots, \bot) \mapsto (p, s_w, m_1, \ldots, m_d)$.

We define $\gamma : V(\mathcal{B}) \to V(\mathcal{A}_{\mathcal{D}})$ as follows: $(p, s_w, m_1, \ldots, m_d) \mapsto (p, s_w, m_1, \ldots, m_d, \bot, \ldots, \bot)$. Notice that γ is well defined and is the inverse function of μ. This makes μ a bijection. It only remains to show that μ is simplicial and color preserving.

Consider $\sigma \in F(\mathcal{A}_{\mathcal{D}})$, from the definition of $F(\mathcal{A}_{\mathcal{D}})$ there exists $\hat{\sigma}$ such that $(p, w, \bot, \ldots, \bot) \in \sigma \Rightarrow (p, w) \in \hat{\sigma}$. It is clear that μ maps $\sigma \mapsto \hat{\sigma}$. Since $\hat{\sigma} \in F(\mathcal{B})$, then μ is a simplicial map. Notice that, by definition, μ preserves the process id, its local state and all message buffers in D. Therefore μ is a color preserving simplicial map. □

This isomorphic copy $\mathcal{A}_{\mathcal{D}}$ of the protocol complex $\mathcal{B}$ will be useful, since under certain conditions it corresponds to a subcomplex of $\mathcal{A}$.

Definition 9. *(Indistinguishability of Runs). Runs α and β are indistinguishable for a process p if p has the same sequence of states in α and β until p decides [4]. Notice that since p has the same sequence of states until decision for runs α and β, then the minimal final view for p is the same for both runs α and β. This means that the simplices σ_α and σ_β that correspond to runs α and β, share vertex $(p, s_w, m_1, \ldots, m_k)$, where $(s_w, m_1, \ldots, m_k)$ corresponds to the minimal final view of p at both runs α and β.*

This definition can be generalized to a set D of processes. We say that $\alpha \overset{D}{\sim} \beta$ if α is indistinguishable from β until decision for all $p \in D$. Following the previous observation, this translates naturally as D-skel(σ_α) = D-skel(σ_β) where D-skel$(\sigma_\alpha) = \{(p, w) \in \sigma_\alpha \ : \ p \in D\}$.

Definition 10. *(Compatibility of Runs). Let $\mathcal{R}$ and $\mathcal{R}'$ be sets of runs, possibly from different synchrony models. Runs $\mathcal{R}'$ are compatible with runs $\mathcal{R}$ for processes in D, denoted by $\mathcal{R}' \preceq_D \mathcal{R}$, if $\forall \alpha \in \mathcal{R}' \exists \beta \in \mathcal{R} : a \overset{D}{\sim} \beta$.*

If $\mathcal{R}$ and $\mathcal{R}'$ are sets of runs for the same protocol A, and in the same synchrony model $\mathcal{M}$, then both induce subcomplexes of a common protocol complex $\mathcal{A}$. We will call those subcomplexes $\overline{\mathcal{R}}$ and $\overline{\mathcal{R}'}$ respectively. We define *D-skel*$(\overline{\mathcal{R}'})$ as the subcomplex of $\overline{\mathcal{R}'}$ where all vertices of *D-skel*$(\overline{\mathcal{R}'})$ correspond to processes of D with views from $\overline{\mathcal{R}'}$. Given these definitions, it is clear that $\mathcal{R}' \preceq_D \mathcal{R}$ if and only if *D-skel*$(\overline{\mathcal{R}'}) \subseteq$ *D-skel*$(\overline{\mathcal{R}})$.

Notice that the previous equivalence is restrictive, since it only works for sets of runs from the same protocol and the same synchronicity model. However, we will give a more general definition that allows runs from different protocols and different synchrony models.

Definition 11. *(D-View embedding). Let A and B be protocols with a non-empty common set of processes S and $D \subseteq S$. Consider sets of runs $\mathcal{R}$ and $\mathcal{R}'$ from protocol A in model $\mathcal{M}$ and B in model $\mathcal{M}'$ respectively, and the subcomplexes $\overline{\mathcal{R}}$ and $\overline{\mathcal{R}'}$ that correspond to runs $\mathcal{R}$ and $\mathcal{R}'$,* *respectively.*

We say that $\overline{\mathcal{R}'}$ is D-view embedded in $\overline{\mathcal{R}}$ if for every $(p, w_s, m_1, \ldots, m_k) \in D\text{-}skel(\overline{\mathcal{R}'})$ the following conditions hold for every $1 \leq i \leq k$.

- $p_i \notin S \Rightarrow m_i = \perp$
- *There exists $(p, w_s, m'_1, \ldots m'_r) \in V(\overline{\mathcal{R}})$ such that $m'_i = \begin{cases} m_i & \text{if } p_i \in S \\ \perp & \text{if } p_i \notin S \end{cases}$*
- *$\mu : D\text{-}skel(\overline{\mathcal{R}'}) \to \overline{\mathcal{R}}$ defined by $(p, w_s, m_1, \ldots, m_k) \mapsto (p, w_s, m'_1, \ldots, m'_r)$ is a simplicial map.*

Notice that μ is an embedding of $D\text{-}skel(\overline{\mathcal{R}'})$, i.e. an injective simplicial map.

If both sets of runs come from the same algorithm ($B = A$) and the same synchrony model ($\mathcal{M} = \mathcal{M}'$), then the embedding is given by the inclusion $\iota : D\text{-}skel(\overline{\mathcal{R}'}) \to \overline{\mathcal{R}}$; $\iota(p, w) = (p, w)$. This matches with the previous observation that $\mathcal{R}' \preceq_D \mathcal{R}$ if and only if $D\text{-}skel(\overline{\mathcal{R}'}) \subseteq D\text{-}skel(\overline{\mathcal{R}})$ in this case.

