FAKULTÄT FÜR !NFORMATIK

Topology in Distributed Computing

DIPLOMARBEIT

zur Erlangung des akademischen Grades

Diplom-Ingenieur

im Rahmen des Studiums

Technische Informatik

ausgeführt von

Thomas Nowak
Matrikelnummer 0425201

an der
Fakultät für Informatik der Technischen Universität Wien

Betreuer: Univ.Prof. Dr. Ulrich Schmid

Wien, 18.03.2010

(Unterschrift Verfasser)

(Unterschrift Betreuer)

Erklärung

Thomas Nowak
Rechte Wienzeile 73/23
1050 Wien

Hiermit erkläre ich, dass ich diese Arbeit selbstständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit – einschließlich Tabellen, Karten und Abbildungen –, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

Wien, 18.03.2010

(Unterschrift)

Abstract

Topology is the general mathematical theory of convergence. Distributed computing is the formal investigation of communicating concurrent processes. We explore applications of topology to distributed computing in two directions: (1) Point-set topology and (2) algebraic topology.

We use the former to study the topological structure of infinite execution trees. This enables us to unify a number of impossibility proofs, in particular, the impossibility of *distributed consensus* — the task of all processes in a system agreeing on a single value — in various (close to) asynchronous systems with crash failures.

The latter is used to look into the combinatorial structure of *configurations*, i.e., the collection of current process states in the system. Configurations are regarded as simplices in a simplicial complex, and topological incompatibility of such complexes is utilized to prove the impossibility of a generalization of distributed consensus in certain systems. The particular problem considered is *k-set agreement*, which is the task of letting all processes agree to values within a set of at most k elements.

Kurzfassung

Topologie ist die mathematisch adäquate Art, um über Konvergenz zu sprechen. Distributed Computing ist das formale Studium von verteilten Systemen. Die Arbeit beschäftigt sich mit zwei Anwendungen der Topologie im Bereich des Distributed Computing: (1) Mengentheoretische Topologie und (2) algebraische Topologie.

Erstere wird verwendet, um die topologische Struktur von unendlichen Bäumen, die die Information über mögliche Ausführungen der Algorithmen subsumieren, zu untersuchen. Dieses Wissen wird verwendet, um einen einheitlichen Beweis der Unmöglichkeit von *Distributed Consensus* in mehreren Systemmodellen zu geben. Consensus ist das Einigen aller Prozesse des Systems auf einen einzigen Wert.

Zweitere wird verwendet, um die kombinatorische Struktur von *Konfigurationen*, also der Zusammenfassung aller lokaler Zustände der Prozesse, zu untersuchen. Hierbei wird eine Konfiguration als Simplex in einem Simplizialkomplex aufgefasst. Die topologische Unvereinbarkeit solcher Komplexe ermöglicht einen Beweis der Unmöglichkeit von *k-Set Agreement* in gewissen Systemen. Das ist eine Verallgemeinerung des Consensus-Problems: Es wird nicht mehr verlangt, dass sich die Prozesse auf nur einen Wert einigen, sondern es wird erlaubt, dass bis zu k unterschiedliche Werte auftreten.

Contents

1. Introduction

This thesis deals with applications of topology to distributed computing. These are twofold: Firstly, we use point-set topology to provide a unifying topological framework for consensus impossibility proofs. Secondly, we present the impossibility proof of k-set agreement by Herlihy and Shavit (1993) which uses algebraic topology.

1.1. Distributed Computing

Consider a system of N processes that communicate by means of passing messages. All processes take steps simultaneously at times $t = 0, 1, 2, \dots$ and in zero time. All message delays are equal to $1/2$, i.e., processes at time $t+1$ have received all messages sent in computing steps at time t. Processes are modeled as state machines and run a local algorithm which governs state transitions and message sendings. Interesting questions to ask might include:

(1) How many steps does it take until the last process has terminated?

(2) How many messages are sent in the execution of the algorithm?

(3) Is the algorithm correct, i.e., does it indeed fulfill its task specification?

The investigation of such questions is the realm of *distributed computing.*

We can spice things up a bit by varying model parameters. For example, we may allow more general message delays than fixing them all at exactly $1/2$. Likewise, we might choose not to fix the times at which processes take steps to exactly $0, 1, 2, \dots$ Of course, also the restriction that all processes take steps simultaneously might seem overly limiting.

We may also introduce the possibility of lost messages: In the completely synchronous model with message delays equal to $1/2$, suppose that in every time frame $[t, t+1)$, up to $N-1$ message may be lost. That is, these messages do not get delivered although all other messages are delivered timely. A surprising result (Santoro and Widmayer 1989) is that even in such a system with relatively few faults (there exist up to $N^2 - N$ point-to-point links; at most $N-1$ are lossy each round) it is *impossible* for any deterministic algorithm to solve *consensus.* Consensus is the task of all processes in the system agreeing on a single value.

1.2. Topology

Topology is the general mathematical theory of convergence. Its most popular special case is the study of metric spaces. It tackles questions like:

(1) Does the image of a continuous function $f : [0, 1] \to \mathbb{R}$ have a maximum?

(2) Does every Cauchy sequence converge?

(3) How many holes does a given manifold have?

(4) Can you cut two pizzas in half with a single cut, no matter how ugly they are shaped?

The immediate investigation of topological spaces is called *point-set topology*, which questions (1) and (2) can be attributed to. Another common technique is to assign algebraic structures to topological spaces, reason about relations between these structures and map these insights back into the world of topological spaces. This method is called *algebraic topology*.

1.3. Structure of the Thesis

Chapter 2 introduces distributed computing as a discipline and presents formal system models. In Chapter 3, we talk about an important problem specification in distributed computing: k-set agreement and its important special case, consensus. Chapter 4 investigates execution spaces of distributed algorithms by means of point-set topology and provides a unified proof of the impossibility of consensus in some important system models. Chapter 5 deals with methods from algebraic topology. It explains the approach taken by Herlihy and Shavit (1993) to prove the impossibility of k-set agreement in the presence of up to k crash failures. A summary of the thesis is given in Chapter 6. Appendix A gives a self-contained introduction to point-set topology.

1.4. A Word on Notation

The purpose of this section is to introduce some conventions of the mathematical notation used in the thesis.

We denote the set of real numbers by $\mathbb{R}$ and the set of non-negative real numbers by $\mathbb{R}^+$. Real intervals are written with round and square parentheses, e.g., $[0, 1) = \{x \in \mathbb{R} \mid 0 \leqslant x < 1\}$. For a set $X \subset \mathbb{R}$, $\inf X$ denotes the infimum of X and $\sup X$ denotes its supremum. The letter $\mathbb{Z}$ denotes the set of integers. We set $\mathbb{N} = \{k \in \mathbb{Z} \mid k \geqslant 1\}$ and $\omega = \{k \in \mathbb{Z} \mid k \geqslant 0\}$.

For arbitrary sets A and B, we write $B^A = \{f : A \to B\}$ to denote the set of all functions with domain A and range B. For a mapping $f : A \to B$ and subsets $A' \subset A$ and $B' \subset B$, we define $f[A'] = \{f(x) \mid x \in A'\}$ and $f^{-1}[B'] = \{x \in A \mid f(x) \in B'\}$. The predicate $A \subset B$ means $\forall x (x \in A \Rightarrow x \in B)$ and we set $P(A) = \{A' \subset A\}$. If M is a set of sets, then $\bigcup M$ denotes the set $\{x \mid \exists A \in M : x \in A\}$.

If (X, d) is a metric space, $x \in X$ and $\varepsilon > 0$, then we write

$$B_\varepsilon(x) = \{y \in X \mid d(x, y) < \varepsilon\}. \tag{1.1}$$

If X is a topological space and $S \subset X$, then $\overline{S}$ denotes the closure of S in X.

Additional notation will be defined when necessary.

2. Distributed Computing Models

This chapter introduces the field of distributed computing to the extent needed to present the results in subsequent chapters. We start by examining some questions that are tackled and then introduce a number of mathematical models that are used in distributed computing.

2.1. Introduction

Distributed computing (Attiya and Welch 2004, Lynch 1996) is the investigation of concurrent processes that communicate by means of some communication medium. Commonly, processes are modeled as deterministic state machines taking steps (performing state transitions) in zero time. Examples of communication media include point-to-point RS232 links, a common data bus or a shared memory area allocated by the Linux kernel. These types of communicating differ in a number of properties:

While changes to a shared memory are potentially visible immediately, *messages* which were sent at time t may arrive at time $t + \delta$ with $\delta > 0$, i.e., the message sent at time t is *not* immediately visible to the receiver. The transmission delay on a bus might be equal for all processes, while transmission delays on point-to-point links might be different for different links (processes).

In message-passing systems, a fundamental distinction is whether message delays are bounded or not. Message delays are *bounded* if there is a constant Δ such that every message sent at time t is guaranteed to have arrived at time $t + \Delta$. Systems that lack this property are called *message asynchronous*.

Other important properties of communicating distributed systems are *process synchrony* properties. The most process synchronous system imaginable might be a system in which all processes run at exactly the same speed, i.e., steps of processes are triggered by perfectly synchronous hardware clocks. A most process asynchronous system is one in which no information whatsoever is available on when processes take a step (perform a state transition). Of course, systems with more synchrony allow for harder problems to be solved than systems with weaker synchrony. It is, for example, impossible to do any kind of real-time clock synchronization in completely asynchronous systems. A major problem, however, is to determine whether one system is “more synchronous” than some other system (e.g., Dolev, Dwork, and Stockmeyer 1987) and for many pairs of systems, none is more synchronous than the other.

Things get even more complicated when components may fail, in particular, in asynchronous systems where no upper bound on message delays or inter-step times of processes exist. The seminal work of Fischer, Lynch, and Paterson (1985) shows that it is not possible in such systems for processes to even agree on a single value (i.e., *consensus* is not possible).

The following sections introduce a number of popular models for distributed systems.

2.2. Asynchronous Message Passing à la FLP

Consider a system of N concurrent processes that communicate by means of point-to-point links, i.e., every process can send messages to any other process. *Asynchronous* message-passing deserves its name because

(1) there is no upper bound on the transmission delay of messages and

(2) there is no upper bound on the inter-step time of processes, i.e., there is no Φ such that a process that took a step at time t is guaranteed to have taken its next step by time $t + \Phi$.

The assumption coverage of this model, i.e., its ability to accurately describe *real* systems, is quite broad since it does not limit the timing behavior in any way.

An algorithm in the asynchronous message-passing system model consists of a state machine for each of the N processes. State changes occur when a process takes a step and the transition function depends on the current internal state and received messages. Apart from the internal state transition, a process may also send messages to other processes. The structure of such a computing step is depicted in Figure 2.1.

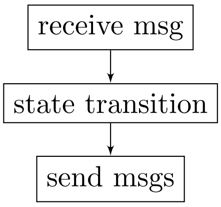


Figure 2.1.: Structure of a single computing step

Major problems arise when processes are allowed to crash, i.e., cease to take subsequent steps. The asynchronous nature of the system model prohibits distinguishing processes which have crashed from processes whose messages are very slow.

2.2.1. A Formal Description

In the asynchronous message-passing system model (Fischer, Lynch, and Paterson 1985), a system consists of N *processes* numbered $1, 2, \ldots, N$ which possess an *internal state*, a *transition function* δ and a *sending function* β. The transition function

δ maps pairs (s, m), consisting of the local state and a received message, to some internal state. A *message* is a pair (p, m) where p is a process name and m is a message content drawn from a pool of possible message contents $\mathcal{M}$ or the void value $\perp$. The sending function β maps pairs (s, m) as above to a finite set of (sent) messages. Every process has a distinguished subset of its set of states — the set of *initial states*. A *configuration* is a tuple $(s_1, s_2, \dots, s_N)$ of internal states of the processes, together with the set of in-transit messages — the *message buffer*.

An important point to understand is the relationship between the two notions of *event* and *step*, which we will define now. An *event* in the classical asynchronous message-passing model is a message. If in some configuration C, the message buffer holds the message (event) $e = (p, m)$, then we say that event e is *applicable* to configuration C and we may *apply* e to C by the following means: We define the *successor configuration* $C' = e(C)$ by the following procedure.

(1) Remove $e = (p, m)$ from the message buffer.

(2) Determine the internal successor state of process p by invoking the transition function δ using message content m and p's current state.

(3) Determine, and add to the message buffer, the messages (q, n) sent by process p to processes q by invoking the sending function β.

The pair (C, C') where $C' = e(C)$ for some event e is called a *step*. If we can apply event e to configuration C, we say that e is *applicable* to C. An event has the ability to trigger different steps, depending upon the configuration it is applied to.

An infinite sequence of events that are in turn applicable to C is called a *schedule* starting from C. If $(e_1, e_2, \dots)$ is a schedule starting from configuration C, we define the *corresponding sequence of steps* as follows: We set $C_0 = C$ and $C_{k+1} = e_k(C_k)$ for $k \geqslant 0$; the corresponding sequence of steps is then defined to be $\big((C_0, C_1), (C_1, C_2), (C_2, C_3), \dots\big)$. Such a corresponding sequence of steps is called a *run* or an *execution*.

A process is called *non-faulty* or *correct* in some run or schedule if it takes steps infinitely often. A process is called *faulty* if it is not non-faulty. A run or schedule is called *admissible* (with respect to the model parameter $f \geqslant 0$) if every message sent to non-faulty processes is received and at most f processes are faulty.

2.3. Omission Failure Model

In the *synchronous* message-passing model (Lynch 1996, Part I), all processes $p_1, p_2, \dots, p_N$ take their steps at the same time, e.g., every process takes a step at times $t = 0, 1, 2, 3, \dots$ Furthermore, there *does* exist an upper bound on message delays, namely every message sent at time t is delivered *before* time $t + 1$. That is, processes execute in lock-step rounds: Every process is guaranteed to have received all messages that were sent to it before the current computing step. Figure 2.2 contains a space-time diagram of a synchronous execution; the diagonal arrows indicate messages.

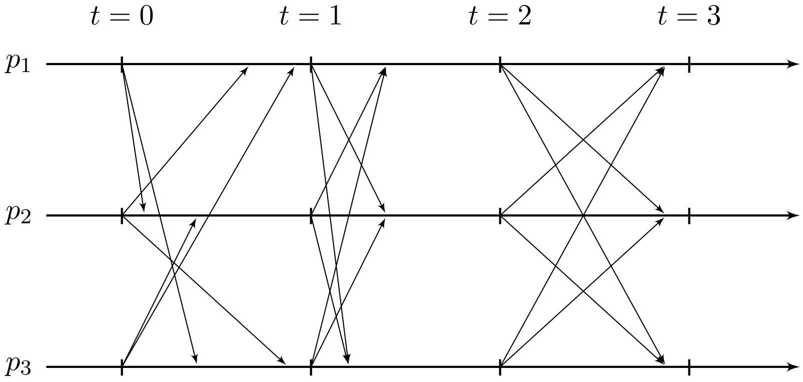


Figure 2.2.: Synchronous message-passing

In the synchronous *omission failure* model (Santoro and Widmayer 1989, Section 4.1), in every round, i.e., in every time interval $[t, t+1)$, up to $N-1$ messages may be lost. These omissions create difficulties and yield a number of impossibility results, because the adversary can completely silence a process by omitting all of its outgoing messages.

2.3.1. A Formal Description

In the synchronous omission failure system model, a system consists of N *processes* numbered $1, 2, \ldots, N$ which possess an *internal state*, a *transition function* δ and a *sending function* β. The transition function δ maps pairs (s, M), consisting of the local state and a set of (received) messages, to some internal state. A *message* is a pair (p, m) where p is a process and m is the *message content* taken from a set $\mathcal{M}$ of possible message contents. The sending function β maps an internal state s to a set M of messages such that every other process occurs in the first component of an element in M. Every process has a distinguished subset of its set of states — the set of *initial states*. A *configuration* is a tuple $(s_1, s_2, \ldots, s_N)$ of internal states of the processes.[1]

An *event* in the model is a set $O \subset \{1, \ldots, N\}^2 \setminus \{(1,1), (2,2), \ldots, (N,N)\}$ with $|O| \leqslant N-1$, the set of *omissions*. We define the *successor configuration* $C' = O(C)$ by the following procedure.

(1) Determine the sent messages of all processes by invoking the sending functions β.

[1] Contrary to the asynchronous message passing model, it is no longer necessary to remember the state of the medium in a configuration. This is because every message is received in the same step in which it was sent. See below for the exact step semantics.

(2) Ignore all messages over links in O.

(3) Determine the internal successor state of all processes by invoking the transition functions δ using the newly received (non-ignored) messages.

The pair (C, C') where $C' = O(C)$ for some event O is called a *step*.

An infinite sequence of events is called a *schedule*. If $(O_1, O_2, \dots)$ is a schedule and C is a configuration, we define the *corresponding sequence of steps* as follows: We set $C_0 = C$ and $C_{k+1} = O_k(C_k)$ for $k \geqslant 0$; the corresponding sequence of steps is then defined to be $\big((C_0, C_1), (C_1, C_2), (C_2, C_3), \dots\big)$. Such a corresponding sequence of steps is called a *run* or an *execution*.

A process is called *non-faulty* or *correct* in some run or schedule if infinitely many message sent by it get delivered. A process is called *faulty* if it is not non-faulty.

2.4. Asynchronous Shared Memory

In this section, we consider a system of N processes communicating by means of M *shared read-write registers*. These registers can hold an unbounded amount of information and support two types of operations: `read` and `write`. Operation $\texttt{read}(R)$ returns the value of register R and operation $\texttt{write}(R, v)$ writes value v to register R. The fundamental limitation in this model is that processes can perform only one of the operations `read` and `write` in a single computing step. Hence a process performing a `write` does not know which value it overwrites.