More generally, we will show that we can formulate compatibility of runs in terms of D-view embedding.

Claim 2. *Let $\mathcal{R}$ and $\mathcal{R}'$ be sets of runs from algorithms A and B in model $\mathcal{M}$ and $\mathcal{M}'$ respectively. Let S be the set of common processes for $\mathcal{R}$ and $\mathcal{R}'$ and $D \subseteq S$. Then $\mathcal{R}' \preceq_D \mathcal{R} \Leftrightarrow \overline{\mathcal{R}'}$ is D-view embedded in $\overline{\mathcal{R}}$.*

Proof. We will show first that $\mathcal{R}' \preceq_D \mathcal{R} \Rightarrow \overline{\mathcal{R}'}$ is D-view embedded in $\overline{\mathcal{R}}$.

For $(p, w_s, m_1, \ldots, m_k) \in V(D\text{-}skel(\overline{\mathcal{R}'}))$, recall that $p \in D$ and $(w_s, m_1, \ldots, m_k)$ corresponds to a minimal final view for a run α in $\mathcal{R}'$. Notice that since $\mathcal{R}' \preceq_D \mathcal{R}$, there exists a run $\beta \in \mathcal{R}$ with $\alpha \overset{D}{\sim} \beta$. Then, there exists $(p, w'_s, m'_1, \ldots, \ldots, m'_r) \in V(\overline{\mathcal{R}})$ that corresponds to the minimal final view of p for run β. Since $(w'_s, m'_1, \ldots, \ldots, m'_r)$ and $(w_s, m_1, \ldots, m_k)$ are indistinguishable for p, their local states are the same, $w_s = w'_s$, message buffers for common processes match, and message buffers for processes that are not common must be empty. From these observations it follows that $m_i = \perp$ for all $p_i \notin S$ and that $m'_i = \begin{cases} m_i & \text{if } p_i \in S \\ \perp & \text{if } p_i \notin S \end{cases}$

Therefore $\mu : D\text{-}skel(\overline{\mathcal{R}'}) \to \overline{\mathcal{R}}$ is well defined. To show that it is an embedding of $D\text{-}skel(\overline{\mathcal{R}'})$ we need to show that it is injective and simplicial.

Suppose that $(p, w_s, m_1, \ldots, m_k)$ and $(p, w'_s, m'_1, \ldots, m'_k)$ both map to $(p, \hat{w}_s, \hat{m}_1, \ldots, \hat{m}_r)$. From the definition, then $w_s = \hat{w}_s = w'_s$. It is also required that the message buffers for processes not in S be empty, therefore, we only need to be concerned about message buffers for processes in S being the same. However, from the definition we also have that $m_i = \hat{m}_i = m'_i$ for all $p_i \in S$. Therefore μ is injective.

It only remains to show that μ is a simplicial map. Since by definition μ preserves process ids (it is a chromatic vertex map), we only need to check that facets (maximal simplices) of $D\text{-}skel(\overline{\mathcal{R}'})$ are mapped inside a facet in $\overline{\mathcal{R}}$. Consider a facet σ' in $D\text{-}skel(\overline{\mathcal{R}'})$, from the definition of $\mathcal{R}'$, there exists a run α that corresponds to σ'. Since $\mathcal{R}' \preceq_D \mathcal{R}$, there exists a run β, and a facet $\sigma \in \overline{\mathcal{R}}$ such that $\mu(\sigma') \subseteq \sigma$. Therefore μ is a D-view embedding from $\overline{\mathcal{R}'}$ into $\overline{\mathcal{R}}$.

Now, lets assume that $\overline{\mathcal{R}'}$ is D-view embedded into $\overline{\mathcal{R}}$. Then there exists a D-view embedding $\mu : D\text{-}skel(\overline{\mathcal{R}'}) \to \overline{\mathcal{R}}$. It follows from the definition of a D-view embedding and from the definition of $D\text{-}skel(\overline{\mathcal{R}'})$ that $\mathcal{R}' \preceq_D \mathcal{R}$.

□

Definition 12 (Decision map split). *Let $\mathcal{A}$ be the protocol complex for a given algorithm A on a model $\mathcal{M} = \langle \Pi \rangle$ and $\mathcal{A}'$ a non-empty subcomplex of $\mathcal{A}$. Let $D \subseteq \Pi$ be a set of processes in $\mathcal{M}$, $\overline{D} = \Pi \backslash D$ and $B = A_{|D}$ the restriction of algorithm A to D in a given model $\mathcal{M}' = \langle D \rangle$ with possibly different synchrony, resulting in protocol complex $\mathcal{B}$. We say that D splits the decision map of A at $\mathcal{A}'$ with respect to $\mathcal{M}'$ if $\mathcal{B}$ is D-view embedded in $\mathcal{A}'$ and $\mu_{|\mathcal{A}'} = \mu_{\mathcal{D}} * \mu_{|\overline{D}}$, where $\mu_{\mathcal{D}}$ is the decision map for restricted algorithm B at the extended complex $\mathcal{A}_{\mathcal{D}}$, $\mu_{|\mathcal{A}'}$ is the decision map μ of $\mathcal{A}$ restricted to $\mathcal{A}'$ and $\mu_{|\overline{D}}$ is the decision map μ restricted to $\overline{D}\text{-}skel(\mathcal{A}')$.*

Lemma 13. *Let $\mathcal{A}$ be the protocol complex for a given algorithm A in a model $\mathcal{M} = \langle \Pi \rangle$ and $\mathcal{A}'$ a non-empty subcomplex of $\mathcal{A}$. Let $D \subseteq \Pi$ be a set of processes in $\mathcal{M}$, $B = A_{|D}$ a restriction of algorithm A to D in a model $\mathcal{M}'$ with possibly different synchrony, $\mathcal{A}_{\mathcal{D}}$ the extended complex of B with respect to Π and μ be the decision map for $\mathcal{A}$, then D splits μ at $\mathcal{A}'$ with respect to $\mathcal{M}' \Leftrightarrow \mathcal{A}_{\mathcal{D}} = D\text{-}skel(\mathcal{A}')$.*

Proof. Suppose that D splits μ at $\mathcal{A}'$. Then $\mu_{|\mathcal{A}'} = \mu_{\mathcal{D}} * \mu_{|\overline{D}}$. Therefore

$$\mu_{|\mathcal{A}'}(p, w) = \begin{cases} \mu_{\mathcal{D}}(p, w) & \text{if } p \in D, \\ \mu(p, w) & \text{otherwise.} \end{cases}$$

By definition if $(p, w) \in V(\mathcal{A}')$ and $p \in D$, since $\mu_{\mathcal{D}}$ can be applied to (p, w), then $(p, w) \in Dom(\mu_{\mathcal{D}}) = \mathcal{A}_{\mathcal{D}}$. Therefore, $D\text{-}skel(\mathcal{A}') \subseteq \mathcal{A}_{\mathcal{D}}$. Conversely, since by Definition 7, $\mathcal{B}$ is D-view embedded in $\mathcal{A}'$. Therefore, by Claim 2, $\mathcal{A}_{\mathcal{D}}$ is a subcomplex of $D\text{-}skel(\mathcal{A}')$. Consequently $D\text{-}skel(\mathcal{A}') = \mathcal{A}_{\mathcal{D}}$.

Now lets assume that $D\text{-}skel(\mathcal{A}') = \mathcal{A}_{\mathcal{D}}$. Therefore in $\mathcal{A}'$, each process from D reached a decision before receiving messages from $\overline{D}$. Notice that since $\mathcal{A}_{\mathcal{D}}$ is isomorphic to $\mathcal{B}$, the processes from D at $\mathcal{A}'$ decide in the same way as the restricted algorithm. Therefore,

$$\mu(p, w) = \begin{cases} \mu_{\mathcal{D}}(p, w) & \text{if } p \in D, \\ \mu_{|\overline{D}}(p, w) & \text{if } p \in \overline{D}. \end{cases}$$

This shows that $\mu = \mu_{\mathcal{D}} * \mu_{|\overline{D}}$, and since $D\text{-}skel(\mathcal{A}') = \mathcal{A}_{\mathcal{D}}$, the protocol complex of the restricted algorithm $\mathcal{B}$ is D-view embedded in $\mathcal{A}'$. Hence D splits μ at $\mathcal{A}'$ with respect to $\mathcal{M}'$.

□

We will show in the next section that decision map splitting is equivalent to finding a partition of Π into $D, \overline{D}$ and a set of runs where D decides independently from $\overline{D}$. However, notice that $\overline{D}$ could use information from D to decide.

7.2 Topological BRS Theorem

Since we have already established an equivalence between topological conditions and run compatibility, we can proceed to state a slightly more general version of the BRS theorem from a topological perspective. Recall that the BRS theorem requires that some conditions (A)-(D) hold, in order to guarantee the k-set agreement impossibility. In the following lemmas, we will state topological properties that are slightly weaker than the original conditions (A)-(D).

Let $\mathcal{M} = \langle \Pi \rangle$ be a system model and A an algorithm that runs in model $\mathcal{M}$. Assume that each $p \in \Pi$ starts with a different input value. Also assume that there is a distinguished set $D \subseteq \Pi$, and a partition of $\Pi \backslash D = \overline{D}$ given by $D_1, \ldots, D_{k-1}$. Let $\{v_1, \ldots, v_{k-1}\}$ be a fixed set of different values.

The original BRS theorem formulates the following conditions for runs of algorithm A in model $\mathcal{M}$ and a restricted model $\mathcal{M}' = \langle D \rangle$.

dec-D If $p_j \in D$ then p_j receives no messages from any process in $\overline{D}$ until every process in D has decided.

dec-$\overline{\mathbf{D}}$ For every set D_i, value v_i was proposed by some $p \in \overline{D}$, and there is some $q \in D_i$ that decides v_i.

$\mathcal{R}_{(D)}$ denotes the set of runs from $\mathcal{M}$ where **dec-D** holds. $\mathcal{R}_{(D,\overline{D})}$ denotes the set of runs from $\mathcal{M}$ where both **dec-D** and **dec-$\overline{\mathbf{D}}$** hold.

(A) $\mathcal{R}_{(D)}$ is nonempty.

(B) $\mathcal{R}_{(D)} \preceq_D \mathcal{R}_{(D,\overline{D})}$.

(C) Consensus is not solvable in $\mathcal{M}'$.

(D) $\mathcal{M}'_{A_{|D}} \preceq_D \mathcal{M}_A$.

We can generalize the statement of the BRS theorem by slightly relaxing conditions (A)-(D).

Claim 3. *Let $\mathcal{M}$, $\mathcal{M}'$, A, D, $\overline{D}$ and D_i be as defined for the BRS-theorem. Then, (A)-(D) imply*

(A′) $\mathcal{R}_{(D)}$ is nonempty.

(B′) Consensus is not solvable in $\mathcal{M}'$

(C′) $\mathcal{M}'_{A_{|D}} \preceq_D \mathcal{R}_{(D,\overline{D})}$.

Proof. Notice that (A′) and (A) are the same, also (B′) and (C). Therefore it is enough to show that (B) and (D) imply (C′).