As in asynchronous message-passing, there is no upper bound on inter-step times of processes.[2] Also, the possibility of processes crashing introduces difficulties.

A sometimes convenient simplification is to limit registers to be *single-writer* registers. That is, a shared register has a single process assigned to it which is the only process that may write to the register. It is known (Attiya and Welch 2004, Theorem 10.9) that this is not a serious restriction.

2.4.1. A Formal Description

In the asynchronous shared memory system model (Attiya and Welch 2004, Section 4.1), a system consists of (a) N *processes* numbered $1, 2, \dots, N$ which possess an *internal state*, a *transition function* δ and a *shared memory operation function* β and (b) M shared read-write registers which possess a *value*. The shared memory operation function β maps an internal state s to a shared memory operation, i.e., $\texttt{read}(R)$ or $\texttt{write}(R, v)$. The transition function δ maps pairs (s, v), consisting of the local state and the return value of the shared memory operation $\beta(s)$, to some internal state. Every process has a distinguished subset of its set of states — the set

[2] However, there is no delay between performing a `write` and the time the written value becomes visible to other processes. Thus, message delay $\delta = 0$ in the language of message passing models.

of *initial states*. A *configuration* is a tuple $(s_1, s_2, \ldots, s_N)$ of internal states of the processes, together with a tuple $(v_1, v_2, \ldots, v_M)$ of shared memory register values.

An *event* in the asynchronous shared memory model is a process number $j \in \{1, 2, \ldots, N\}$. We define the *successor configuration* $C' = j(C)$ by the following procedure.

(1) Determine the next shared memory operation by process p_j by invoking the shared memory operation function β.

(2) Perform the shared memory operation by process p_j, i.e., change the register value in case of a `write` operation.

(3) Determine the internal successor state of process p_j by invoking the transition function δ using the return value from (2).

The pair (C, C') where $C' = j(C)$ for some event j is called a *step*.

An infinite sequence of events that are in turn applicable to C is called a *schedule* starting from C. If $(j_1, j_2, \ldots)$ is a schedule starting from configuration C, we define the *corresponding sequence of steps* as follows: We set $C_0 = C$ and $C_{k+1} = j_k(C_k)$ for $k \geqslant 0$; the corresponding sequence of steps is then defined to be $\big((C_0, C_1), (C_1, C_2), (C_2, C_3), \ldots\big)$. Such a corresponding sequence of steps is called a *run* or an *execution*.

A process is called *non-faulty* or *correct* in some infinite run or schedule if it takes steps infinitely often. A process is called *faulty* if it is not non-faulty. A run or schedule is called *admissible* (with respect to the model parameter $f \geqslant 0$) if at most f processes are faulty.

In case of *single-writer* registers, the set of allowed operations that may occur as the image of functions β is restricted such that there do not exist two processes p_i and p_j with operation functions β_i and β_j that can perform writes to a common register R.

2.4.2. Atomic Snapshots

A system with shared read-write registers supports *atomic snapshots* if there exists, besides `read` and `write`, a third operation, namely `scan`() which returns all register values at once, i.e., a tuple $(v_1, v_2, \ldots, v_M)$ of register values.

2.5. Safety and Liveness

The notions of safety and liveness properties were introduced by Lamport (1977) and have been well adopted in the distributed computing community. Lamport used these notions to subdivide correctness proofs of programs into smaller and more homogeneous pieces.

Intuitively, a safety property is the statement that “something will *not* happen” (Lamport 1977). For instance, take the sentence “No message is ever sent.” The “thing” that should not happen according to this statement is that a message is sent. At any time in an execution, if already a message was sent, there is no way that the execution fulfills the above safety property, no matter how the execution continues. Hence if a “bad thing” happened in an execution prefix, any execution that extends this prefix does not fulfill the safety property.

A liveness property is the statement that “something *must* happen” (Lamport 1977). An example would be the sentence “Every message that was sent is eventually received.” The important point is that at any time in an execution, even if not all sent messages were received yet, it is still possible that the execution fulfills the above liveness property (because the message can be received later). Hence for any finite execution prefix, there exists an execution extending this prefix that fulfills the liveness property.

The immediate formalization of these two notions is contained in the following definition.

Definition 2.1. Let $\mathcal{S}_\mathcal{A}$ be the set of admissible executions of some algorithm $\mathcal{A}$. A *property* of executions is a subset $\mathcal{P} \subset \mathcal{S}_\mathcal{A}$.

We call a property $\mathcal{P}$ a *safety* property if the following holds: For all $E \in \mathcal{S}_\mathcal{A} \setminus \mathcal{P}$ exists some $n \in \mathbb{N}$ such that every $E' \in \mathcal{S}_\mathcal{A}$ that coincides with E in the first n components holds $E' \notin \mathcal{P}$. We call a property $\mathcal{P}$ a *liveness* property if the following holds: For all $E \in \mathcal{S}_\mathcal{A}$ and every $n \in \mathbb{N}$ there exists an $E' \in \mathcal{P}$ that coincides with E in the first n components. □

It should be noted that the intuitive meaning of these notions is sometimes in conflict with Definition 2.1, in particular in the presence of failures. An investigation of this problem and alternative definitions were given by Charron-Bost, Toueg, and Basu (2000).

3. Problem Specifications

In this chapter, we discuss two prominent problems in distributed computing: the *consensus* problem and the *k-set agreement* problem which is a generalization of consensus. By the term "problem" we mean a specification on the behavior of an algorithm, which is said to "solve a problem" if all its executions satisfy the specification. The reason why we introduce exactly these two problems is firstly their fundamentality and secondly that we will prove impossibility of their solution in specific system models in later chapters.

3.1. Consensus

Informally, consensus is the task of getting all processes in a distributed system to agree on a single value. It is known (e.g., Fischer, Lynch, and Paterson 1985, Dolev, Dwork, and Stockmeyer 1987, Fich and Ruppert 2003) that consensus, as easy as the problem specification might seem, is in fact impossible to solve in a variety of system models in the presence of faults.

Every process starts its execution with a prescribed input value and decides upon termination on an output value. We will consider consensus only in system models with model parameter $f \geqslant 1$. Otherwise, consensus is trivially solvable. One simplification[1] that we make is that the set of possible input and output values is equal to $\{0, 1\}$. This special case of consensus is called *binary* consensus.

Formally, input and output values are modeled in the following way: First, we impose the restriction that every process has to have at least two distinct initial states. For every process p_j, let S_j denote its set of states and $I_j \subset S_j$ its set of initial states. We demand $|I_j| \geqslant 2$. Input values are modeled by a mapping $\iota_j : I_j \to \{0, 1\}$ which we demand to be non-trivial. Output values are modeled by a mapping $\delta_j : S_j \to \{0, 1, \perp\}$. We say that process p_j has *decided on* $v \in \{0, 1\}$ *in state* $s \in S_j$ if $\delta_j(s) = v$. We demand that decisions are *irrevocable*, i.e., if s is part of some configuration C, $\delta_j(s) \in \{0, 1\}$, and configuration C' follows C in some execution, then $\delta_j(s') = \delta_j(s)$ where s' is p_j's state in C'. Hence, we may extend δ_j to execution of the algorithm.

Of course, even with f crash failures, agreement on a value can be achieved trivially by programming every process to decide on 0. Hence, we limit our attention to non-trivial consensus. We say that an algorithm *solves consensus* if:

[1] In reality, this does not make the problem any simpler, just the notation. And since we are doing impossibility results, it suffices to limit ourselves to this special case.

(T) For every admissible execution holds: Every process that is correct[2] decides on some value. *(Termination)*

(A) For every admissible execution holds: No two correct processes decide on differing values. *(Agreement)*

(V) For every admissible execution holds: If the execution starts from an initial configuration in which all input values are equal to v, then all correct processes decide on v. *(Validity)*

3.2. k-Set Agreement

Consensus is 1-set agreement. In k-set agreement with $k \in \mathbb{N}$, we expand the set of possible input (and output) values to $\{1, 2, \ldots, M\}$ with $M \geqslant N$ and replace condition (A) with

(k-A) For every admissible execution holds: No $k + 1$ correct processes decide to pairwise differing values. *(k-Agreement)*

Definition 3.1. Let $\mathcal{S}$ be the set of admissible executions of a k-set agreement (or consensus) algorithm and let C be a configuration in $\mathcal{S}$. We say that C is *α-valent* if all successor configurations of C if which a decision was reached have the decision value α. In this case, we call C *univalent*, otherwise *multivalent*, or in the case $M = 2$ *bivalent*. □

[2] See descriptions in 2.2.1, 2.3.1 and 2.4.1 for details when a process is considered *correct*.

4. Point-Set Topology

In this chapter, we treat techniques from elementary point-set topology (see Appendix A for an introduction to the subject) with respect to their applicability to distributed computing. We show how to equip execution spaces with a natural topology that can be used to derive impossibility results. In particular, we re-prove FLP impossibility (Fischer, Lynch, and Paterson 1985) in this novel topological framework.

4.1. The Topology of Execution Spaces

This section introduces the necessary tools for formulating the main result of this thesis in Section 4.2. We show how to equip a space of executions of some distributed algorithm with a certain topology that helps us express executional properties in a topological manner.

But before we talk about execution *spaces*, we have to fix the term "execution" and explain what we mean by it.[1] Common to all models of distributed computing is the notion of a *configuration*, meaning a snapshot of the state of a system. That is, a configuration encompasses information about the internal state of every process and the state of the communication medium (e.g., messages in transit or contents of shared memory). An *execution* is a sequence of configurations such that each configuration in the sequence is a successor of the former ones. Notice that the meaning of these two notions is heavily model-dependent. From the topological viewpoint, we are not interested in the ontological question of what a configuration really *is*; we only need to know which successor configurations are possible. Thus, we "shift focus from the structure of protocols for a distributed system to the structure of the set of possible schedules of a distributed system." (Saks and Zaharoglou 2000)

We denote by $\mathcal{C}_{\mathcal{A}}$ the set of all configurations of algorithm $\mathcal{A}$. Let $\mathcal{S}_{\mathcal{A}}$ denote the set of admissible executions, which is a subset of the set $\mathcal{C}_{\mathcal{A}}^{\omega}$ of sequences of configurations.[2] We will equip the latter space with a natural topology that induces a topology on its subset $\mathcal{S}_{\mathcal{A}}$.

When there is no danger of ambiguity, we will write $\mathcal{C}$ and $\mathcal{S}$ for $\mathcal{C}_{\mathcal{A}}$ and $\mathcal{S}_{\mathcal{A}}$, respectively.

[1] The formal definition of these terms were given in Chapter 2.

[2] Executions, in models that we consider, are infinite *per definitionem.*

We endow $\mathcal{C}$ with the discrete topology, i.e., every subset of $\mathcal{C}$ is defined to be open. This topology is induced by the metric

$$d_D : \mathcal{C} \times \mathcal{C} \to \mathbb{R}^+, \quad d_D(C, C') = \begin{cases} 0 & \text{if } C = C' \\ 1 & \text{else.} \end{cases} \tag{4.1}$$

The natural topology to endow $\mathcal{C}^\omega = \prod_{n\in\omega} \mathcal{C}$ with is the product topology (see Section A.2.5).

Lemma 4.1. The product topology on $\mathcal{C}^\omega$ is induced by the metric

$$d\big((C_k), (C'_k)\big) = 2^{-\inf\{j \mid C_j \neq C'_j\}} \tag{4.2}$$

Proof. We have to show that the sets that are open with respect to the product topology (Definition A.14) are exactly those sets that are open with respect to the metric (Example A.3).

Let $\mathcal{A} \subset \mathcal{C}^\omega$ be open with respect to the metric d. The definition of openness with respect to the metric asserts existence of $\varepsilon(\gamma) > 0$ for every $\gamma = (C_k) \in \mathcal{A}$ such that

$$\mathcal{A} = \bigcup_{\gamma \in \mathcal{A}} B_{\varepsilon(\gamma)}(\gamma). \tag{4.3}$$

From this equation we derive that it suffices to show that $B_\varepsilon(\gamma)$ is open with respect to the product topology whenever $\varepsilon > 0$. In this case, choose the integer K minimal with the property $2^{-K} \leqslant \varepsilon$. This choice implies

$$B_\varepsilon(\gamma) = B_{2^{-K}}(\gamma) = \{\gamma' \mid \gamma \text{ and } \gamma' \text{ agree in the first } K \text{ components}\}. \tag{4.4}$$

If $\pi_m : \mathcal{C}^\omega \to \mathcal{C}$ denotes the projection onto the mth component, then the inverse image $\pi_m^{-1}\big[\{C_m\}\big]$ of the open set $\{C_m\} \subset \mathcal{C}$ is exactly the set of elements in $\mathcal{C}^\omega$ whose mth component is equal to C_m. Also, by definition, these inverse images are open with respect to the product topology. We thus conclude on the openness of $B_\varepsilon(\gamma)$ with respect to the product topology, because the latter set in (4.4) is equal to

$$\bigcap_{m=0}^{K} \pi_m^{-1}\big[\{C_m\}\big]. \tag{4.5}$$

To prove the converse direction, it suffices to show that all sets of the form $\pi_m^{-1}[O]$ where O is a subset[3] of $\mathcal{C}$ are open with respect to the metric d. But we may write

$$\pi_m^{-1}[O] = \bigcup_{\gamma \in \pi_m^{-1}[O]} B_{2^{-m}}(\gamma) \tag{4.6}$$

because both sides are equal to the set of elements in $\mathcal{C}^\omega$ whose mth element is in O. The openness of $B_{2^{-m}}(\gamma)$ with respect to the metric d now concludes the proof. ■

[3] Note that all sets $O \subset \mathcal{C}$ are open, because we equipped $\mathcal{C}$ with the discrete topology (see Example A.4(2)).

Finally, we endow $\mathcal{S} \subset \mathcal{C}^\omega$ with the subset topology (Example A.9), that is, the topology induced by the same metric.

This topology on execution spaces was introduced by Alpern and Schneider (1985). They characterized safety and liveness properties of executions in a topological way. Namely, a property is a safety property if and only if the set of executions satisfying it is closed (Definition A.8) with respect to the previously defined topology. Similarly, a property is a liveness property if and only if the set of executions satisfying it is dense (Definition A.11) with respect to this topology. They used this characterization to prove that any property is an intersection of a safety and a liveness property. Also, in non-pathological cases, every property is an intersection of two liveness properties. We will now retrace these insights.

In the following, let $\mathcal{C}$ be the set of configurations of some algorithm.

Lemma 4.2. A property $\mathcal{P} \subset \mathcal{C}^\omega$ is a safety property if and only if $\mathcal{P}$ is closed in $\mathcal{C}^\omega$.

Proof. It is equivalent to prove that $\mathcal{P}$ is a safety property if and only if its complement $\mathcal{P}^c$ is open.

Let $\mathcal{P}$ be a safety property. Then, by definition, for every $E \in \mathcal{P}^c$, there exists some index k such that for every $E' \in \mathcal{C}^\omega$ that agrees with E in the first k components, we have $E' \in \mathcal{P}^c$. By setting $\varepsilon = 2^{-k}$, we arrive at the insight that $B_\varepsilon(E) \subset \mathcal{P}^c$ and we are done.

Conversely, let $\mathcal{P}^c$ be an open set. Let $E \in \mathcal{P}^c$. We have to show that there exists some index k such that for any $E' \in \mathcal{C}^\omega$ that coincides with E in the first k components, we have $E' \in \mathcal{P}^c$. The set $\mathcal{P}^c$ being open, there exists some $\varepsilon > 0$ such that $B_\varepsilon(E) \subset \mathcal{P}^c$. Let $k \in \mathbb{N}$ such that $2^{-k} < \varepsilon$. Then the set of all E' that coincide with E in the first k components is a subset of $B_\varepsilon(E)$, which concludes the proof. ■

Lemma 4.3. A property $\mathcal{P} \subset \mathcal{C}^\omega$ is a liveness property if and only if $\mathcal{P}$ is dense in $\mathcal{C}^\omega$.

Proof. Let $\mathcal{P}$ be a liveness property and let $E \in \mathcal{C}^\omega$ and $\varepsilon > 0$. We will show that there exists some $E' \in \mathcal{P}$ such that $d(E, E') < \varepsilon$. Let $k \in \mathbb{N}$ such that $2^{-k} < \varepsilon$. By definition of liveness, there exists an extension E' of the execution fragment formed by taking the first k components of E such that $E' \in \mathcal{P}$. But then $d(E, E') < 2^{-k} < \varepsilon$ and we are done.

Conversely, let $\mathcal{P}$ be a dense set. Let $(C_0, C_1, \ldots, C_k)$ be an execution fragment. We have to show that there exists some extension $E' \in \mathcal{P}$ of the fragment. Let E be any extension of the fragment. Because $\mathcal{P}$ is dense, there exists $E' \in \mathcal{P}$ such that $d(E, E') < 2^{-k-1}$, which shows that E' is also an extension of $(C_0, C_1, \ldots, C_k)$. ■

The following result was proved by Alpern and Schneider (1985).

Theorem 4.1. Let $\mathcal{P} \subset \mathcal{C}^\omega$ be any property. Then there exists a safety property $\mathcal{S}$ and a liveness property $\mathcal{L}$ such that $\mathcal{P} = \mathcal{S} \cap \mathcal{L}$.