Let γ be a run in $\mathcal{M}'_{A_{|D}}$. From condition (D), there exists a run $\gamma' \in \mathcal{M}_A$ such that $\gamma \overset{D}{\sim} \gamma'$. Since γ' is indistinguishable from γ until decision for D, each process $p \in D$ in run γ' decides before receiving a message from $\overline{D}$. Therefore $\gamma' \in \mathcal{R}_{(D)}$.

Also notice that from (B) we get that there exists $\gamma'' \in \mathcal{R}_{(D,\overline{D})}$ such that $\gamma' \overset{D}{\sim} \gamma''$. Since $\overset{D}{\sim}$ is transitive, we have that $\gamma \overset{D}{\sim} \gamma''$. This proves that (C′) holds. □

Lemma 14. *Let $\mathcal{M}$, $\mathcal{M}'$, A, D, $\overline{D}$ and D_i be as defined for the BRS-theorem. Then (A′)–(C′) are equivalent to the following :*

1. *There exists a non empty subcomplex $\mathcal{A}'$ of $\mathcal{A}$ such that $D\text{-}skel(\mathcal{A}') = \mathcal{A}_{\mathcal{D}}$ (the extended complex for $A_{|D}$ with respect to $\mathcal{M}$).*
2. *For each D_i, the decision map $\mu_{|\mathcal{A}'}$ maps every view from D_i into a decision configuration that includes v_i as a decision value.*
3. *Each v_i is the input value for some process $p \in \overline{D}$ at subcomplex $\mathcal{A}'$.*
4. *Consensus is not solvable in $\mathcal{M}' = \langle D \rangle$.*

Proof.
First we will show that (A′)–(C′) ⇒ 1.–4.

It follows from (C′), that $\mathcal{B}$ is D-view embedded in $\overline{\mathcal{R}_{(D,\overline{D})}}$, where $\overline{\mathcal{R}_{(D,\overline{D})}}$ is the subcomplex of $\mathcal{A}$ that corresponds to runs $\mathcal{R}_{(D,\overline{D})}$. This implies that
$\mathcal{A}_{\mathcal{D}} \subseteq D\text{-}skel(\overline{\mathcal{R}_{(D,\overline{D})}})$.

We define

- $F(\mathcal{A}') = \{\sigma \in F(\overline{\mathcal{R}_{(D,\overline{D})}}) \ : \ D\text{-}skel(\sigma) \in F(\mathcal{A}_{\mathcal{D}})\}$.
- $V(\mathcal{A}') = \{(p,w) \in V(\mathcal{A}) \ : \ (p,w) \in \sigma \in F(\mathcal{A}')\}$.

Notice that this trivially implies $\mathcal{A}_{\mathcal{D}} \subseteq \mathcal{A}'$, so $\mathcal{A}' \neq \varnothing$. It also follows from the definition of $\mathcal{A}'$ that $\mathcal{A}' \subseteq \overline{\mathcal{R}_{(D,\overline{D})}}$, since $F(\mathcal{A}') \subseteq F(\overline{\mathcal{R}_{(D,\overline{D})}})$. Therefore condition 1 holds for $\mathcal{A}'$.

Conditions 2. and 3. follow from $\mathcal{A}' \subseteq \overline{\mathcal{R}_{(D,\overline{D})}}$.

Condition 4. is the same as (B′).

This shows that (A′)–(C′) ⇒ 1.–4.

We will now show that 1.–4. ⇒ (A′)–(C′).

From 1. we get that $\mathcal{B}$ is D-view embedded in $\mathcal{A}'$ since $\mathcal{A}_{\mathcal{D}} = D\text{-}skel(\mathcal{A}')$. From conditions 2. and 3. we get that $\mathcal{A}' \subseteq \overline{\mathcal{R}_{(\overline{D})}}$, therefore $\mathcal{A}_{\mathcal{D}} \subseteq \overline{\mathcal{R}_{(\overline{D})}}$, where $\overline{\mathcal{R}_{(\overline{D})}}$ is the complex of all runs where dec-$\overline{D}$ holds. Notice that since $\mathcal{A}_{\mathcal{D}} = D\text{-}skel(\mathcal{A}')$ is the extended view complex of the restricted algorithm, then for each run in $\mathcal{A}'$, the processes from D reached a decision before receiving messages from $\overline{D}$. Since $\mathcal{A}' \subseteq \overline{\mathcal{R}_{(D)}}$, we therefore get $\mathcal{A}' \subseteq \overline{\mathcal{R}_{(D,\overline{D})}}$. Since $\mathcal{A}'$ is non-empty, it follows that $\varnothing \neq \mathcal{R}_{(D,\overline{D})} \subseteq \mathcal{R}_{(D)}$. This shows that (A′) holds.

Condition 4. is the same as condition (B′).

Finally, since $\mathcal{A}_{\mathcal{D}} = D\text{-}skel(\mathcal{A}') \subseteq \overline{\mathcal{R}_{(D,\overline{D})}}$, condition (C′) holds.

□

This gives us a topological equivalence of a slightly stronger BRS theorem and some insight into the conditions required for stating a partition based impossibility theorem from a topological point of view. This is further developed in the following theorem.

Theorem 5 (Decision split theorem)**.** *Let $\mathcal{M} = \langle \Pi \rangle$ be a model and A, an algorithm that runs in $\mathcal{M}$. Let $\mathcal{A}$ be the protocol complex of A, $\mathcal{M}' = \langle D \rangle$ be a model with D a subset of Π, and μ the decision map for $\mathcal{A}$. Assume that the following conditions hold:*

(a) There exists a non-empty subcomplex $\mathcal{A}'$, such that D splits μ at $\mathcal{A}'$ with respect to $\mathcal{M}'$.

(b) Consensus is not solvable in $\mathcal{M}'$.

(c) Processes in $\overline{D} = \Pi \backslash D$ always decide at least $k-1$ input values from $\overline{D}$ for runs in $\mathcal{A}'$.

Then A does not solve k-set agreement.

Proof. Assume by contradiction that A solves k-set agreement. In particular $\mu_{|\mathcal{A}'}$ always decides on at most k input values. Notice that since μ always decides at least $k-1$ input values from $\overline{D}$ for runs in $\mathcal{A}'$, then $\mu_{|\overline{D}}$ always decides on at least $k-1$ input values from $\overline{D}$. From Condition (a) and Definition 7, we have that $\mu_{|\mathcal{A}'} = \mu_{\mathcal{D}} * \mu_{|\overline{D}}$. Since all possible decision values from $\mu_{\mathcal{D}}$ are disjoint from at least $k-1$ decision values in $\mu_{|\overline{D}}$, and $\mu_{|\mathcal{A}'}$ solves k-set agreement, then $\mu_{\mathcal{D}}$ always decides on at most 1 decision value. Therefore $\mu_{\mathcal{D}}$ solves consensus at $D\text{-}skel(\mathcal{A}') = \mathcal{A}_{\mathcal{D}}$. Since $\mathcal{A}_{\mathcal{D}} \cong \mathcal{B}$, where $\mathcal{B}$ is the protocol complex for algorithm B in model $\mathcal{M}'$, consensus is solvable in $\mathcal{M}'$. This contradicts Condition (b). □

Corollary 5. *Let $\mathcal{M} = \langle \Pi \rangle$ be a model and A an algorithm that runs in $\mathcal{M}$. Let $\mathcal{A}$ be the protocol complex of A, D a subset of Π and $\mathcal{M}' = \langle D \rangle$ a model in which the restricted algorithm $A_{|D}$ can be executed. Assume that the following conditions hold.*

(a′) There exists a non-empty subcomplex $\mathcal{A}'$, such that $D\text{-}skel(\mathcal{A}') = \mathcal{A}_{\mathcal{D}}$.

(b′) Consensus is not solvable in $\mathcal{M}'$.

(c′) Processes in $\overline{D} = \Pi \backslash D$ always decide at least $k-1$ input values from $\overline{D}$ for runs in $\mathcal{A}'$.

Then A does not solve k-set agreement on $\mathcal{M}$.

Proof. The result follows from decision splitting being equivalent to $\mathcal{A}_{\mathcal{D}} = D\text{-}skel(\mathcal{A}')$ and applying the decision split theorem. □

Corollary 6. *Let $\mathcal{M}$, $\mathcal{M}'$, A, D, $\overline{D}$ and D_i be as defined for the BRS-theorem. If conditions 1.–4. hold then A does not solve k-set agreement.*

Proof. Notice that Conditions 1. and 4. correspond to (a′) and (b′) respectively; 2. and 3. imply condition (c′). Therefore, Conditions 1.–4. satisfy (a′) –(c′). It follows from Corollary 12 that A does not solve k-set agreement. □

Corollary 7. *Let $\mathcal{M}$, $\mathcal{M}'$, A, D, $\overline{D}$ and D_i be as defined for the BRS-theorem. If conditions (A′)–(C′) hold, then A does not solve k-set agreement.*

Proof. It follows from Lemma 10 and Corollary 13 that A does not solve k-set agreement. □

7.3 Shared memory model

In the previous sections we developed a topological framework that allowed us to state a version of the BRS-theorem from the topological point of view. In this section we will restrict our attention to the specific shared memory model.

We consider a set of processes $\Pi = \{p_1, \ldots, p_n\}$ and a shared snapshot memory $M = (e, m_1, \ldots, m_n)$ where e is a buffer for global variables and each m_i corresponds to the local memory portion of process p_i in the snapshot memory.

In order to argue about k-set agreement impossibility in a more simple way, we will assume that the protocols are full information immediate snapshot layered protocols. Although these assumptions simplify our analysis, they do not reduce computability power. [33]

In this model, each process executes a predefined number r of asynchronous rounds or layers. Each round i consists of concurrent write-read snapshots. A write-read snapshot at layer i consists of writing the full view (a local state) of a process into its corresponding part of the memory, and immediately after writing, taking a snapshot of the views from processes at the same layer i.

The initial view of a process consists only of its input value, and therefore during the first round, each process only writes its input value to the shared memory.

Notice that in this model, each process' current view contains the history of previous views, so we need not be concerned with overwriting previous views in the shared memory.

Formally, we define the view for a given process p at round i in the following way.

- If $i = 0$, the view consists of a tuple (p, s, v) where p is the process id, s is the initial local state of p and v is the input value for p.
- If $i > 0$ then the view consists of a tuple $(p, s, v_1, \ldots, v_n)$ where each v_j corresponds to either the view of process p_j at the end of round $i - 1$ if the write-read execution for round i happened before or at the sime time as the write-read execution for process p. Otherwise $\perp$ represents that p finished its write-read round i before process j.

Notice that this definition for the processes views is very useful, since it has a nice combinatorial structure. More specifically, one can show an isomorphism between the standard chromatic subdivision of the input complex and the protocol complex for a general 1-layer immediate snapshot protocol. Since each layer i is only determined by the previous layers and the scheduling of layer i, it follows by induction that a $k + 1$ layered protocol complex corresponds to the $k + 1$-th chromatic subdivision of the input complex.

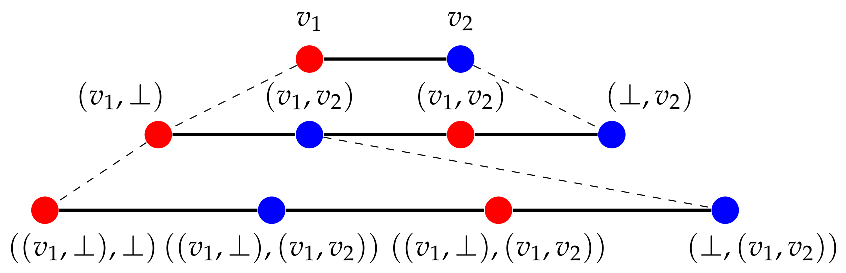

FIGURE 7.1: Input complex, 1-layer protocol complex and 2-layer protocol complex for two processes.