Proof. Define $\mathcal{S} = \overline{\mathcal{P}}$ to be the topological closure of $\mathcal{P}$ and $\mathcal{L} = \mathcal{S}^c \cup \mathcal{P}$. Then of course $\mathcal{S} \cap \mathcal{L} = \mathcal{P}$. It is also clear that $\mathcal{S}$ is closed, hence a safety property by Lemma 4.2. It remains to show that $\mathcal{L}$ is dense (Lemma 4.3).

By Lemma A.4(4), we have

$$\overline{\mathcal{L}} = \overline{\overline{\mathcal{P}}^c} \cup \overline{\mathcal{P}} \supset \overline{\mathcal{P}}^c \cup \overline{\mathcal{P}} = \mathcal{C}^\omega \tag{4.7}$$

which concludes the proof. ∎

Notice that the above notions of closedness and density, but also of safety and liveness, are dependent on the surrounding space $\mathcal{C}^\omega$. We used these notions in a "global" sense, meaning that they were understood with respect to the whole space $\mathcal{C}^\omega$. When considering a specific algorithm running in a specific model, however, we may mean something different by "safety" and "liveness".

For example, consider a consensus algorithm $\mathcal{A}$ with set $\mathcal{S} \subset \mathcal{C}^\omega$ of admissible executions. Then $\mathcal{C}_0$, the set of initial configurations of $\mathcal{A}$, is not equal to $\mathcal{C}$, because there exist configurations in which processes have decided, but no process has decided in an initial configuration. It follows that $\mathcal{S}$ is not dense in $\mathcal{C}^\omega$, since every $E \in \mathcal{S}$ starts with an initial configuration. Hence $\mathcal{S}$, as a property in $\mathcal{C}^\omega$, is not a liveness property. But $\mathcal{S}$ itself is of course a liveness (and even a safety) property in $\mathcal{S}$. It follows that the notions of safety and liveness are *relative* notions. Of course, statements analog to that of Lemma 4.2 and Lemma 4.3 hold relative to some set $\mathcal{S}$ of executions.

4.1.1. Motivation

In this subsection, we will demonstrate topological proof techniques for impossibility results with the help of a simple example.

Consider a consensus algorithm $\mathcal{A}$ for $N \geqslant 3$ processes communicating by means of single-reader multiple-writer shared read-write registers.[4] Of these processes, $f = N - 1$ might fail by crashing. According to Section 2.4, a configuration C in this model consists of a tuple $(s_1, s_2, \dots, s_N)$ and a tuple $(v_1, v_2, \dots, v_M)$ where s_i is the internal state of process p_i and v_i is the content of a shared memory register. An event in this system model is a process number $j \in \{1, 2, \dots, N\}$, which expresses that process p_j takes a step. An admissible schedule is a sequence of process numbers in which at least one process occurs infinitely often (recall that $f = N - 1$). Hence every sequence in $\{1, 2, \dots, N\}^{\mathbb{N}}$ is an admissible schedule.

We work with two sequence spaces: the space of schedules and the space of executions. In this example, the set of schedules is equal to $\Sigma = \{1, \dots, N\}^{\mathbb{N}}$ and the set $\mathcal{S}$ of admissible executions of some algorithm is a subset of $\mathcal{C}^\omega$ where $\mathcal{C}$ denotes the set of all configurations. We equip *both* sets with the topology discussed at the beginning of Section 4.1.

[4] This is very similar to the model of Dolev, Dwork, and Stockmeyer 1987, Theorem I1.1 and a special case of the system model in Section 2.4.

Lemma 4.4. Let $\mathcal{S}$ be the set of admissible executions of some consensus algorithm $\mathcal{A}$. Define the map $\Delta : \mathcal{S} \to \{0,1\}$ such that $\Delta(E)$ is the decision value of algorithm $\mathcal{A}$ in execution E. Then Δ is continuous.

Proof. It suffices to show that Δ is locally constant, i.e., for all $E \in \mathcal{S}$, there exists some neighborhood $\mathcal{N}$ of E (Definition A.6) such that Δ is constant on $\mathcal{N}$, that is, $\Delta(E') = \Delta(E)$ for all $E' \in \mathcal{N}$.

Let $E \in \mathcal{S}$ be some admissible execution of $\mathcal{A}$. By the termination property of consensus, there exists some configuration C in E such that some process has already decided. Let k be an index such that the kth configuration in E is equal to C. We claim that

$$\mathcal{N} = \big\{E' \in \mathcal{S} \mid E' \text{ coincides with } E \text{ up to the } k\text{th configuration}\big\} \tag{4.8}$$

is the desired neighborhood. It is clear that Δ is constant on $\mathcal{N}$, because by the agreement condition of consensus, no other consensus decision value is possible after a process has decided.

It remains to show that $\mathcal{N}$ is indeed a neighborhood of E. Define $\varepsilon = 2^{-k}$. With the metric defined in (4.2), we conclude that $\mathcal{N}$ is the set of admissible executions that have distance to E less than ε. Thus $\mathcal{N}$ is an ε-ball, hence open. ■

Let $\mathcal{S}$ denote the set of admissible executions. When we fix some initial configuration I, by the semantics of the model, every admissible schedule determines exactly one admissible execution. This induces a mapping $f_I : \Sigma \to \mathcal{S}$, details of which are depicted in Figure 4.1.

$$\begin{tikzcd}
 & & \text{initial config. } C_0\text{; schedule } (j_1, j_2, j_3, j_4, \dots) \arrow[d, mapsto, "f"] & & & \\
C_0 \arrow[r, bend left, "j_1"] & C_1 \arrow[r, bend left, "j_2"] & C_2 \arrow[r, bend left, "j_3"] & C_3 \arrow[r, bend left, "j_4"] & C_4 \arrow[r, bend left, "j_5"] & \cdots
\end{tikzcd}$$

Figure 4.1.: Mapping from schedules to executions

If $\mathcal{C}_0$ denotes the set of initial configurations, then $I \mapsto f_I$ is a map with domain $\mathcal{C}_0$. Hence f may be viewed[5] as a function $\mathcal{C}_0 \times \Sigma \to \mathcal{S}$.

Lemma 4.5. If we equip $\mathcal{C}_0$ with the discrete topology, then $f : \mathcal{C}_0 \times \Sigma \to \mathcal{S}$ as defined above is continuous.

Proof. Let (I, σ) be an element of $\mathcal{C}_0 \times \Sigma$ and let $(I_k, \sigma_k)_k$ be a sequence converging to (I, σ). Then $I_k \to I$ and $\sigma_k \to \sigma$. We will show $f(I_k, \sigma_k) \to f(I, \sigma)$.

[5] by uncurrying

Since $\mathcal{C}_0$ carries the discrete topology, convergence of (I_k) means that it is eventually constant (and equal to I). Hence it is no loss of generality to assume $I_k = I$ for all k.

It remains to show that $f_I(\sigma_k) \to f_I(\sigma)$. Let $\varepsilon > 0$. Choose $n \in \mathbb{N}$ such that $2^{-n} < \varepsilon$. Because the nth component of (σ_k) must be eventually constant for every n, there exists some $K \in \mathbb{N}$ such that the first n components of σ_k agree for all $k \geqslant K$. But then, by construction of $f_I(\sigma)$, also the first n components of $f_I(\sigma_k)$ agree for all $k \geqslant K$. Hence

$$d\big(f_I(\sigma), f_I(\sigma_k)\big) < 2^{-n} < \varepsilon \text{ for all } k \geqslant K \tag{4.9}$$

and we are done. ■

By definition of the system model, $f : \mathcal{C}_0 \times \Sigma \to \mathcal{S}$ is surjective. But by Tychonoff's theorem (Theorem A.3), Σ and also $\mathcal{C}_0 \times \Sigma$ are compact. Hence $\mathcal{S}$ is a continuous image of a compact space, which implies that $\mathcal{S}$ itself is compact (Lemma A.10). Since $\mathcal{C}^\omega$ is metrizable, hence Hausdorff, we conclude that $\mathcal{S}$ is closed in $\mathcal{C}^\omega$ (Lemma A.9). Thus, $\mathcal{S}$ is a safety property in $\mathcal{C}^\omega$ (Lemma 4.2).

The established continuity (both Δ and f_I are continuous) of $\Delta \circ f_I : \Sigma \to \{0,1\}$ for $I \in \mathcal{C}_0$ has the following consequence: The sets Σ_α, $\alpha \in \{0,1\}$, of schedules σ for which $\Delta(f_I(\sigma)) = \alpha$, i.e., where the algorithm decides on α in execution $f_I(\sigma)$, are closed in Σ, because they are inverse images of the closed sets $\{\alpha\} \subset \{0,1\}$ under a continuous mapping. Being closed in a compact space, the sets Σ_α are compact (Lemma A.9). But since each of these two sets is the complement of the other in Σ, they are also open. The following lemma now establishes a uniform bound K such that every execution starting from I is univalent after the Kth step. By passing to the maximum over all $I \in \mathcal{C}_0$, we get a uniform bound K not restricted to a particular initial configuration. In other words, C_K is univalent for *every* $(C_0, C_1, \dots, C_K, \dots) \in \mathcal{S}$.

Lemma 4.6 (Lebesgue)**.** Let (X,d) be a compact metric space and let $(U_\lambda)_{\lambda\in\Lambda}$ be an open covering of X, i.e., every U_λ is open and $X = \bigcup U_\lambda$. Then there exists some $\varepsilon > 0$ such that for any $x \in X$, the ball $B_\varepsilon(x)$ is contained in one of the U_λ.

Proof. For every $x \in X$ let $\delta_x > 0$ such that $B_{\delta_x}(x) \subset U_\lambda$ for some λ. The family of balls $B_{\delta_x/2}(x)$ indexed by $x \in X$ forms an open covering of X. By compactness (Definition A.13), there exist $x_1, x_2, \dots, x_m \in X$ such that

$$X = \bigcup_{j=1}^{m} B_{\delta_j/2}(x_j). \tag{4.10}$$

where $\delta_j = \delta_{x_j}$. We set $\delta = \min\{\delta_1, \delta_2, \dots, \delta_m\}$ and $\varepsilon = \delta/2$.

Let $x \in X$, then $x \in B_{\delta_j/2}(x_j)$ for some j. Hence $d(x,x_j) < \delta_j/2$. Let now $y \in B_\varepsilon(x)$, then $d(x,y) < \varepsilon \leqslant \delta_j/2$. The triangle inequality implies $d(y,x_j) < \delta_j$, hence

$$B_\varepsilon(x) \subset B_{\delta_j/2}(x) \subset B_{\delta_j}(x_j) \subset U_\lambda \tag{4.11}$$

for some λ and we are done. ■

Corollary 4.1. There exists a $K \in \mathbb{N}$ such that for every execution $(C_0, C_1, \dots) \in \mathcal{S}$, the Kth configuration, C_K, is univalent. ■

On to the non-topological part of the impossibility proof: Together with the existence of a bivalent initial configuration,[6] we have established the existence of a fork, i.e., there exists a bivalent configuration C and direct successor configurations D_0 and D_1 of C such that D_α is α-valent. We show that such a fork is impossible.[7] Then, we have proved consensus impossibility.

Let p be the process taking the step $C \to D_0$ and let q be the process taking the step $C \to D_1$. The processes p and q are distinct.

Case 1: Both p and q perform `read` operations. Since $f = N-1 \geqslant 2$, we can choose a third process r and apply the schedule $(r, r, \dots)$ to both D_0 and D_1. The resulting decision value (the value that r decides on) is in both cases the same, because the local state of r and all register values that r can read are the same in D_0 and D_1. But since D_0 is 0-valent and D_1 is 1-valent, this is a contradiction.

Case 2: p performs a `read` and q performs a `write` operation. The same trick as in Case 1 works. Choose a process other than p and other than the reader of the register that q writes to.

Case 3: Both p and q perform `write` operations. Silence the readers of both registers that get written to by p and q.

4.1.2. Execution Trees

Executions are sequences of configurations. At any point (configuration) in such a sequence, it is often possible to choose from more than one successor configuration, as governed by the system model and the algorithm. One can, in a natural way, assign a decision tree to any set of executions that captures the decision of choosing a successor. We will characterize in Section 4.1.3 certain sets of executions whose decision trees capture all the information about the original set. This follows an idea by Lubitch and Moran (1995).

Let $\mathcal{C}$ be the set of configurations of an algorithm $\mathcal{A}$ and let $\mathcal{S} \subset \mathcal{C}^\omega$. We will construct a tree $T(\mathcal{S})$ that reflects the local decisions of choosing a successor configuration. We construct it inductively, of course. First of all, we insert a root $\perp$. We then connect to it nodes labeled with every configuration C_0 that occurs as an initial configuration in $\mathcal{S}$, see Figure 4.2. These are exactly the vertices at depth 1. Suppose now that we already constructed the tree up to depth n. We describe how to construct the vertices at depth $n+1$. Let C_{n-1} be a vertex at depth n. There exists a unique path $(\perp, C_0, C_1, \dots, C_{n-1})$ from $\perp$ to C_{n-1}. We connect to C_{n-1}

[6] Existence of a bivalent initial configuration is established by a bit-flipping argument. This standard proof technique can be examined in Attiya and Welch 2004, Lemma 5.16.

[7] cf. Dolev, Dwork, and Stockmeyer 1987, Lemma I1.1.1

nodes labeled with every configuration C_n such that $(C_0, C_1, \dots, C_{n-1}, C_n)$ occurs as a prefix of some execution in $\mathcal{S}$.

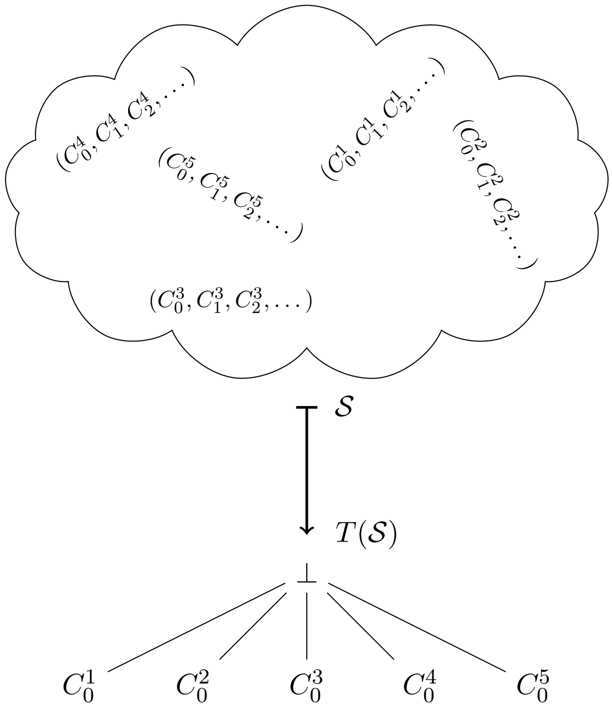

$(C_0^4, C_1^4, C_2^4, \dots)$
$(C_0^5, C_1^5, C_2^5, \dots)$
$(C_0^1, C_1^1, C_2^1, \dots)$
$(C_0^2, C_1^2, C_2^2, \dots)$
$(C_0^3, C_1^3, C_2^3, \dots)$
$\mathcal{S}$
$T(\mathcal{S})$
$\perp$
C_0^1
C_0^2
C_0^3
C_0^4
C_0^5

Figure 4.2.: Constructed tree up to depth 1

Note that this procedure indeed does result in a tree since no single vertex is connected twice to a predecessor node. In particular, vertices are *not* configurations, but only labeled by configurations. Hence it is indeed possible that there exist two different nodes in $T(\mathcal{S})$ that are labeled with the same configuration. The labels of nodes at depth n are exactly the configurations that occur in the $(n-1)$th component of executions in $\mathcal{S}$. Also, nodes in this tree may have infinite degree. But, as we will see later, for any specific distributed system model and algorithm running on it, the tree of the set of admissible executions will be locally finite, i.e., nodes will have finite degree.

By construction, every execution in $\mathcal{S}$ corresponds to an infinite path in the tree $T(\mathcal{S})$. The converse, however, is not true in general. There exist sets $\mathcal{S}$ and infinite paths in $T(\mathcal{S})$ that do not correspond to an execution in $\mathcal{S}$. For example, consider a shared memory algorithm for $N \geqslant 3$ processes and up to $f = N - 2$ crash faults. The corresponding set $\mathcal{S}$ contains all sequences of process numbers with the property that at least two processes occur infinitely often. In the tree $T(\mathcal{S})$ every node has exactly N children — one for every process. Hence the infinite path $(1, 1, \dots)$ exists in $T(\mathcal{S})$, but this sequence is not an element of $\mathcal{S}$. In the following section, we will explore for which sets $\mathcal{S}$ every path in the assigned tree describes an execution is $\mathcal{S}$.

4.1.3. Path–Sequence Duality

For a tree $T(\mathcal{S})$ as constructed in 4.1.2, let $P\big(T(\mathcal{S})\big)$ denote the set of executions in $\mathcal{C}^\omega$ that correspond to infinite paths in $T(\mathcal{S})$. The relation $P\big(T(\mathcal{S})\big) \supset \mathcal{S}$ always holds. We are interested in those $\mathcal{S}$ for which $P\big(T(\mathcal{S})\big) = \mathcal{S}$. These are the sets of executions for which it suffices to make local decisions when constructing an admissible execution. For all sets that do not have this property, we have to filter out some executions constructed by virtue of paths in order to arrive at the set of admissible executions. The following theorem solves the above question.