In the next section we will show that a partitioning argument for the processes is not suitable for proving the impossibility of set agreement by giving a general protocol that solves set agreement if all runs allow a partition of the processes.

7.4 Partition compatible runs

Notice that the central idea regarding partition arguments is to exploit limited communication between sets of processes.

Definition 13. *We define a set of runs S as partition compatible if for any run α, there exists a pair of participating processes p and q, and a round i such that p does not receive information from q during round i.*

We define the following 1-layer protocol P for the immediate snapshot model with a set of processes $\Pi = \{p_0, \ldots, p_n\}$.

Let p_i be a process and $(m_0, \ldots, m_n)$ its view.

Notice that since we are considering the iterated immediate snapshot model, then the protocol is determined only by the number of communication rounds (1 in this case) and the decision map.

We define $\mu(p_i, m_0, \ldots, m_n) = \begin{cases} m_i & \text{if } \perp = m_j \text{ for some } j \in \{1, \ldots, n\} \\ m_{(i+1) \bmod n+1} & \text{otherwise .} \end{cases}$ Notice that since μ always chooses the input value for p_i unless all other input values have been observed, then μ satisfies the validity condition.

Lemma 15. *Let S be a set of partition compatible runs for a 1-round immediate snapshot protocol, then μ solves n-set agreement in S.*

Proof. Let $\alpha \in S$ be a partition compatible run for a 1-round immediate snapshot protocol. Notice that if the set of participating processes of α has cardinality less than $n+1$, then the decision map μ trivially solves n-set agreement.

For the sake of readability, we will assume for the rest of the proof that any increase or decrease of a process index is considered to be modulo $n+1$.

Assume that α has $n+1$ participating processes. Let p_j be one of the processes that executed their round in the last step of the protocol. Since α has $n+1$ participating processes, then p_j observed all of the input values. From the definition of μ, then p_j decides the input value for p_{j+1}.

Assume by contradiction that μ does not solve n-set agreement for α. Since p_j did not decide on its input value, and μ does not solve n-set agreement, then p_{j-1} decides the input value for p_j. Since p_{j-1} decided the input value for p_j, then p_{j-1} also was among the processes that executed their round in the last step. It follows from an inductive argument, that all of Π executed their round in their last step. Therefore, α is an execution where all processes from Π executed concurrently. It follows from the immediate snapshot model specification that all processes hear from each other in α. This contradicts the fact that α was a partition compatible run. □

An immediate consequence for this lemma is that a partitioning argument cannot be used for showing n-set agreement impossibility for 1-round immediate snapshot protocols. However, in order to show that a partitioning argument cannot be used for a general shared memory protocol, we need to show this result for any number of layers.

In order to do so, we will define a decision map for any fixed number of layers.

Consider a general k-layered immediate snapshot protocol. Let p_i be a process, and $(m_0, \ldots, m_n)$ its final view. We denote $L^j(m_0, \ldots, m_n)$ as the view for p_i at layer j. Alternatively, if α is a run of an r-layered protocol and $s < r$ then $L^s(\alpha)$ denotes the s-layered protocol run induced by α. We say that a view $v = (m_0, \ldots, m_n)$ of a process at a layer s

is incomplete if either $\perp \in v$ or if there exists $j < s$ and $0 \leq r \leq n$ such that $L^j(m_r)$ is an incomplete view.

$$\mu^1 = \mu$$

$$\mu^{k+1}(p_i, m_0, \ldots, m_n) = \begin{cases} \mu^k(p_i, L^k(m_0, \ldots, m_n)) & \text{if } (m_0, \ldots, m_n) \text{ is an incomplete view} \\ \mu^k(p_{i+1}, L^k(m_0, \ldots, m_n)) & \text{otherwise} \end{cases}$$

Lemma 16. *Let S be a set of partition compatible runs for an k-round immediate snapshot protocol, then the decision map μ^k solves n-set agreement in S.*

Proof. We will proceed by induction over the number of rounds.

Notice that the base of induction is already given by the previous lemma.

Lets assume by induction that μ^k solves n-set agreement for a general k-layered immediate snapshot protocol.

Now lets consider a general $k+1$-layered protocol. Let α be a partition compatible run for the $k+1$ layered protocol. Assume that α has a full participating set Π (otherwise n-set agreement is trivial). Notice that $L^k(\alpha)$ is either partition compatible, or it is the run where every process heard from each other during all rounds up to k.

Assume that $L^k(\alpha)$ is partition compatible. Let $(p_i, m_0, \ldots, m_n)$ be a process and its view in the $k+1$ layer protocol. If $\perp \in v = (m_0, \ldots, m_n)$, then v is an incomplete view, and $\mu^{k+1}(p_i, v) = \mu^k(p_i, v)$. If $\perp \notin v$, since $L^k(\alpha)$ is partition compatible, there exists a pair of processes p_s and p_t and a round $u \leq k$ such that p_s does not hear from p_t during round u. Notice that since $\perp \notin v$, then m_s corresponds to the view of process p_s until layer k. From the previous assumptions, it follows that $\perp \in L^u(m_s)$. Therefore m_s is an incomplete view and thus v. It follows that $\mu^{k+1}(p_i, v) = \mu^k(p_i, v)$. By induction hypothesis we have that μ^k solves n-set agreement, therefore if $L^k(\alpha)$ is partition compatible, then μ^{k+1} solves n-set agreement.