Theorem 4.2. Let $\mathcal{S} \subset \mathcal{C}^\omega$. The following are equivalent:

(1) $P\big(T(\mathcal{S})\big) = \mathcal{S}$

(2) $\mathcal{S}$ is closed in $\mathcal{C}^\omega$

In this case, $\mathcal{S}$ is compact if and only if $T(\mathcal{S})$ is locally finite.

Proof. (1)⇒(2): Let $E = (C_0, C_1, \dots) \notin \mathcal{S} = P\big(T(\mathcal{S})\big)$. Then either C_0 is not an initial configuration in $\mathcal{S}$ or there exists some $n \in \mathbb{N}$ such that C_n is not a child of C_{n-1} in $T(\mathcal{S})$. In the first case, no execution that starts with C_0 is an element of $\mathcal{S}$, i.e., $B_1(E) \subset \mathcal{C}^\omega \setminus \mathcal{S}$. In the second case, by definition of $T(\mathcal{S})$, we have $B_{2^{-n}}(E) \subset \mathcal{C}^\omega \setminus \mathcal{S}$. We conclude that $\mathcal{C}^\omega \setminus \mathcal{S}$ is open, hence $\mathcal{S}$ is closed.

(2)⇒(1): Let $E = (C_0, C_1, \dots) \in P\big(T(\mathcal{S})\big)$. We show $E \in \mathcal{S}$. By definition of $T(\mathcal{S})$, for any $n \in \mathbb{N}$ there exists some execution $E_n \in \mathcal{S}$ that starts with $(C_0, C_1, \dots, C_n)$. Since $E_n \to E$ as $n \to \infty$ and since $\mathcal{S}$ is closed, the claim follows from Lemma A.3.

Now, let $T(\mathcal{S})$ be locally finite, i.e., every node has finite degree. Then, for every $n \in \omega$, there are only finitely many configurations possible to appear as the nth component of an execution in $\mathcal{S}$. Denote the set of possible configurations in the nth component by $\mathcal{C}_n$. Then, $\mathcal{S} \subset \prod_{n\in\omega} \mathcal{C}_n$. The latter set being compact (Theorem A.3 and Example A.10), the compactness of $\mathcal{S}$ follows from Lemma A.9(1).

Finally, let $\mathcal{S}$ be compact. Let v be any vertex in $T(\mathcal{S})$. We will show that v has only finitely many children. Suppose that $v = \perp$ is the root and $\perp$ has infinitely many children, i.e., $\mathcal{C}_0$ is infinite. Consider the following open covering of $\mathcal{S}$:

$$B_1(C_0, C_1, \dots) = \big\{E' \in \mathcal{S} \mid E' \text{ starts with } C_0\big\} \quad \text{where } C_0 \in \mathcal{C}_0 \tag{4.12}$$

By compactness of $\mathcal{S}$, there exists a finite subcovering, but this is impossible. The case $v \neq \perp$ is completely analogous. ■

The advantage of closed sets of executions is now obvious: We can restrict ourselves to *local* decisions when constructing an execution in the set. So, if we are considering some closed subset of the set of admissible executions, we are guaranteed admissibility of the execution constructed in a local fashion.

4.2. Topological Impossibility

This section contains the core of our topological impossibility proofs. We begin with the topological main theorem:

Theorem 4.3. Let (X, d) be a metric space, $A \subset X$ closed in X and $C \subset X$ compact. If $A \cap C = \emptyset$, then

$$d(A, C) = \inf \{d(a, c) \mid a \in A, c \in C\} > 0. \tag{4.13}$$

Proof. Suppose not, i.e., $d(A, C) = 0$. Define the map $f_A : C \to [0, \infty)$ by $f_A(x) = \inf\{d(x, a) \mid a \in A\}$. If we are able to show that f_A is continuous, then we are done. Because then the continuous function f_A attains its minimum in the compact set C (Lemma A.11), i.e., there exists some $c \in C$ with $f_A(c) = 0$. But since c is an element of the open set $X \setminus A$, there exists some $\varepsilon > 0$ such that $B_\varepsilon(c) \subset X \setminus A$. This implies $f_A(c) = \inf\{d(c, a) \mid a \in A\} \geqslant \varepsilon > 0$, a contradiction.

It remains to show that f is continuous. Let $\varepsilon > 0$ and $c \in C$. We choose $\delta = \varepsilon$. For $c' \in C$ with $d(c, c') < \varepsilon$, we have

$$\left|f_A(c) - f_A(c')\right| \leqslant d(c, c') + f_A(c') - f_A(c') < \varepsilon \tag{4.14}$$

where we assumed without loss of generality that $f_A(c) \geqslant f_A(c')$ and used that $f_A(c) \leqslant d(c, c') + f_A(c')$ by the triangle inequality and that taking the infimum preserves weak inequalities. ■

If we are able to find a compact (hence closed, Lemma A.9(2)) set $\mathcal{K} \subset \mathcal{S}$ for which $d(\mathcal{K}_0, \mathcal{K}_1) = 0$ where $\mathcal{K}_\alpha$ denotes the set of α-deciding executions, we are done. We then know that the $\mathcal{K}_\alpha$ are compact, because the decision function Δ is continuous. But this is a direct contradiction to Theorem 4.3.

We will, however, use a slightly different argument to derive a contradiction. Most often, we will not directly reason with sequences of configurations. What we will rather do is follow an idea which was introduced by Lubitch and Moran (1995) and generalized by Moses and Rajsbaum (2002): We use *schedulers* to construct executions.

Definition 4.1. Let $\mathcal{S}$ be the set of admissible executions of some algorithm. A *scheduler* for $\mathcal{S}$ is a metric space X together with a continuous map $f : X \to \mathcal{S}$. A scheduler is called *closed* if X is compact. □

We will use schedulers to describe the construction of executions in a (closed) subset of $\mathcal{S}$ (namely the image of f). An example of a scheduler is the above mapping f of Section 4.1.1. We prominently used the fact that f and in particular that $\Delta \circ f$ was a continuous mapping. The general result about transportation of properties in Theorem 4.3 is the following lemma.

Lemma 4.7. Let $f : X \to \mathcal{S}$ be a closed scheduler. The following assertions are true:

(1) If $A \subset X$ is closed then $f[A] \subset \mathcal{S}$ is closed and compact.

(2) $d(A, B) = 0$ implies $d(f[A], f[B]) = 0$ for all $A, B \subset X$.

Proof. (1): Every closed set $A \subset X$ is compact since X is compact (Lemma A.9(1)). Since f is continuous, we may deduce that $f[A]$ is compact (Lemma A.10). But $\mathcal{S}$ is Hausdorff and hence every compact set is closed by Lemma A.9(2); in particular $f[A]$.

(2): Let $d(A, B) = 0$. If we set $\varepsilon_k = 2^{-k}$ we get by the uniform continuity of f (similar to Theorem A.2) the existence of $\delta_k > 0$ such that

$$d(x, y) < \delta_k \Rightarrow d(f(x), f(y)) < 2^{-k} \tag{4.15}$$

By hypothesis there exist sequences (a_k) in A and (b_k) in B such that $d(a_k, b_k) < \delta_k$ for all $k \in \mathbb{N}$. The implied relation $d\left(f(a_k), f(b_k)\right) \to 0$ as $k \to \infty$ now concludes the proof. ■

Any closed scheduler we will construct in the subsequent will, like $\mathcal{S}$, itself be a sequence space, i.e., $X \subset L^{\mathbb{N}}$ for some set L. Hence a closed scheduler X can be viewed as the set of paths in the locally finite tree $T(X)$; see Theorem 4.2. Often, the set L will be a set of *layers* (Moses and Rajsbaum 2002), i.e., each $\ell \in L$ will correspond to a finite sequence of events. For example, a layer in the shared memory model will be every process taking one step in some fixed order. The fact that each layer is a fixed finite sequence of events will immediately establish continuity of f. Hence the continuity of $\Delta \circ f$ where Δ is the decision function will establish the compactness of both X_0 and X_1 where X_α is the set of schedules in which the algorithm decides on α. It remains to show $d(X_0, X_1) = 0$ to complete the impossibility proof by virtue of Theorem 4.3 — see Section 4.2.1.

In a way, a scheduler $f : X \to \mathcal{S}$ defines a sub-model of $\mathcal{S}$. This is particular apparent if we consider schedulers that consist of sequences of layers: In $\mathcal{S}$, it is defined which configurations may follow which. The layering limits these possibilities and takes "shortcuts" from one configuration to another. Thus, the tree $T(X)$ can be seen as a sub-tree of $T(\mathcal{S})$ in some sense with the additional convenient property that $P\big(T(X)\big) = X$ (if we are considering a closed scheduler).

4.2.1. Additional Structure — Configuration Similarity

Up to now, we solely considered a single structural entity regarding the set of executions: In which order configurations may occur in an execution. But there is more information to configurations than their order. To be precise, we now introduce two similarity relations on the set of configurations; one model-dependent (process similarity) and one model-independent (valence similarity).

Definition 4.2. Let C and C' be configurations. We write $C \sim_v C'$ if it is not the case that one is 0-valent and the other is 1-valent. Then C and C' are called *valence similar.* □

Definition 4.3. Let C and C' be configurations of a message-passing algorithm, i.e., an algorithm in either of the models of Sections 2.2 or 2.3. We write $C \sim_p C'$ and call C and C' *process similar* if C and C' differ in the state of at most one process. We denote the transitive closure of this relation by the same name and symbol. □

Definition 4.4. Let C and C' be configurations of a single-writer shared-memory algorithm (Section 2.4). We write $C \sim_p C'$ and call C and C' *process similar* if C and C' differ in the state and registers[8] of at most one process. We denote the transitive closure of this relation by the same name and symbol. □

The following lemmata provide a relation between these two similarity notions and an argument why $d(X_0, X_1) > 0$, i.e., a 0-1-fork, is often not possible in closed sets.

Lemma 4.8. Consider either a message-passing or a single-writer shared memory model with at most $f \geqslant 1$ crash faults. Let $\mathcal{S}$ be a closed set of admissible executions that

(1) has the possibility to *silence* a process, i.e., from every configuration C and for every process p exists an infinite path in $T(\mathcal{S})$ starting from C in which p does not take steps.

(2) is *locally uniform*, i.e., if a sequence of events that does not involve process q is applicable to a configuration C in $\mathcal{S}$, then it is applicable to all configurations in $\mathcal{S}$ that differ from C at most in the state (and registers) of q.

Let C and C' be configurations in $\mathcal{S}$. If there exist successors D and D' of C and C' respectively such that $D \sim_p D'$, then $C \sim_v C'$

Proof. Suppose not, i.e., without loss of generality C is 0-valent and C' is 1-valent. Since C is 0-valent, so is its successor D. By the same token, D' is 1-valent. The relation $D \sim_p D'$ implies the existence of processes $q_1, q_2, \ldots, q_t$ and configurations $D_0, D_1, \ldots, D_t$ such that $D_0 = D$, $D_t = D'$, and configurations D_{j-1} and D_j differ exactly in the state of process q_j. By transitivity of valence similarity, it suffices to show that if D and D' differ only in the state of a single process q, then $D \sim_v D'$.

Because we assumed that we can silence process q, there exists a sequence of events starting from both D and D' that does not include steps taken by q. By the event semantics of the particular models, the processes other than q decide on the same values in both executions, hence the valency of D and D' is the same. This is a contradiction. ■

Lemma 4.9. Let C be a bivalent configuration. Then not all successor configurations of C are valency similar. ■

[8] We say that register R belongs to process p if p is the sole writer of R.

4.3. Impossibility Results

We will list a number of impossibility proofs in this section that utilize our topological framework. In particular, we give a closed scheduler for each of the models. We will use arguments from Moses and Rajsbaum 2002, Sections 3, 7 and 8.

4.3.1. Asynchronous Message Passing

We now present a topological proof of the consensus impossibility result (Fischer, Lynch, and Paterson 1985) in the model introduced in Section 2.2.

We fix some enumeration $p_1, p_2, \dots, p_N$ of the processes. The scheduler we will use is the following. We choose

$$L_\pi = \{\mathrm{aok}_\pi, \mathrm{except}_\pi\} \cup \{\mathrm{delayed}_\pi(i) \mid 1 \leqslant i \leqslant N\} \tag{4.16}$$

for any permutation[9] $\pi \in S_N$, $L = \bigcup_{\pi \in S_N} L_\pi$ as the set of layers, and $X = L^{\mathbb{N}}$. The mapping $f : X \to \mathcal{S}$ is defined to be the application of a sequence of layers to these initial configurations. More precisely, every layer in L is defined to be a finite sequence of events (see below for the exact definition) and the application of a sequence $(\ell_1, \ell_2, \ell_3, \dots)$ of layers to an initial configuration C_0 is defined by concatenating all layers to a single schedule σ and taking the corresponding sequence of steps.

It remains to describe the layers aok_π, except_π and $\mathrm{delayed}_\pi(i)$. In layer aok_π, every process takes steps in the order governed by the permutation π, i.e., in the order $p_{\pi(1)}, p_{\pi(2)}, \dots, p_{\pi(N)}$, each process receives all messages sent to it thus far. Layer except_π is the same as aok_π except that process $p_{\pi(N)}$ does not take steps. In layer $\mathrm{delayed}_\pi(i)$, processes take steps in the order of π and all messages are received *except* for messages sent from $p_{\pi(i)}$ to $p_{\pi(i)+1 \bmod N}$ in this very layer. (They are likely to be received in the next layer — namely if and only if the recipient takes a step in the next layer.)

We equip the finite set L with the discrete topology and $X = L^{\mathbb{N}}$ with the product topology. Finite sets are always compact and by Tychonoff's theorem (Theorem A.3), also X is compact. The mapping f is continuous, because the nth component of $f(x)$ only depends on the first n components of $x \in X$.[10] Hence we are dealing with a closed scheduler.

Let C be a configuration. For any layer $\ell \in L$, let $C \cdot \ell$ denote the configuration that arises when applying ℓ to C. We want to show that the precondition of Lemma 4.8 is fulfilled. The set $f[X]$ of executions is closed and has the ability to silence and is locally uniform. Let $D = C \cdot \ell$ and $D' = C \cdot \ell'$ be children of C in the tree $T(X)$. We want to show that $D \sim_v D'$ with Lemma 4.8. In a first step, we restrict ourselves to the case $\ell, \ell' \in L_\pi$ for some $\pi \in S_N$. This sub-claim will follow if we

[9] We denote the set of all permutations of $\{1, 2, \dots, N\}$ by S_N.

[10] cf. Lemma 4.5

show $C \cdot \ell \sim_v C \cdot \mathrm{aok}_\pi$ for all $\ell \in L_\pi$. The non-trivial cases are $\ell = \mathrm{except}_\pi$ and $\ell = \mathrm{delayed}_\pi(i)$. In the first case, we note that $C \cdot \mathrm{except}_\pi \cdot \mathrm{aok}_{(\pi\circ\sigma)} = C \cdot \mathrm{aok}_\pi \cdot \mathrm{except}_\pi$ where σ is the permutation with $\sigma(j) \equiv j-1 \mod N$. This is because in both cases the order in which processes take steps is equal to

$$\pi(1), \dots, \pi(N-1), \pi(N), \pi(1), \dots, \pi(N-1) \tag{4.17}$$

with the same sequence of events (receive all messages). The second case is $\ell = \mathrm{delayed}_\pi(i)$. There, we have $C \cdot \mathrm{delayed}_\pi(i) \sim_p C \cdot \mathrm{aok}_\pi$ because these two configurations only differ in the state of a single process (the process to which not all messages were delivered — namely $\pi(i)+1 \mod N$). This concludes the proof of the precondition of Lemma 4.8 if we restrict our choices to a single L_π.

Let now $\pi, \sigma \in S_N$. We want to show $C \cdot \mathrm{aok}_\pi \sim_v C \cdot \mathrm{aok}_\sigma$. For this, write $\pi^{-1} \circ \sigma = \tau_1 \circ \tau_2 \circ \cdots \circ \tau_k$ where every τ_j is a *transposition*, i.e., every τ_j flips the positions of two neighboring elements of $\{1, 2, \dots, N\}$.[11] An elementary result of group theory is the possibility to write a permutation as a product of transpositions. We see that it suffices to show $C \cdot \mathrm{aok}_\pi \sim_v C \cdot \mathrm{aok}_{(\pi\circ\tau)}$ for transpositions $\tau \in S_N$. But this follows from

$$C \cdot \mathrm{aok}_\pi \sim_v C \cdot \mathrm{delayed}_\pi(j) = C \cdot \mathrm{delayed}_{(\pi\circ\tau)}(j) \sim_v C \cdot \mathrm{aok}_{(\pi\circ\tau)} \tag{4.18}$$

if τ flips j and $j+1 \mod N$.

The rest of the proof is as follows: Prove the existence of a bivalent initial configuration by classical means (e.g., a bit-flipping argument; note that we can completely silence a process from an execution using our layers), use Theorem 4.3 to arrive at $d(X_0, X_1) > 0$, deduce the existence of a 0-1-fork and finally use Lemmata 4.8 and 4.9 to derive a contradiction.

Notice that we proved the following result:

Theorem 4.4. There is no consensus algorithm in the asynchronous message-passing model with at most one crash failure. ■

4.3.2. Asynchronous Shared Memory

In this section, we will prove impossibility of consensus in asynchronous system models of Section 2.4 with shared single-writer read-write registers where one process may fail by crashing (e.g., Fich and Ruppert 2003, Section 5.2).

We again fix some enumeration $p_1, p_2, \dots, p_N$ of the processes. Again, we choose for every permutation $\pi \in S_N$ a set $L_\pi = \{\mathrm{aok}_\pi, \mathrm{except}_\pi\}$ of layers and set $L = \bigcup_{\pi\in S_N} L_\pi$. The so-defined scheduler $f : L^{\mathbb{N}} \to \mathcal{S}$ is closed (see previous section for details). In layer aok_π, all processes take a step in the order $\pi(1), \pi(2), \dots, \pi(N)$. In layer except_π, all processes except for $p_{\pi(N)}$ take a step in order $\pi(1), \pi(2), \dots, \pi(N-$

[11] To be more precise, a permutation $\tau \in S_N$ is called a transposition if there exist $1 \leqslant i, j \leqslant N$ such that $i+1 \equiv j \mod N$, $\pi|(\{1, \dots, N\} \setminus \{i, j\})$ is the identity and $\pi|\{i, j\}$ is not the identity.

1). Let C be any configuration. It is $C \cdot \mathrm{aok}_\pi \cdot \mathrm{except}_\pi = C \cdot \mathrm{except}_\pi \cdot \mathrm{aok}_{(\pi\circ\sigma)}$ where $\sigma(j) \equiv j-1 \mod N$ which shows that the hypothesis of Lemma 4.8 holds if we restrict our choices to a single L_π. As above, we are done if we show $C \cdot \mathrm{aok}_\pi \sim_v C \cdot \mathrm{aok}_{(\pi\circ\tau)}$ for every transposition τ. But this follows from the fact that we consider single-writer registers by the following case distinction. Let τ flip indices i and $i+1$.

Case 1: Processes $p_{\pi(i)}$ and $p_{\pi(i+1)}$ both perform a `read` operation or both perform a `write` operation. Then the resulting configurations are equal.

Case 2: Process $p_{\pi(i)}$ performs a `read` and $p_{\pi(i+1)}$ performs a `write`. Then the resulting configurations differ in at most the state of $p_{\pi(i)}$.

Case 3: Process $p_{\pi(i)}$ performs a `write` and $p_{\pi(i+1)}$ performs a `read`. Then the resulting configurations differ in at most the state of $p_{\pi(i+1)}$.

The rest of the impossibility proof follows as above: Existence of a bivalent initial configuration by bit-flipping (since we can completely silence a process), existence of a fork by Theorem 4.3 and contradiction by Lemma 4.8.

Theorem 4.5. There is no consensus algorithm in the asynchronous single-writer shared memory model with at most one crash failure. ■

4.3.3. Transient Message Loss

We will prove the impossibility result of Santoro and Widmayer 1989, Section 4.1 in the model of Section 2.3.

Let $p_1, p_2, \ldots, p_N$ be an enumeration of the set of processes. We define the set of layers to be

$$L = \{\mathrm{loss}(i,j) \mid 1 \leqslant i \leqslant N, 0 \leqslant j \leqslant N\} \tag{4.19}$$

and again describe the scheduler $f : L^{\mathbb{N}} \to \mathcal{S}$ by describing each of the layers. As above, this scheduler is closed. The layer $\mathrm{loss}(i,j)$ is equal to the single event (set of omissions)

$$O = \{(i,k) \mid k \leqslant j \text{ and } k \neq i\}\,. \tag{4.20}$$

For every configuration, we have $C \cdot \mathrm{loss}(i,j) \sim_p C \cdot \mathrm{loss}(i,j-1)$ and $\mathrm{loss}(i,0) = \mathrm{loss}(i',0)$ which implies that the precondition of Lemma 4.8 holds. Note that we can silence any process p_i from any time on by repeatedly issuing $\mathrm{loss}(i,N)$. By the usual method, this concludes the impossibility proof.

Theorem 4.6. There is no consensus algorithm in the synchronous message-passing model with at most $N-1$ per-round message omissions. ■

5. Algebraic Topology

This chapter deals with a different view on topology than that we took in Chapter 4. We will consider *algebraic topology* (Hatcher 2002). In this discipline, we assign to topological spaces certain algebraic objects, reason about relations between these algebraic objects and then translate back these insights to statements about topological spaces. These techniques will enable us to prove the impossibility of k-set agreement.

5.1. Introduction

Algebraic topology splits up into two major threads: homotopy and homology. An example of a construction used in homotopy is the *fundamental group* of a (path-connected) topological space X. It is defined as the quotient of the group of all loops, i.e., continuous maps $[0,1] \to X$ starting from and ending at the same point $x_0 \in X$ where the group operation is defined as the juxtaposition of two loops, with respect to the equivalence relation of homotopy, i.e., continuous deformability of one loop to another. A very natural question to ask is which topological spaces have trivial fundamental groups, i.e., in which spaces are all loops continuously deformable into each other. The class of these spaces is called the class of *simply connected* spaces.

The other major branch of algebraic topology is homology. Similar to homotopy, it deals with spaces of continuous mappings $[0,1]^q \to X$, but unlike homotopy, it does not directly define a group operation on this set, but rather factors the free Abelian group generated by these mappings with respect to a certain equivalence relation. We will need a few techniques from homology in the course of this chapter and we provide a brief introduction to this topic in Section 5.2.

Turning to the world of distributed computing again, what we will do in this chapter is proving impossibility of k-set agreement in asynchronous systems communicating by read-write registers in the presence of up to k crash failures. The proof that we present here was developed by Herlihy and Shavit (1993). Its strategy is to introduce a structure on the set of local processor states of an algorithm, namely that of a *simplicial complex* and reason that the subcomplex of final configurations (configurations in which enough processes have decided) is incompatible with the so-called output complex, i.e., a simplical complex describing decisions that are allowed by the problem statement.

5.2. Homology

This section introduces basic notions of homology theory.

5.2.1. Chain Complexes

In this section, we will discuss the basic algebraic objects we will encounter along the way. These are *chain complexes* and more generally *graded Abelian groups.*

Let $A_* = (A_k)_{k\in\mathbb{Z}}$ be a sequence of Abelian groups. Then we call A_* a *graded Abelian group.* A *morphism φ of degree $m \in \mathbb{Z}$* from A_* to B_* is a sequence $\varphi_k : A_k \to B_{k+m}$ of Abelian group morphisms. We denote a morphism of degree $m = 0$ plainly by the name *morphism.* That is, a graded Abelian group is just an enumerable collection of Abelian groups and a morphism is just an enumerable collection of Abelian group morphisms.

A *chain complex* (A_*, ∂) is a graded Abelian group A_* together with a morphism ∂ of degree -1 from A_* to itself, i.e., for every $k \in \mathbb{Z}$ we have that $\partial_k : A_k \to A_{k-1}$ is an Abelian group morphism, with the additional property that $\partial_k \circ \partial_{k+1} = 0$ for all $k \in \mathbb{Z}$. This restriction is the same as saying that the image $\operatorname{im}\partial_{k+1}$ is a subset of the kernel[1] $\ker\partial_k$. We call ∂ the *boundary operator* of the chain complex.

A morphism $\varphi : C \to D$ of graded Abelian groups between chain complexes is a morphism of chain complexes if and only if for all $k \in \mathbb{Z}$ it holds that $\varphi_{k-1} \circ \partial_k^C = \partial_k^D \circ \varphi_k$. In other words, the diagram in Figure 5.1 commutes.

$$\begin{tikzcd}
\cdots & C_{k-2} \arrow[l, "\partial^C_{k-2}"'] \arrow[d, "\varphi_{k-2}"'] & C_{k-1} \arrow[l, "\partial^C_{k-1}"'] \arrow[d, "\varphi_{k-1}"'] & C_k \arrow[l, "\partial^C_{k}"'] \arrow[d, "\varphi_{k}"'] & C_{k+1} \arrow[l, "\partial^C_{k+1}"'] \arrow[d, "\varphi_{k+1}"'] & C_{k+2} \arrow[l, "\partial^C_{k+2}"'] \arrow[d, "\varphi_{k+2}"'] & \cdots \arrow[l, "\partial^C_{k+3}"'] \\
\cdots & D_{k-2} \arrow[l, "\partial^D_{k-2}"'] & D_{k-1} \arrow[l, "\partial^D_{k-1}"'] & D_k \arrow[l, "\partial^D_{k}"'] & D_{k+1} \arrow[l, "\partial^D_{k+1}"'] & D_{k+2} \arrow[l, "\partial^D_{k+2}"'] & \cdots \arrow[l, "\partial^D_{k+3}"']
\end{tikzcd}$$

Figure 5.1.: Commutative diagram for chain complex morphisms

5.2.2. The Homology Functor

To every chain complex C, we may assign a special graded Abelian group $H_*(C)$ called the *homology* of C. It has very interesting properties and is especially interesting when putting topological spaces into the mix as is done in Section 5.4.1.

Let C be a chain complex. We have already noted that $\operatorname{im}\partial_{q+1}$ is contained in $\ker\partial_q$ for every $q \in \mathbb{Z}$. These two sets being Abelian groups, we may form the

[1] The *kernel* of an Abelian group morphism $f : A \to B$ is defined to be the set of all $a \in A$ with $f(a) = 0$ and is denoted by $\ker f$. It is $\ker f$ an Abelian subgroup of A.

quotient $H_q = \ker \partial_q / \operatorname{im} \partial_{q+1}$ and H_q is again an Abelian group. Hence, $H(C) = (H_q)_{q \in \mathbb{Z}}$ is a graded Abelian group which we call the homology of C.

Note that if $H_q = 0$ for all $q \in \mathbb{Z}$, then $\ker \partial_q = \operatorname{im} \partial_{q+1}$. Hence the homology of a chain complex measures how far the following diagram is from being exact[2] at C_q:

$$\cdots \xleftarrow{\partial_{q-1}} C_{q-1} \xleftarrow{\partial_q} C_q \xleftarrow{\partial_{q+1}} C_{q+1} \xleftarrow{\partial_{q+2}} \cdots \tag{5.1}$$

Let $\varphi : C \to D$ be a chain complex morphism. Because of the defining relation for chain complex morphisms, we may deduce that $\varphi_q[\operatorname{im} \partial^C_{q+1}] \subset \operatorname{im} \partial^D_{q+1}$ and $\varphi_q[\ker \partial^C_q] \subset \ker \partial^D_q$. But this implies that φ_q factors to an Abelian group morphism $\ker \partial^C_q / \operatorname{im} \partial^C_{q+1} \to \ker \partial^D_q / \operatorname{im} \partial^D_{q+1}$. We write φ_* for the resulting morphism $H(C) \to H(D)$. This construction has the property that $(\psi \circ \varphi)_* = \psi_* \circ \varphi_*$ and $(\mathrm{id}_C)_* = \mathrm{id}_{H(C)}$ which indeed qualifies it for the name *functor*.

Let $\psi : C \to D$ be another chain complex morphism. A *chain homotopy from φ to ψ* is a graded Abelian group morphism $h : C \to D$ of degree 1 such that

$$\psi - \varphi = \partial^D \circ h + h \circ \partial^C. \tag{5.2}$$

In this case, we write $\varphi \simeq \psi$ and it is $\varphi_* = \psi_* : H(C) \to H(D)$.

5.3. Simplicial Complexes

A *simplicial complex* $\mathcal{C}$ is a set of sets with the following property:

(LC) If $A \in \mathcal{C}$ and $B \subset A$, then $B \in \mathcal{C}$.

In other words, a simplicial complex is left-closed with respect to the set inclusion relation. The elements of $\mathcal{C}$ are called *simplices* and the elements of the set $\bigcup \mathcal{C}$ are called *vertices*. For a simplex $S \in \mathcal{C}$, we define its *dimension* $\dim S = |S| - 1$. We set $\dim \mathcal{C} = \sup_{S \in \mathcal{C}} \dim S$.

A *vertex map* between two simplicial complexes $\mathcal{C}$ and $\mathcal{D}$ is a map $f : \bigcup \mathcal{C} \to \bigcup \mathcal{D}$. It is called *simplicial* if for every $S \in \mathcal{C}$, $f[S] \in \mathcal{D}$, i.e., every simplex in $\mathcal{C}$ gets mapped to a simplex in $\mathcal{D}$. Simplicial complexes together with simplicial vertex maps form a category.

5.3.1. Simplicial Homology

In this section, we will assign a chain complex $C(\mathcal{C})$ and also its homology $H(\mathcal{C})$ to any simplicial complex $\mathcal{C}$.

[2] A sequence $A \xrightarrow{f} B \xrightarrow{g} C$ where A, B, C are Abelian groups and $f : A \to B$, $g : B \to C$ are Abelian group morphisms is called *exact* if $\operatorname{im} f = \ker g$.

Definition 5.1. Let S be a set. We define the *free Abelian group generated by S* to be the group $\mathbb{Z}^S$. □

Let $\mathcal{C}$ be a simplicial complex and fix any total order on its set of vertices. For any $q \geqslant 0$ let $\mathcal{C}_q$ denote the set of q-dimensional simplices in $\mathcal{C}$ and let C_q be the free Abelian group generated by $\mathcal{C}_q$. The family $(C_q)_{q\in\mathbb{Z}}$ is a graded Abelian group. We will now define a boundary operator $\partial_q : C_q \to C_{q-1}$. Let $S = \{v_0, v_1, \ldots, v_q\} \in \mathcal{C}_q$ with $v_i < v_j$ for $i < j$. It suffices to define ∂_q for such elements by the universal property of free Abelian groups.[3] Set

$$\partial_q(S) = \sum_{k=0}^{q}(-1)^k \{v_0, v_1, \ldots, v_{k-1}, v_{k+1}, \ldots, v_q\}. \tag{5.3}$$

Lemma 5.1. With the above definition, $\partial_q \circ \partial_{q+1} = 0$.

Proof. Let $S = \{v_0, v_1, \ldots, v_{q+1}\} \in \mathcal{C}_{q+1}$ with $v_i < v_j$ for $i < j$. Then

$$\begin{aligned}
\partial_q(\partial_{q+1}(S)) &= \partial_q\left(\sum_{k=0}^{q+1}(-1)^k(S \setminus \{v_k\})\right) = \sum_{k=0}^{q+1}(-1)^k\partial_q(S \setminus \{v_k\}) \\
&= \sum_{k=0}^{q+1}(-1)^k\left(\sum_{\ell=0}^{k-1}(-1)^\ell(S \setminus \{v_\ell, v_k\}) + \sum_{\ell=k}^{q}(-1)^\ell(S \setminus \{v_k, v_{\ell+1}\})\right) \\
&= \sum_{k=0}^{q+1}(-1)^k\left(\sum_{\ell=0}^{k-1}(-1)^\ell(S \setminus \{v_\ell, v_k\}) + \sum_{\ell=k+1}^{q+1}(-1)^{\ell-1}(S \setminus \{v_k, v_\ell\})\right) \\
&= \sum_{0\leqslant\ell<k\leqslant q+1}(-1)^{k+\ell}(S \setminus \{v_\ell, v_k\}) - \sum_{0\leqslant k<\ell\leqslant q+1}(-1)^{k+\ell}(S \setminus \{v_k, v_\ell\}) \\
&= 0
\end{aligned} \tag{5.4}$$

as claimed. ■

It is clear that a different choice of the total order on $\bigcup\mathcal{C}$ yields an isomorphic chain complex. This concludes the definition of the chain complex $C(\mathcal{C}) = (C_*, \partial)$. We set $H(\mathcal{C}) = H(C(\mathcal{C}))$.

Let $f : \bigcup\mathcal{C} \to \bigcup\mathcal{D}$ be a simplicial vertex map. Define for every $q \geqslant 0$ the map $f_\# : C_q(\mathcal{C}) \to C_q(\mathcal{D})$ for $S \in \mathcal{C}_q$, by

$$f_\#(S) = \begin{cases} f[S] & \text{if } \dim f[S] = q \\ 0 & \text{else.} \end{cases} \tag{5.5}$$

The so-defined map $f_\# : C(\mathcal{C}) \to C(\mathcal{D})$ is a chain map and we have $(g \circ f)_\# = g_\# \circ f_\#$ and $\left(\mathrm{id}_{\bigcup\mathcal{C}}\right)_\# = \mathrm{id}_{C(\mathcal{C})}$.

[3] Let G be the free Abelian group generated by some set S, let H be an Abelian group and let $f : S \to H$ be a map. Then there exists a unique morphism $\varphi : G \to H$ such that $\varphi|S = f$.

Definition 5.2. Let $\mathcal{C}$ be a simplicial complex and let $k \geqslant 0$. We say that $\mathcal{C}$ is *k-acyclic* if $H_0(\mathcal{C}) \cong \mathbb{Z}$ and $H_q(\mathcal{C}) = 0$ for $1 \leqslant q \leqslant k$. □

Let $\mathcal{C}$ and $\mathcal{D}$ be simplicial complexes and let $\Sigma : \mathcal{C} \to P(\mathcal{D})$ be a mapping with the following properties:

(1) $\Sigma(S)$ is a simplicial complex for every $S \in \mathcal{S}$.

(2) $\Sigma(S) \subset \Sigma(S')$ if $S \subset S'$.

(3) $\Sigma(S)$ is $(q-1)$-acyclic for every $S \in \mathcal{S}$ with $\dim S = q$.

Then we call Σ an *acyclic carrier.* Let $\varphi : C(\mathcal{C}) \to C(\mathcal{D})$ be a chain map. We say that φ is *carried* by Σ if $T \in \Sigma(S)$ for all $S \in \mathcal{C}$ and $T \in \mathcal{D}$ with $c_T \neq 0$ where $\varphi(S) = \sum_{T \in \mathcal{D}} c_T T$. The following is Herlihy and Rajsbaum 2000, Theorem 3.3:

Theorem 5.1. Let $\Sigma : \mathcal{C} \to P(\mathcal{D})$ be an acyclic carrier.