Now lets assume that $L^k(\alpha)$ is not partition compatible. Therefore, any view for any process in α is complete up until layer k, thus, for any pair of process views v, v' we have that $L^k(v) = L^k(v')$. Consider a process p_s that executed its $k+1$ round last. Since α has a full participating set, then p_s has a complete view v'. From the definition, then $\mu^{k+1}(p_s, v') = \mu^k(p_{s+1}, L^k(v'))$. Since α is partition compatible, then there exists a minimal t such that p_{s+t} has an incomplete view. From the definition, it follows that $\mu^{k+1}(p_{s+t-1}, v') = \mu^k(p_{s+t}, L^k(v')) = \mu^{k+1}(p_{s+t}, L^k(v''))$ where v'' is the view from process p_{s+t}. It follows from a simple pigeonhole principle that μ^{k+1} solves n-set agreement in this case.

This completes the induction and the proof for any number of rounds. □

Theorem 6. *Let S be any set of partition compatible runs in the Iterated Immediate Snapshot model. Then there exists a protocol P that solves set agreement for any run in S.*

Proof. Let k be the number of rounds executed in S. From the previous lemma it follows that the generalized k-layered full information protocol with decision map μ^k solves set agreement. □

Chapter 8

Conclusions

8.1 Summary

This work was intended to be as self contained as possible, however some basic notions of topology and mathematics are required. We started by introducing the topoological framework with examples by modeling several logic puzzles.

We introduced the preliminary concepts and definitions needed to define task solvability, first by defining tasks and the computational models. Then we presented the preliminary definitions regarding combinatorial topology, simplicial complexes and maps. We also defined the input, output and protocol complexes. This definitions were useful as a link between combinatorial topology and distributed computing.

Once we established this connection, we showed a result that further explored the topological link with distributed computing. We gave a statement of the Asynchronous Computability Theorem (ACT) and the classical Simplicial Approximation Theorem.

These results form the basis of our first result, since we gave a chromatic generalization for the Simplicial Approximation Theorem. In order to achieve this goal, we proposed the new notion of continuous task that allows us to reformulate the ACT in terms of continuous functions.

We showed that any continuous task is equivalent to a task that is solvable in the asynchronous shared memory model.

Finally we expressed the topological point of view for partitioning arguments. In particular, we gave a fully topological formulation of the BRS theorem. This allowed us to partially understand the need for a topological tool in order to prove k-set agreement impossibility. With this in mind, we analyzed how a partitioning argument assumption defines a particular topology in the asynchronous shared memory model, and we showed that set agreement is solvable.

8.2 Future Work

Regarding continuous tasks, it would be interesting to consider other models different from shared memory that allow us to formulate a continuous task specification.

An open question would also be if continuous tasks can be composed and what advantages would composing them represent.

It would also be interesting to analyze for which problems are continuous tasks general enough, and if there is some model under which a continuous task is not solvable, but its induced task is solvable.

We consider that some interesting near future work would be to show that general partitioning assumptions along with some crash resilience property allows us to solve more agreement tasks, and therefore show that partitioning arguments are not suitable for a k-set agreement impossibility proof. We consider this towards showing that combinatorial topology (or equivalent) tools are necessary for proving k-set agreement impossibility.

Bibliography

[1] Michael J. Fischer and Nancy A. Lynch and M. S. Paterson (1985) Impossibility of Distributed Consensus with one Faulty Process *Journal of the ACM* 32(2), 374 – 382.

[2] Maurice Herlihy and Nir Shavit (1993) The asynchronous computability theorem for t-resilient tasks *STOC '93 Proceedings of the twenty-fifth annual ACM symposium on Theory of computing*, 111–120

[3] M. Herlihy, N. Shavit (1999). The topological structure of asynchronous computability. *J. ACM* 46(6): 858-923

[4] Martin Biely and Peter Robinson and Ulrich Schmid (2011). Easy Impossibility Proofs for k-set Agreement in Message Passing Systems. *Proceedings of the 30th Annual ACM Symposium on Principles of Distributed Computing (PODC'11)* 227–228.

[5] Danny Dolev, Nancy A. Lynch, Shlomit S. Pinter, Eugene W. Stark, and William E. Weihl(1986). Reaching Approximate Agreement in the Presence of Faults *J. ACM* .

[6] Borowsky, E. and E. Gafni (1993) Generalized FLP impossibility result for t-resilient asynchronous computations *Proceedings of the 25th ACM Symposium on Theory of Computing* 91–100.

[7] Attiya, H. and S. Rajsbaum (1996). The combinatorial structure of wait-free solvable tasks *Proceedings of the 10th International Workshop on Distributed Algorithms*, Lecture Notes in Computer Science 1151, Springer-Verlag, Berlin. 322–343.

[8] Yoram Moses and Sergio Rajsbaum (2012). A Layered Analysis of Consensus. *SIAM J. Comput* 31(4), 989–1021.

[9] Saks, M.E., & Zaharoglou, F. (1993). Wait-free k-set agreement is impossible: the topology of public knowledge. *SIAM J. Comput.* 29(5), 1149–1483.

[10] Maurice Herlihy and Sergio Rajsbaum and Mark R. Tuttle (2000). An Overview of Synchronous Message-Passing and Topology. *Electronic Notes in Theoretical Computer Science* 39(2), 1–17.

[11] Hagit Attiya and Armando Castañeda (2013). A non-topological proof for the impossibility of k-set agreement *Theoretical Computer Science* 512, 41-48.

[12] H. Attiya, A. Bar-Noy, D. Dolev (1995) Sharing memory robustly in message passing systems *J. ACM* 42 (1) 121–132.

[13] H. Attiya, J. Welch (2004). Distributed Computing: Fundamentals, Simulations, and Advanced Topics *Wiley-Interscience*, second ed.