(1) There exists a chain map $C(\mathcal{C}) \to C(\mathcal{D})$ that is carried by Σ.

(2) If $\varphi, \psi : C(\mathcal{C}) \to C(\mathcal{D})$ are both carried by Σ and $\dim S = \dim \Sigma(S)$ for all $S \in \mathcal{C}$, then $\varphi = \psi$. □

5.4. Algebraic vs. Combinatorial Topology

This section explains the commonalities of pure algebraic topology, i.e., the investigation of topological spaces with help of assigned algebraic structures, and combinatorial topology, i.e., the investigation of combinatorial structures with help of assigned algebraic structures. To be more precise, we introduce the homology of a topological space and the geometric realization of a simplicial complex and show that these two constructions are compatible. Results from this section are not needed later on and are presented to deepen the reader's understanding of these interconnections.

5.4.1. Singular Homology

We will now show how to relate the algebraic construction of homology to topological spaces. More precisely, we will assign to every topological space X a chain complex $C(X)$ and a graded Abelian group $H(X)$ called its *singular homology.*

Definition 5.3. Let $q \geqslant 0$. The topological space

$$\Delta^q = \left\{ x \in \mathbb{R}^{q+1} \mid \sum x_j = 1 \text{ and } x_j \geqslant 0 \text{ for all } j \right\} \tag{5.6}$$

is called the *q-dimensional standard simplex.* □

The qth component of the graded Abelian group of $C(X)$ is defined to be the free Abelian group generated by the set of continuous mappings $\Delta^q \to X$. It remains to define the boundary operator $\partial_q : C_q(X) \to C_{q-1}(X)$. It suffices to define this map on the generators of $C_q(X)$. So let $\sigma : \Delta^q \to X$ be continuous. Consider the following continuous functions $\delta_q^i : \Delta^{q-1} \to \Delta^q$ for $1 \leqslant i \leqslant q+1$:

$$\delta_q^i(x_1, \ldots, x_q) = (x_1, \ldots, x_{i-1}, 0, x_i, x_{i+1}, \ldots, x_q) \tag{5.7}$$

It is easy to see that this function really has values in Δ^q. We now define $\partial_q(\sigma)$ by the equation

$$\partial_q(\sigma) = \sum_{i=1}^{q+1} (-1)^{i-1} \left(\sigma \circ \delta_q^i\right). \tag{5.8}$$

Lemma 5.2. For any topological space X and any $q \in \mathbb{Z}$, with the above definition of $C(X)$, it holds that $\partial_q \circ \partial_{q+1} = 0$. □

Hence $C(X)$ is really a chain complex. We may thus form its homology $H(C(X))$ or just $H(X)$ in short.

Let $f : X \to Y$ be a continuous mapping. We define the mapping $f_\# : C(X) \to C(Y)$ by setting $f_\#(\sigma) = f \circ \sigma$ for continuous $\sigma : \Delta^q \to X$. It holds that $(g \circ f)_\# = g_\# \circ f_\#$ and $(\mathrm{id}_X)_\# = \mathrm{id}_{C(X)}$. We denote by f_* the mapping $H(X) \to H(Y)$ induced by the chain complex morphism $f_\#$.

Definition 5.4. Let X be a topological space and let $k \geqslant 0$. We say that X is *k-acyclic* if $H_0(X) \cong \mathbb{Z}$ and $H_q(X) = 0$ for $1 \leqslant q \leqslant k$. □

5.4.2. Geometric Realization of Simplicial Complexes

In this section, we assign a topological space $|\mathcal{C}|$ to every simplicial complex $\mathcal{C}$, called its *geometric realization.*

Let $\mathcal{C}$ be a simplicial complex. As a set, we define $|\mathcal{C}|$ by

$$|\mathcal{C}| = \left\{ \alpha : \bigcup \mathcal{C} \to [0,1] \;\middle|\; \sum_{v \in \bigcup \mathcal{C}} \alpha(v) = 1 \text{ and } \{v \mid \alpha(v) \neq 0\} \in \mathcal{C} \right\} \tag{5.9}$$

and we define the topology on $|\mathcal{C}|$ by the metric

$$d(\alpha, \beta) = \sqrt{\sum_{v \in \bigcup \mathcal{C}} (\alpha(v) - \beta(v))^2}. \tag{5.10}$$

For a simplicial vertex map $f : \bigcup \mathcal{C} \to \bigcup \mathcal{D}$, we define $|f| : |\mathcal{C}| \to |\mathcal{D}|$ by

$$|f|(\alpha)(w) = \sum_{f(v)=w} \alpha(v). \tag{5.11}$$

Consequences of these definitions are (Spanier 1966, Sec. 3.1):

(1) $|f|$ is continuous.

(2) $|g \circ f| = |g| \circ |f|$ and $\left|\mathrm{id}_{\bigcup \mathcal{C}}\right| = \mathrm{id}_{|\mathcal{C}|}$

(3) If $\mathcal{C}$ is finite then $|\mathcal{C}|$ is a compact Hausdorff space.

5.4.3. Equivalence

The following theorem relates the homologies of $\mathcal{C}$ and $|\mathcal{C}|$ (Spanier 1966, Sec. 4.6, Theorem 8).

Theorem 5.2. Let $\mathcal{C}$ be a simplicial complex. Then $H(\mathcal{C}) \cong H(|\mathcal{C}|)$. □

5.5. Configuration Complexes

This section introduces simplicial complexes that we will assign to algorithms in order to reason about the topological structure of these algorithms. In these complexes, vertices will represent a state of a single process and a simplex consisting of vertices $s_1, s_2, \ldots, s_N$ will represent a reachable configuration in which the ith process has local state s_i.

5.5.1. Input Complexes

k-set agreement is a decision task, i.e., every process has an input value and computes an output value. Simplicial complexes provide a convenient way to describe the structure of possible combinations of input and output values.

For example, consider the complex of initial configurations of the t-resilient binary consensus problem. Here, every process p_j starts with a private input value $x_j \in \{0,1\}$. For simplicity, we assume that every process has only two distinct initial states: one with input value $x_j = 0$ and one with input value $x_j = 1$. Denote by (j, α) the initial state of process p_j with input value $x_j = \alpha$. Then the set of vertices is equal to

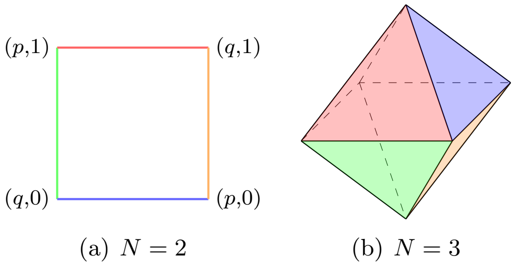
$$\{(1,0),(1,1),(2,0),(2,1),\ldots,(N,0),(N,1)\}\,. \tag{5.12}$$

Basic simplices are sets of the form

$$S = \{(1,\alpha_1),(2,\alpha_2),\ldots,(N,\alpha_N)\} \tag{5.13}$$

where $\alpha_j \in \{0,1\}$. The *input complex* for the binary consensus problem is defined by the set of these basic simplices and its subsets. A geometric realization of the input complex for $N = 2$ and $N = 3$ is depicted in Figure 5.2.

The input complex of k-set agreement is similar to that of consensus except that the set in which the input values x_j may vary is changed from $\{0,1\}$ to $\{1,2,\ldots,M\}$ where $M \geqslant N$.



(a) $N = 2$ (b) $N = 3$

Figure 5.2.: Input Complexes

5.5.2. Output Complexes

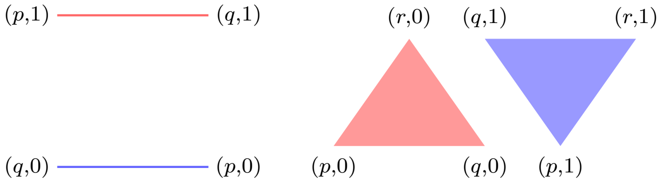
We may also look at the output complex of the t-resilient binary consensus problem, i.e., the complex that describes the possible output values y_j. Vertices here are also of the form (j, α) which reflects the fact that process p_j has decided to output value $y_j = \alpha$. *Basic* simplices are either of the form

$$S = \{(j_1, 0), (j_2, 0), (j_3, 0), \dots, (j_r, 0)\} \tag{5.14}$$

or of the form

$$S = \{(j_1, 1), (j_2, 1), (j_3, 1), \dots, (j_r, 1)\} \tag{5.15}$$

where all j_ℓ are distinct and $r \geqslant N - t$, that is, at least $N - t$ processes have decided and they all decided to the same value. The *output complex* for the t-resilient binary consensus problem is defined by the set of these basic simplices and its subsets. Examples for $N = 2$ and $N = 3$ are depicted in Figure 5.3.



(a) $N = 2$ (b) $N = 3$

Figure 5.3.: Output Complexes

The output complex of the t-resilient k-set agreement problem is defined by the set of simplices of the following form

$$S = \{(j_1, y_1), (j_2, y_2), \dots, (j_r, y_r)\} \tag{5.16}$$

where $|\{j_1, \dots, j_r\}| \geqslant N - t$ and $|\{y_1, \dots, y_r\}| \leqslant k$ and all of its subsets.

The *problem specification* of a decision task is a mapping Δ that maps a basic input simplex S^{n-1} to the set of basic output simplices that are allowed as decision values given the specified initial configuration (input simplex).

5.5.3. Protocol Complexes

When considering an initial configuration ($\triangleq$ an input simplex) of a decision task, one may ask which *final configurations*, i.e., configurations in which all processes have halted (decided or crashed), are reachable from this initial configuration.

The answer to this question defines a mapping from input simplices to sets of final configurations. For an input simplex $S \in \mathcal{I}$ with $\dim S \geqslant n - t - 1$, i.e., $|S| \geqslant n - t$, we define the complex $\mathcal{P}(S)$ to be the reachable subcomplex of $\mathcal{P}$ where only processes take steps that appear in S with initial states as specified in S. The set of configurations (viewed as simplices) that occur in the image of this mapping define a complex, the *protocol complex*. It encompasses the information which final configurations may occur when running the protocol (algorithm). To be more precise, a simplex

$$S = \{C_{j_1}, C_{j_2}, C_{j_3}, \ldots, C_{j_r}\} \tag{5.17}$$

is in the protocol complex if and only if

(1) C_{j_k} is an internal state of process with number j_k in which this process has decided

(2) all j_k are distinct

(3) there exists an execution of the protocol in which exactly the processes with numbers $j_1, j_2, \ldots, j_r$ decide and all other processes crash (and do not reach a decision value)

(4) in the above execution, process p_{j_k} halts with internal state equal to C_{j_k}.

Consider the protocol complex $\mathcal{P}$ of some protocol that solves a decision task with input complex $\mathcal{I}$, output complex $\mathcal{O}$ and decision map Δ. Then, we may assign to every final state of a process (vertex in $\mathcal{P}$) its decision value (vertex in $\mathcal{O}$). More precisely, a final state C_j gets mapped to $\alpha \in \{1, 2, \ldots, M\}$ if and only if process p_j has decided to α in state C_j. This mapping can be extended in a natural manner to a map from simplices in $\mathcal{P}$ to simplices in $\mathcal{O}$. Denote this simplicial vertex map by $\delta : \bigcup \mathcal{P} \to \bigcup \mathcal{O}$.

5.6. Impossibility of k-Set Agreement

In this section, we prove the impossibility of *wait-free* k-set agreement in asynchronous shared-memory environments, as presented in Herlihy and Shavit 1993. Of course, we will heavily rely on methods from algebraic topology. In particular, we will define a class of protocols ("full information protocols") that has stronger system assumption than the asynchronous shared-memory model and analyze its protocol complex. This will help us derive a contradiction. A fortiori, this will establish the impossibility result for usual asynchronous shared-memory systems.

5.6.1. Full Information Protocols

Full information protocols communicate using a set of single-writer shared-memory variables which allows for taking *atomic snapshots*, i.e., the contents of all variables are read in a single step. Denote by $\mathcal{RW}_k^n$ the asynchronous shared-memory model with read-write registers where at most k processes may fail by crashing and denote by $\mathcal{FI}_k^n$ the model of full information protocols where at most k processes may fail by crashing.

The set of algorithms in $\mathcal{FI}_k^n$ is quite restricted: The set $\mathcal{S}_0 \cong \mathcal{I}$ of initial states of processes are arbitrary, but the rest of the algorithm is defined by a single decision vertex map $\delta : \mathcal{F} \to \bigcup \mathcal{O} \cup \{\perp\}$ where $\mathcal{O}$ is an output complex and the set $\mathcal{F}$ is the set of *full information states*. The set $\mathcal{F}$ consists of tuples of the form $(v_1, v_2, \ldots, v_n)$ where every v_j is either contained in $\bigcup \mathcal{I} \cup \{\perp\}$ or again such a tuple. In a full information protocol each process

(1) has a single unbounded shared register it can write.

(2) writes its input value to its shared register in its first step.

(3) after the first step, repeatedly reads the values v_j from all registers R_j, puts them together into a tuple $v = (v_1, \ldots, v_n)$ and writes this tuple v into its own register.

(4) if $\delta(v) \neq \perp$ after such an iteration, the process halts with output value equal to $\delta(v)$.

Theorem 5.3. If there exists a k-set agreement protocol in $\mathcal{RW}_k^n$, then there exists a k-set agreement protocol in $\mathcal{FI}_k^n$.

Proof. The non-trivial part of this proof is to simulate multiple-writer registers with single-writer registers. This is part of Attiya and Welch 2004, Theorem 10.9. ◻

Attiya and Welch 2004, Theorem 10.15 shows that the converse of Theorem 5.3 also holds.

5.6.2. Properties of Full Information Protocols

The most important property of full information protocols to us is the following (Herlihy and Shavit 1993, Corollary 4.9):

Theorem 5.4. Let $\mathcal{P}$ be a protocol in $\mathcal{FI}_{n-1}^n$ and let $S \in \mathcal{I}$ with $\dim S = q$. Then $\mathcal{P}(S)$ is $(q-1)$-acyclic. □

5.6.3. This Implies Impossibility

This section contains the proof of Herlihy and Rajsbaum 2000, Corollary 5.3.

Lemma 5.3. Let $\mathcal{P}$ be a protocol in $\mathcal{FI}^n_{n-1}$. Then $S \mapsto \mathcal{P}(S)$ is an acyclic carrier.

Proof. Property (1) of an acyclic carrier is fulfilled by definition of $\mathcal{P}(S)$. Property (3) is Theorem 5.4. We will show property (2). So let $S \subset S'$ in $\mathcal{I}$. Let $T \in \mathcal{P}(S)$, we have to show $T \in \mathcal{P}(S')$. By definition, T is a final configuration of the protocol where only processes in S take steps with initial states as in S. Since $S \subset S'$, the initial states of processes in S are equal in S and S'. Also, since it is admissible to crash processes initially, every execution contributing to $\mathcal{P}(S)$ also contributes to $\mathcal{P}(S')$. Hence $T \in \mathcal{P}(S')$ as claimed. ■

Theorem 5.5. Let $\mathcal{P}$ be a protocol in $\mathcal{FI}^n_{n-1}$ that solves k-set agreement. Then $k \geqslant n$.

Proof. Suppose by contradiction that $k < n$.

Let $S^{n-1} = \{(1,1),(2,2),(3,3),\dots(n,n)\} \in \mathcal{I}$ and set $\mathcal{A} = P(S^{n-1}) \subset \mathcal{I}$ which is a simplicial subcomplex of $\mathcal{I}$. Let $\mathcal{B} \subset \mathcal{O}$ be defined to contain sets of the form

$$\{(j_1,y_1),(j_2,y_2),\dots,(j_r,y_r)\} \in \mathcal{O} \tag{5.18}$$

where $1 \leqslant y_\ell \leqslant n$ for all $1 \leqslant \ell \leqslant r$.

Define $\pi : \bigcup \mathcal{B} \to \bigcup \mathcal{A}$ by $\pi(j,y) = (y,y)$. Let $\sigma : C(\mathcal{I}) \to C(\mathcal{P})$ be a chain map that is carried by $S \mapsto \mathcal{P}(S)$ (Theorem 5.1(1)). Now set $\phi = \pi_\# \circ \delta_\# \circ \sigma : C(\mathcal{A}) \to C(\mathcal{A})$ and $\Sigma : \mathcal{A} \to P(\mathcal{A})$, $\Sigma(S) = P(S)$. Σ is an acyclic carrier. It is obvious that Σ carries ϕ.

Theorem 5.1(2) now implies that $\phi = \mathrm{id}_{C(\mathcal{A})}$. But since the simplex $S^{n-1} = \{(1,1),(2,2),\dots,(n,n)\}$ does not occur in the image of ϕ because $k < n$, we derive a contradiction. ■

6. Summary

We investigated two applications of topology to problems in distributed computing. These were impossibility proofs of (a) consensus in a number of 1-resilient systems and (b) k-set agreement in asynchronous k-resilient systems. For this, we used methods from (a) point-set topology and (b) algebraic topology.

Point-set topology helped us in providing a way of reasoning about execution trees in a unified way to prove impossibility of consensus. We regarded an execution as a sequence of configurations and equipped the sequence space of all possible executions with a metric which had the property that those executions are close together which share a long common prefix. We then used schedulers to pass to a closed (hence compact) subspace of the space of all executions. This subspace satisfied the precondition of Lebesgue's lemma which provided a uniform step bound after which every configuration is univalent. A model-dependent analysis of configuration similarity then concluded the impossibility proofs.