[14] O. Biran, S. Moran, S. Zaks (1990) A combinatorial characterization of the distributed 1-solvable tasks *J. Algorithms*11 (3) 420–440

[15] E. Borowsky, E. Gafni (1997) A simple algorithmically reasoned characterization of wait-free computations *Proc. 16th ACM Symposium on Principles of Distributed Computing, PODC'97* , pp. 189–198.

[16] E. Borowsky, E. Gafni, N. Lynch, S. Rajsbaum (2001) The BG distributed simulation algorithm *Distrib. Comput* 14 (3) 127–146.

[17] Z. Bouzid, E. Gafni, P. Kuznetsov (2014) Strong equivalence relations for iterated models *Proc. 18th Int'l Conference on Principles of Distributed Systems* OPODIS 2014, in: Lecture Notes in Comput. Sci., vol. 8878, Springer, 2014, pp. 139–154.

[18] A. Castañeda, S. Rajsbaum, M. Raynal (2015) Specifying concurrent problems: beyond linearizability and up to tasks *29th Int. Symposium Distributed Computing, DISC'15, in: Lecture Notes in Comput. Sci., vol. 9363, Springer, 2015, pp. 420–435.*

[19] S. Chaudhuri (1993) More choices allow more faults: set consensus problems in totally asynchronous systems *Inform. and Comput.* 105 132–158.

[20] E. Gafni, S. Rajsbaum (2010) Distributed programming with tasks *14th Int'l Conference Principles of Distributed Systems, OPODIS'10* Lecture Notes in Comput. Sci., vol. 6490, Springer, , pp. 205–218.

[21] M.P. Herlihy (1991) Wait-free synchronization *ACM Trans* Program. Lang. Syst. 13 (1) 124–149.

[22] M.P. Herlihy, S. Rajsbaum (1997) The decidability of distributed decision tasks *Proc. 29th ACM Symposium on Theory of Computing,STOC'97, ACP Press, pp. 589–598.*

[23] M.P. Herlihy, S. Rajsbaum (2003) A classification of wait-free loop agreement tasks *Theoret. Comput. Sci.* 291 (1) (2003) 55–77.

[24] M.P. Herlihy, S. Rajsbaum (2010) The topology of shared-memory adversaries *Proc. 29th ACM Symp. on Principles of Distributed Computing, PODC 2010, ACM Press, 2010, pp. 105–113.*

[25] M.P. Herlihy, S. Rajsbaum, M. Raynal (2013) Power and limits of distributed computing shared memory models *Theoret. Comput. Sci.* 509 3–24.

[26] M.C. Loui, H.H. Abu-Amara (1987) Memory requirements for agreement among unreliable asynchronous processes *Adv. Comput. Res. 4 163–183.*

[27] C.R.F. Maunder Algebraic Topology *Dover Publications*1990, 393 pages.

[28] J.R. Munkres (2000) Elements of Algebraic Topology *Pearson*, 2nd edition, 2000, 537 pages.

[29] S. Rajsbaum (2010) Iterated shared memory models *Proc. 9th Latin American Symposium Theoretical Informatics, LATIN 2010,* Comput. Sci., vol. 6034, Springer, pp. 407–416.

[30] M. Raynal (2013) Concurrent Programming: Algorithms, Principles, and Foundations *Springer* ISBN 978-3-642-32027-9, 515 pages.

[31] M. Raynal, J. Stainer (2012) Increasing the power of the iterated immediate snapshot model with failures detectors *Proc. 19th Int'l Colloquium on Structural Information and Communication Complexity* SIROCCO'12, in: Lecture Notes in Comput. Sci., vol. 7355, Springer, pp. 231–242.

[32] E. Gobault, J.Ledent, S. Rajsbaum (2018) A Simplicial Complex Model for Dynamic Epistemic Logic to study Distributed Task Computability. *Proc. 9th Int. Symp. Games, Automata, Logics and Formal Verification (GandALF'18) Electronic Proceedings in Theoretical Computer Science* 277, pp. 73–87.

[33] Maurice Herlihy and Dmitry Kozlov and Sergio Rajsbaum (2013). Distributed Computing Through Combinatorial Topology. *Morgan Kaufmann* First Edition, 336 pp.

[34] Dmitry Kozlov (2008). Combinatorial Algebraic Topology. *Springer-Verlag Berlin Heidelberg* First Edition, 390 pp.

[35] Kozlov, D.N. (2015). Combinatorial topology of the standard chromatic subdivision and Weak Symmetry Breaking for 6 processes. *Springer, Chapter from Configuration Spaces*, 155-194.

[36] Edwin H. Spanier (1966) Algebraic Topology. *Springer-Verlag New York* First Edition, 548 pp.

[37] Jeffrey Seely (2016, April 17). Topological Data Analysis. Retrieved from `https://jsseely.github.io/notes/TDA/`

[38] Lev Gorodinsky (2017, September 18). The Asynchronous Computability Theorem. Retrieved from `https://medium.com/@eulerfx/the-asynchronous-computability-theorem-171e9d7b9423`

[39] Bill Casselman (2017, June) The Joy of Barycentric Subdivision. Retrieved from `http://www.ams.org/publicoutreach/feature-column/fc-2017-06`

[40] S. Rajsbaum, Perspectives, a little introduction to Distributed Computing via Topology. (2016).

[41] Chang, K., (2015) "A math problem from Singapore goes viral: When is Cheryl's birthday?" The New York Times (15 April 2015). [Chang 2015 available online (html)]

[42] Y. Afek, H. Attiya, D. Dolev, E. Gafni, M. Merritt, N. Shavit (1993) Atomic snapshots of shared memory *J. ACM* 40 (4) 873–890.