Algebraic topology, in particular homology, was introduced to examine simplicial complexes. To utilize this, we considered input and output complexes of decision tasks and configuration complexes of protocols (algorithms). The function that maps input simplices to the complex of possible final configurations starting from it turned out to be an acyclic carrier. This was then used to derive the impossibility of wait-free k-set consensus, following Herlihy and Shavit (1993).

A. Topological Prerequisites

A.1. Motivation and Examples

Topology (Bourbaki 1989) is the mathematical discipline that explores the concept of “closeness” and emerging notions. Fundamental is the notion of “neighborhood”. Informally speaking, a topological space is a set together with a structure on this set that specifies which points (elements of the set) are close to each other.

A.1.1. Distances

Topological spaces (though in disguise) are actually encountered in every beginning calculus class. More specifically, the real line $\mathbb{R}$ is a topological space and many of its famous properties are in fact of topological nature. The following example shall exemplify how specific topological concepts might look like.

Example A.1 (The real line)**.** The dominant and natural notion of closeness depends on the definition of *distance* between two real numbers. For real numbers x and y, their distance is defined as

$$d(x, y) = |x - y| \tag{A.1}$$

(see Figure A.1), where $|z|$ denotes the absolute value of z. Starting from this definition, we may now state what it means for real numbers to be close to each other. We may call x and y to be *ε-close* if their distance satisfies

$$d(x, y) < \varepsilon \tag{A.2}$$

where ε is some positive number. So, for every ε, we get a different notion of closeness. Of course, these notions are not independent of each other. The most important dependencies are:

(1) The only point that is ε-close to x for *all* ε, is x itself.

(2) If x is ε-close to y, then y is also ε-close to x.

(3) If x and z are ε-close and z and y are ε-close, then x and y are 2ε-close.

We will later see how these properties generalize to the formal definition of a topology.

We have already discussed an easy but important (topological) property of the real line; above property (1): For any two *distinct* real numbers x and y, there exists

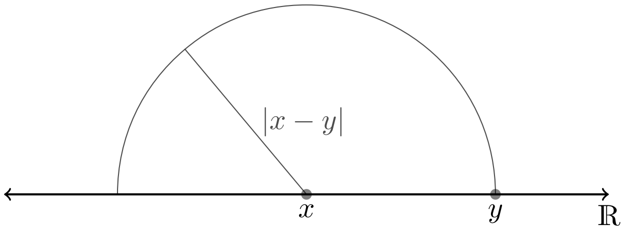


Figure A.1.: Distances on the real line

some positive ε such that x and y are *not* ε-close, for we may choose $\varepsilon = d(x,y)$. This property characterizes the real line as a topological T_1 space. (The properties T_ι for $\iota \in \{0,1,2,3,3\frac{1}{2},4\}$ are the so-called *separation axioms* for topological spaces.) But $\mathbb{R}$ satisfies even more: it is a T_2 or *Hausdorff space. In subsequent sections, we will define for any topological space what it means to be of this important class of spaces, i.e., to be Hausdorff and show some of their convenient properties.* □

We observe that we may define the notion of ε-closeness on any set that, as in the previous example, has a distance function d defined on it. This generalization leads to the definition of *metric spaces* which lie in the class of topological spaces. But before we formally define this, we look at a slight generalization of Example A.1, namely the Euclidean spaces $\mathbb{R}^n$, and discuss in more detail the topological structure and properties that these spaces carry.

Example A.2 (Euclidean spaces)**.** As an analogue of the real absolute value, we have the *norm* of a vector $x \in \mathbb{R}^n$:

$$\|x\| = \left\| \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix} \right\| = \sqrt{x_1^2 + x_2^2 + \cdots + x_n^2} \tag{A.3}$$

Thus, the distance of two vectors x and y in $\mathbb{R}^n$ is defined as

$$d(x,y) = \|x-y\|\,. \tag{A.4}$$

The three properties of Example A.1 still hold. Property (3), also known as the triangle inequality, is depicted in Figure A.2. Its name comes from the fact that in a triangle, the length of any edge is less than the sum of lengths of the other two.

The *ε-neighborhood* of a point x is the set of all points that are ε-close to x. It is also called a *ball* with center x and radius ε and is denoted by $B_\varepsilon(x)$. Now, an *open* set is a set $X \subset \mathbb{R}^n$ such that, for every $x \in X$, there exists an ε-neighborhood of x that is contained in X. Intuitively, an open set is a set that has “a little room” around every of its points, i.e., it does not have a “sharp boundary”. The situation is sketched below in Figure A.3.

Examples of open sets include:

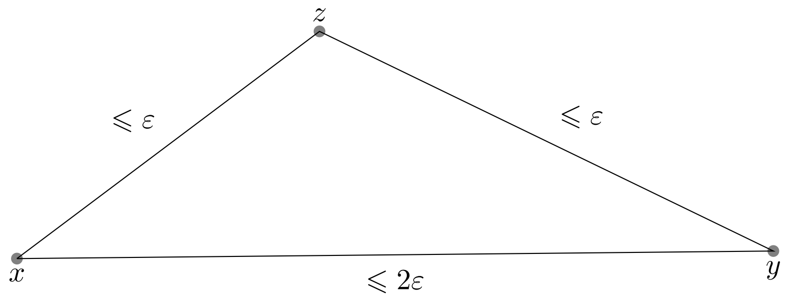


Figure A.2.: Triangle inequality in $\mathbb{R}^2$

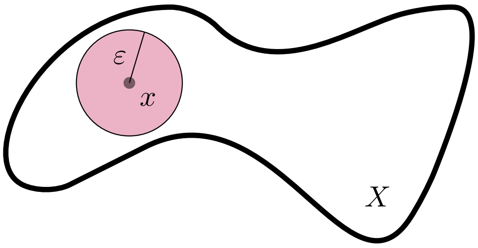


Figure A.3.: Point x has an ε-neighborhood that is contained in X

(1) In $\mathbb{R}^1$, the so-called "open intervals"

$$(a, b) = \{x \in \mathbb{R} \mid a < x < b\} \tag{A.5}$$

are in fact open. For if $x \in (a, b)$, then, by definition, $x - a > 0$ and $b - x > 0$. Hence with $\varepsilon = \min\{x - a, b - x\}$, we get $B_\varepsilon(x) \subset (a, b)$: We may assume without loss of generality that $\varepsilon = x - a$, i.e., $x - a \leqslant b - x$. But then,

$$B_\varepsilon(x) = (x - (x - a), x + (x - a)) \subset (a, x + (b - x)) = (a, b). \tag{A.6}$$

We have just proved that every set of the form (a, b) for real numbers a and b is open. Note that this result still holds if the interval (a, b) is the empty interval, i.e., if $b \leqslant a$. The empty set is always trivially open since there are no elements in it to be checked by the defining condition for open sets. The result even holds if $a = -\infty$ or $b = +\infty$. More generally, every open interval in a totally ordered set is indeed open in the induced *order topology.*

We note that every ε-ball in $\mathbb{R}^1$ is of the form $(x - \varepsilon, x + \varepsilon)$, hence an open interval, hence open. This is part of a more general principle.

(2) In $\mathbb{R}^n$, we may also define "open intervals" by setting

$$(a, b) = \prod_{\iota=1}^{n} (a_\iota, b_\iota) = \{x \in \mathbb{R}^n \mid a_\iota < x_\iota < b_\iota \text{ for every } 1 \leqslant \iota \leqslant n\}. \tag{A.7}$$

Similar to the above case, we may choose

$$\varepsilon = \min\{|x_1 - a_1|, |b_1 - x_1|, \ldots, |x_n - a_n|, |b_n - x_n|\} \tag{A.8}$$

and arrive at the insight that these open intervals are also open sets in the topological sense. Again, we may allow for the a_ι and b_ι to be infinity (positive or negative).

Contrary to the above, however, it is not the case that every ε-ball in $\mathbb{R}^n$ is an open interval, i.e., of the form (a, b) for some $a, b \in \mathbb{R}^n$. But nonetheless, ε-balls are always open as we will see next.

(3) ε-neighborhoods in $\mathbb{R}^n$ are open. This fact is due to the triangle inequality which we already discussed above. Let $x \in \mathbb{R}^n$ be any point and $r > 0$ any radius. We will show that the ball

$$B_r(x) = \{y \in \mathbb{R}^n \mid \|y - x\| < r\} \tag{A.9}$$

is open in $\mathbb{R}^n$: Let $y \in B_r(x)$. Choose $\varepsilon = r - \|y - x\|$. It remains to show that $B_\varepsilon(y)$ is a subset of $B_r(x)$. So, let $z \in B_\varepsilon(y)$, i.e.,

$$\|z - y\| < r - \|y - x\|\,. \tag{A.10}$$

Then, by the triangle inequality and (A.10),

$$\|z - x\| \leqslant \|z - y\| + \|y - x\| < (r - \|y - x\|) + \|y - x\| = r \tag{A.11}$$

and we are done, because this implies $z \in B_r(x)$. The proof is pictured in Figure A.4.

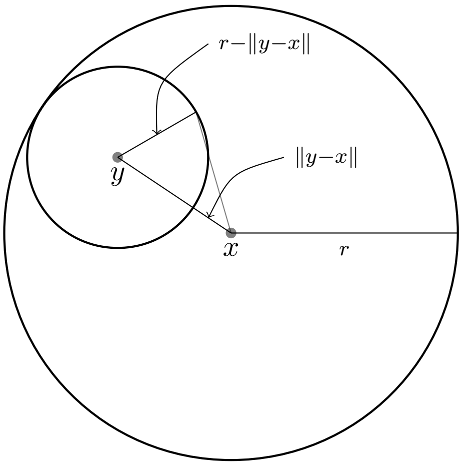


Figure A.4.: ε-balls are open

We have defined the notion of an open set in Euclidean spaces and identified some important classes of sets to be open. In the following, we will generalize the ideas of this example to spaces that are equipped with some way of measuring distances. These spaces are known as *metric spaces.* □

Metric spaces are an immediate generalization of Euclidean spaces. As with any generalization, the idea is to purposely ignore certain aspects and properties of the object in question and focus on just a very limited number of properties that these objects have in common. In our case, the important notion that generalizes Euclidean spaces to metric spaces is that of *distance*. The idea is to forget everything we know about the Euclidean norm $\|\cdot\|$ except that we may use it to define the distance of two points x and y by taking the norm of their difference. Thus, we take the entity "norm" and build a new machine out of it: A machine that takes two points as input and outputs a number — their distance $d(x, y)$.

After identifying a notion that lends itself to generalization, it is crucial to work out which basic properties have to be attributed to it such that one can *define* the notion by means of these properties. We already have listed these properties for our case: Properties (1), (2), (3) from Example A.1 which we will use in the following definition.

Definition A.1. Let X be a non-empty set and $d : X \times X \to [0, \infty)$ a function with the following properties.

(M1) $d(x, y) = 0$ holds if and only if $x = y$

(M2) $d(x, y) = d(y, x)$ for all $x, y \in X$

(M3) $d(x, z) \leqslant d(x, y) + d(y, z)$ for all $x, y, z \in X$

Then we call d a *metric* on X and X a *metric space*. □

It will be the purpose of the next example to explore some properties of such spaces. Note that metric spaces are an important special case of topological spaces. In particular, execution spaces which will deliver our main results are in fact metric spaces.

Example A.3 (Metric spaces)**.** Let X denote a metric space throughout this example. An ε-ball around $x \in X$ is again defined as

$$B_\varepsilon(x) = \{y \in X \mid d(x, y) < \varepsilon\}. \tag{A.12}$$

We also repeat the definition of an open set: A set $A \subset X$ is called open if for every $x \in A$, there exists some $\varepsilon > 0$ such that $B_\varepsilon(x) \subset A$. We could now repeat the proof of the fact that every ε-ball is open from the previous example basically word-by-word. But instead, we will explore properties that are a bit more advanced.

Let us begin by proving that every union of open sets is again open: Let A_ι be open sets for every ι in some non-empty index set I and denote their set-theoretic union by A. We will show that A is open. For every $x \in A$, by the definition of union, there exists some ι_0 such that $x \in A_{\iota_0}$. Now, because A_{ι_0} is open, there exists some $\varepsilon > 0$ such that $B_\varepsilon(x) \subset A_{\iota_0}$. But $A_{\iota_0} \subset A$ means that we are done.

In particular, every union of balls is open. What is interesting now, is that the converse also holds true: Every open set is a union of balls. To prove this, let A be an open set. For any $x \in A$, denote by ε_x some positive number such that $B_{\varepsilon_x}(x) \subset A$. By definition of openness of A, these numbers do exist. We claim that

$$A = \bigcup_{x \in A} B_{\varepsilon_x}(x). \tag{A.13}$$

It is clear that A is contained in the right-hand side of (A.13), because x is contained in any ball around itself that has positive radius. For the opposing direction, note that any ball that appears in the union is a subset of A by construction. Hence the union itself is a subset of A, which concludes the proof. We have just glanced at a very important notion: that of a *basis* of a topology. With this notion, we can express the last result as: The balls form a basis of the topology that is induced by the metric.

We may now ask, of course, if intersections of open sets are again open. Unfortunately, this is *not* true in general as the following example shows: Let $X = \mathbb{R}$ and $d(x,y) = |y - x|$. The sets $A_k = (-\infty, 1/k)$ are all open. However, their intersection

$$\bigcap_{k=1}^{\infty} (-\infty, 1/k) = \left\{ x \in \mathbb{R} \mid x < \frac{1}{k} \text{ for all } k \in \mathbb{N} \right\} = (-\infty, 0] \tag{A.14}$$

is not. The fact that $(-\infty, 0]$ is not open can be seen in the following way: Of course, $0 \in (-\infty, 0]$. But for every $\varepsilon > 0$, we have $\varepsilon/2 \in B_\varepsilon(0)$, while $\varepsilon/2 \notin (-\infty, 0]$, hence the first is not a subset of the latter. A picture clarifying the situation is drawn in Figure A.5.

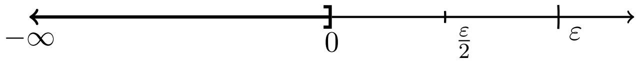


Figure A.5.: The set $(-\infty, 0]$ is not open

A *finite* intersection of open sets, however, is indeed again open as the following reasoning shows: Let $A_1, A_2, \ldots, A_k$ be open sets and let A denote their set-theoretic intersection. We will show that A is open. So, as always, let $x \in A$. Since the A_j are all open, there exist $\varepsilon_1, \varepsilon_2, \ldots, \varepsilon_k$ such that $B_{\varepsilon_j}(x) \subset A_j$ for all $1 \leqslant j \leqslant k$. We set $\varepsilon = \min\{\varepsilon_1, \varepsilon_2, \ldots, \varepsilon_k\}$. But then $B_\varepsilon(x) \subset B_{\varepsilon_j}(x)$ for every j and hence $B_\varepsilon(x) \subset A$ by definition of A.

By inspection of the preceding proof, we find the reason why it does not work in the case of infinitely many sets: The infimum of infinitely many positive numbers need not be positive. And this is exactly what happened in our counterexample: If we choose $x = 0$, then the maximal possible ε_k such that $B_{\varepsilon_k}(0)$ is contained in $(-\infty, 1/k)$ is equal to $1/k$. If we now try to set ε as in the proof above, we get

$$\varepsilon = \inf\{\varepsilon_k \mid k \in \mathbb{N}\} = \inf\{1/k \mid k \in \mathbb{N}\} = 0 \tag{A.15}$$

which is not an admissible radius in the definition of openness. □

A.1.2. Compactness in $\mathbb{R}^n$

This subsection introduces the concept of compactness in the special case of the Euclidean spaces $\mathbb{R}^n$ and tries to communicate a bit of its importance in topology. Compactness is a property of a subset of a topological space that can "make local things global". An example for this would be the well-known theorem "A continuous real function defined on the interval $[a, b]$ is uniformly continuous". Here, the set $[a, b]$ is compact, continuity is a local property and uniform continuity is a global property. We start with the

Definition A.2. A set $C \subset \mathbb{R}^n$ is *compact* if it is bounded (i.e., there some ball with radius $R > 0$ that contains C) and its complement is an open set. □

The most important property of compact sets is the following Theorem whose proof's insight-length ratio is too low to demonstrate it here.

Theorem A.1 (Heine-Borel)**.** Let $C \subset \mathbb{R}^n$ be compact. Further, let A_ι be a family of open sets, indexed by some set I, that *covers* C, i.e.,

$$C \subset \bigcup_{\iota \in I} A_\iota \tag{A.16}$$

Then there exists some *finite* subfamily $A_{\iota_1}, A_{\iota_2}, \dots, A_{\iota_k}$ that covers C, i.e.,

$$C \subset \bigcup_{j=1}^{k} A_{\iota_j} \tag{A.17}$$

□

This theorem can also hold as a definition of compactness: A subset of $\mathbb{R}^n$ is compact if and only if it satisfies the condition of Theorem A.1. The opposing direction is not too hard to prove and is demonstrated in order to get some feeling with the condition of Theorem A.1.

Lemma A.1. Let $C \subset \mathbb{R}^n$ satisfy the condition of Theorem A.1, i.e., for every family of open sets that covers C, there exists a finite subfamily that covers C. Then C is compact.

Proof. We have to show that (1) C is bounded (i.e., there exists some real $R > 0$ such that $C \subset B_R(0)$) and (2) its complement $\mathbb{R}^n \setminus C$ is open.

To prove (1), we choose the following family of open sets: The family of all balls $B_r(0)$ where $r > 0$ is a real number. We already know that these are open. Of course, C is covered by this family of sets, because every $x \in C$ is contained in the ball $B_{2\|x\|}(0)$ for obvious reasons. By hypothesis now, there exists a finite subfamily

$B_{r_1}(0), B_{r_2}(0), \dots, B_{r_k}(0)$ that covers C. But these balls are subsets of the ball $B_R(0)$ where $R = \max\{r_1, r_2, \dots, r_k\}$. Hence

$$C \subset \bigcup_{j=1}^{k} B_{r_j}(0) \subset B_R(0). \tag{A.18}$$

For (2), we have to show that $\mathbb{R}^n \setminus C$ is an open set. So let $x \in \mathbb{R}^n \setminus C$. We define for every $\varepsilon > 0$ the following set

$$D_\varepsilon = \{y \in \mathbb{R}^n \mid d(x,y) > \varepsilon\}. \tag{A.19}$$

In order to use the condition of Theorem A.1, we have to prove that (a) all D_ε are open and (b) C is covered by the D_ε.

Part (a) follows from the triangle inequality: Let $y \in D_\varepsilon$, i.e., $d(x,y) > \varepsilon$. We have to find some $\delta > 0$ such that $d(x,z) > \varepsilon$ for all z with $d(y,z) < \delta$. We claim that this is satisfied by $\delta = d(x,y) - \varepsilon$. Let $d(y,z) < d(x,y) - \varepsilon$, then

$$d(x,z) \geqslant d(x,y) - d(y,z) > d(x,y) - (d(x,y) - \varepsilon) = \varepsilon, \tag{A.20}$$

hence D_ε is open. Part (b) is obvious.

By hypothesis, there exist $\varepsilon_1, \varepsilon_2, \dots, \varepsilon_k$ such that $C \subset \bigcup_{j=1}^{k} D_{\varepsilon_j}$. By setting $\varepsilon = \min\{\varepsilon_1, \varepsilon_2, \dots, \varepsilon_k\} > 0$, we have $C \subset D_\varepsilon$ and thus

$$B_\varepsilon(x) \subset \mathbb{R}^n \setminus D_\varepsilon \subset \mathbb{R}^n \setminus C \tag{A.21}$$

which shows that $\mathbb{R}^n \setminus C$ is open. ■

It is hoped that this proof has created some insight on the nature of the condition of Theorem A.1. Most important and a major source of misunderstandings is the following triviality: The condition does *not* claim that there has to *exist* some finite family of open sets that covers C. This would be trivially fulfilled by any set since $\mathbb{R}^n$ as a subset of itself is an open set and covers any other. The condition reads that *any* open covering of C, no matter how ugly it might look like, has to have a finite subcovering. This observation is of utmost importance.

We will now proceed by showing the result announced in the introduction to this subsection whose proof exemplifies the routinely used reasoning known as "compactness argument". For this, we recall some basic definitions from calculus.

Definition A.3 (Continuity)**.** Let U be a subset of $\mathbb{R}$ and $f : U \to \mathbb{R}$ a function. The function f is called *continuous at a point* $x \in U$ if the following condition is satisfied: For every $\varepsilon > 0$ there exists some $\delta > 0$ such that $x' \in U$ and $|x - x'| < \delta$ implies $|f(x) - f(x')| < \varepsilon$.
The function f is called *continuous* if f is continuous at every point $x \in U$. □

Definition A.4 (Uniform continuity)**.** Let U be a subset of $\mathbb{R}$ and $f : U \to \mathbb{R}$ a function. The function f is called *uniformly continuous* if the following condition is satisfied: For every $\varepsilon > 0$ exists some $\delta > 0$ such that $x, x' \in U$ and $|x - x'| < \delta$ implies $|f(x) - f(x')| < \varepsilon$. □

Of course, uniform continuity implies continuity, but the converse is not true in general as the example $U = \mathbb{R}$ and $f(x) = x^2$ shows. Our goal for now, however, will be to show that the converse does hold in a special case, namely that of compact intervals. The set of compact real intervals is quite easy to determine: It is exactly the set of bounded closed intervals $[a, b]$. The following proof is a most prototypical compactness argument.

Theorem A.2 (Heine-Cantor)**.** Let $f : [a, b] \to \mathbb{R}$ be a function. If f is continuous, then f is uniformly continuous.

Proof. We will use Theorem A.1. Let $\varepsilon > 0$. Since f is continuous, there exists a $\delta_x > 0$ for every $x \in [a, b]$ such that $|x - x'| < \delta_x$ implies $|f(x) - f(x')| < \varepsilon/2$. The condition $|x - x'| < \delta_x$ can be reformulated as $x' \in B_{\delta_x}(x)$. Since every δ_x is greater than zero, we have

$$[a, b] \subset \bigcup_{x\in[a,b]} B_{\delta_x/2}(x) \tag{A.22}$$

which means that the family $B_{\delta_x}(x)$ is an open covering of the interval $[a, b]$. By Theorem A.1, there exist $x_1, x_2, \ldots, x_k$ such that

$$[a, b] \subset \bigcup_{j=1}^{k} B_{\delta_{x_j}/2}(x_j). \tag{A.23}$$

Let $\delta_j = \delta_{x_j}$. If we now set $\delta = \min\{\delta_1, \delta_2, \ldots, \delta_k\}/2$, we are done:

Let $x, x' \in [a, b]$ and $|x - x'| < \delta$. There exists some j such that $x \in B_{\delta_j/2}(x_j)$. By the triangle inequality, $x, x' \in B_\delta(x) \subset B_{\delta_j/2}(x) \subset B_{\delta_j}(x_j)$. We can hence use the definition of δ_j and conclude

$$\left|f(x) - f(x')\right| \leqslant |f(x) - f(x_j)| + \left|f(x_j) - f(x')\right| < \frac{\varepsilon}{2} + \frac{\varepsilon}{2} = \varepsilon. \tag{A.24}$$

■

A.2. Topologies

This section formally defines the notion “topology” resp. “topological space” and introduces basic properties.

A.2.1. Open Sets and Neighborhoods

Without further ado, finally, the fundamental

Definition A.5 (Topological space)**.** Let X be a non-empty set. A set $\mathscr{T} \subset P(X)$ is called a *topology on X* if it satisfies the following properties.

(O1) Every union of sets in $\mathscr{T}$ is an element of $\mathscr{T}$.

(O2) Every finite intersection of sets in $\mathscr{T}$ is an element of $\mathscr{T}$.

(O3) $\emptyset \in \mathscr{T}$ and $X \in \mathscr{T}$

In this case, X is called a *topological space.* The sets $O \in \mathscr{T}$ are called *open sets.* □

To be precise, property (O3) could be omitted from the definition since $\bigcup \emptyset = \emptyset$ and $\bigcap \emptyset = X$ by convention.

This definition is indeed a generalization of the notion of openness as defined in Section A.1. We have defined the notion of openness two times now: One time in the language of metric spaces and one time in the language of topological spaces. We also mentioned that every metric space is also a topological space. This can be done, given a metric d on the space X, by the following definition:

$$\mathscr{T} = \{O \subset X \mid \text{for every } x \in O \text{ there is some } \varepsilon > 0 \text{ such that } B_\varepsilon(x) \subset O\} \quad \text{(A.25)}$$

We proved in the language of metric spaces (O1) that arbitrary unions of open sets are again open and (O2) that finite intersections of open sets are open.

Example A.4. We will now give some examples of topologies in simple settings.

(1) The *trivial topology* exists on any non-empty set X. It is defined as $\mathscr{T} = \{\emptyset, X\}$.

(2) The *discrete topology* also exists on any non-empty set X. It is defined as $\mathscr{T} = P(X)$, i.e., the power set of X.

(3) On a two-element set $X = \{a, b\}$, there are four different topologies, namely

$$\{\emptyset, \{a, b\}\}, \{\emptyset, \{a\}, \{a, b\}\}, \{\emptyset, \{b\}, \{a, b\}\}, \{\emptyset, \{a\}, \{b\}, \{a, b\}\}. \quad \text{(A.26)}$$

(4) There are 29 different topologies on a three-element set $X = \{a, b, c\}$, see Figure A.6. From there on, it gets complicated: 355 topologies on four-element sets, 6942 on five-element sets, 209527 on six-element sets, 9535241 on seven-element sets, 642779354 on eight-element sets and so on.

(5) Any topology in which all singleton sets $\{x\}$ are open, is the discrete topology by (O1). □

In Section A.1, we gave a number of examples of open sets, in particular we showed that specific sets that we called "neighborhoods" or "balls" are open. We will now generalize the notion of neighborhood and show some fundamental properties.

Definition A.6 (Neighborhood)**.** Let X be a topological space and $x \in X$. A set $N \subset X$ is called a *neighborhood of* x if there exists an open set $O \subset X$ such that $x \in O$ and O is contained in N, i.e., $O \subset N$. We will denote the set of all neighborhoods of x by $\mathscr{N}(x)$. □

$$\boxed{\begin{array}{l}
\{\emptyset, X\}, \{\emptyset, \{a\}, X\}, \{\emptyset, \{b\}, X\}, \{\emptyset, \{c\}, X\}, \\
\{\emptyset, \{a, b\}, X\}, \{\emptyset, \{a, c\}, X\}, \{\emptyset, \{b, c\}, X\}, \\
\{\emptyset, \{a\}, \{b, c\}, X\}, \{\emptyset, \{b\}, \{a, c\}, X\}, \{\emptyset, \{c\}, \{a, b\}, X\}, \\
\{\emptyset, \{a\}, \{a, b\}, X\}, \{\emptyset, \{b\}, \{a, b\}, X\}, \{\emptyset, \{a\}, \{a, c\}, X\}, \\
\{\emptyset, \{c\}, \{a, c\}, X\}, \{\emptyset, \{b\}, \{b, c\}, X\}, \{\emptyset, \{c\}, \{b, c\}, X\}, \\
\{\emptyset, \{a\}, \{b\}, \{a, b\}, X\}, \{\emptyset, \{b\}, \{c\}, \{b, c\}, X\}, \\
\{\emptyset, \{a\}, \{c\}, \{a, c\}, X\}, \{\emptyset, \{a\}, \{a, b\}, \{a, c\}, X\}, \\
\{\emptyset, \{b\}, \{a, b\}, \{b, c\}, X\}, \{\emptyset, \{c\}, \{a, c\}, \{b, c\}, X\}, \\
\{\emptyset, \{a\}, \{b\}, \{a, b\}, \{b, c\}, X\}, \{\emptyset, \{b\}, \{c\}, \{a, b\}, \{b, c\}, X\}, \\
\{\emptyset, \{a\}, \{c\}, \{a, c\}, \{b, c\}, X\}, \{\emptyset, \{b\}, \{c\}, \{a, c\}, \{b, c\}, X\}, \\
\{\emptyset, \{a\}, \{b\}, \{a, b\}, \{a, c\}, X\}, \{\emptyset, \{a\}, \{c\}, \{a, b\}, \{a, c\}, X\}, \\
\{\emptyset, \{a\}, \{b\}, \{c\}, \{a, b\}, \{b, c\}, \{a, c\}, X\}
\end{array}}$$

Figure A.6.: The 29 topologies on the set $X = \{a, b, c\}$

We note that, in particular, every open set is neighborhood of any of its points. Simple consequences of the definition are:

(1) If $N \in \mathscr{N}(x)$ and $M \supset N$, then $M \in \mathscr{N}(x)$. In particular, every union of neighborhoods of x is a neighborhood of x.

(2) Every finite intersection of neighborhoods of x is a neighborhood of x.

(3) $x \in N$ for all $N \in \mathscr{N}(x)$.

Properties (1) and (2) justify the name *neighborhood filter* for the set $\mathscr{N}(x)$. A less trivial result is the following.

(4) For every $N \in \mathscr{N}(x)$ there exists some $M \in \mathscr{N}(x)$ such that $N \in \mathscr{N}(y)$ for all $y \in M$, i.e., N is is a neighborhood of all points in M.

Of course, any such M has to be a subset of N. It is sufficient to take M to be the open set containing x as demanded in the definition of a neighborhood. In fact, it is possible to take the notion of neighborhood as the primary notion in the definition of a topology, as opposed to taking the notion of openness as we did in Definition A.5. More precisely, open sets are characterized as being those sets that are neighborhoods of all their points. This allows to *define* the notion of an open set in terms of neighborhoods. Now, given a family $\mathscr{N}(x)$ indexed by $x \in X$ with the above properties (1) to (4), we may define the family

$$\mathscr{T} = \{O \subset X \mid O \in \mathscr{N}(x) \text{ for all } x \in X\} \tag{A.27}$$

which, because the $\mathscr{N}(x)$ satisfy properties (1) to (4), is a topology on X such that $\mathscr{N}(x)$ is exactly the set of neighborhoods of x with respect to this topology $\mathscr{T}$.

Example A.5 (Neighborhoods in metric spaces)**.** Let X be a metric space and $x \in X$. We will characterize the set $\mathscr{N}(x) \subset P(X)$, more precisely we show

$$\mathscr{N}(x) = \{N \in P(X) \mid \text{there exists some } \varepsilon > 0 \text{ such that } B_\varepsilon(x) \subset N\}\,. \tag{A.28}$$

Let N be a neighborhood of x. We want to show that N is an element of the right-hand side of (A.28). By definition, there exists some open set $O \subset N$ that contains the point x. Recalling what it means for a set to open in a metric space, we conclude the existence of some $\varepsilon > 0$ such that $B_\varepsilon(x) \subset O \subset N$. Conversely, if N is an element of the right-hand side of (A.28), then we may set $O = B_\varepsilon(x)$ in the definition of neighborhood since $B_\varepsilon(x)$ is open. □

We will now define what it means for a sequence in X to converge to a point.

Definition A.7 (Convergence)**.** Let X be a topological space and $(x_k)_{k\in\mathbb{N}} \in X^{\mathbb{N}}$ a sequence in X. We say that (x_k) *converges to the point* $x \in X$ if for every neighborhood N of x, there exists some integer $K \in \mathbb{N}$ such that for every $k \geqslant K$, $x_k \in N$. In this case, x is called *limit point of* (x_n) and (x_n) is called *convergent.* □

Compare this definition to the definition of convergence in metric spaces: The sequence (x_k) converges to x if "for all $\varepsilon > 0$ there exists some integer $K \in \mathbb{N}$ such that for every $k \geqslant K$, $x_k \in B_\varepsilon(x)$". The generalization of Definition A.7 is that the prototypical neighborhoods $B_\varepsilon(x)$ were replaced by arbitrary neighborhoods of x.

One fact we recall from Example A.3 is that in the case of a metric space X, every open set is the union of ε-balls. In such a case, we call the set of ε-balls a *basis* of the topology. More specifically, a basis $\mathscr{B}$ of a topology $\mathscr{T}$ on X is a set of subsets of X such that every open set $O \in \mathscr{T}$ is a union of elements of $\mathscr{B}$. Trivially, the topology itself is always a basis. A special case occurs when there exists a *countable* basis $\mathscr{B}$ of a topology. These spaces are called *second countable* or AA2 spaces. Euclidean spaces $\mathbb{R}^n$ have a basis consisting of the ε-balls. But they even are AA2 spaces since the set of balls with rational radius and rational center also form a basis of the Euclidean topology, i.e.,

$$\mathscr{B} = \{B_\varepsilon(x) \subset \mathbb{R}^n \mid x \in \mathbb{Q}^n \text{ and } \varepsilon \in \mathbb{Q} \cap (0,\infty)\}\,. \tag{A.29}$$

This set $\mathscr{B}$ is countable by Cantor diagonalization.

AA2 spaces are spaces in which it suffices to use the term "sequence", i.e., a mapping $\mathbb{N} \to X$, when talking about convergence. In other spaces it might be too restrictive for its notion of convergence that the domain $\mathbb{N}$ of the sequence is just countable. We would have to generalize sequences to either *nets* or *filters*, the former being mappings $\Lambda \to X$ where Λ is not necessarily countable, but has to carry an additional structure; that of a *directed set*[1]. If this is not done, popular theorems such as "A map $F : X \to Y$ is continuous if and only if for each sequence x_k in

[1] A directed set is a set Λ together with a preorder (reflexive and transitive relation) $\preceq$ on Λ such that for every λ and μ in Λ, there exists a ν such that $\lambda \preceq \nu$ and $\mu \preceq \nu$.

X converging to x, the sequence $f(x_k)$ converges to $f(x)$" would be plainly false. It *is* true in any topological space, however, if the word "sequence" is replaced by "net". But the introduction of this generalization has a few complications attached to it. For example, it need not be the case that a net has only one limit, but it may have multiple. There even exist nets that converge to every point in the space. Nevertheless, we will not take on the endeavor to explore the theory in this direction since the spaces we will look at do not have this inconvenience.

We call a topological space *Hausdorff* if for every two distinct points x and y, there exist neighborhoods N_x and N_y of x and y, respectively, such that $N_x \cap N_y = \emptyset$. Metric spaces are Hausdorff.

A.2.2. Closure, Interior, Boundary, Density

This subsection introduces accompanying notions for talking about topological spaces.

Definition A.8 (Closure)**.** Let X be a topological space. A set $A \subset X$ is called *closed* if it is the complement of an open set, i.e., if $X \setminus A$ is open.
The *closure* of a set $B \subset X$, denoted by $\overline{B}$, is the least (with respect to set inclusion) closed set that contains B as a subset. □

The first question that arises here is, of course, whether the notion of closure is indeed well-defined. To be more precise, the question is whether there does exist a *least* closed set that contains B for every $B \subset X$. Before we answer this question in the affirmative, we collect some simple facts:

Lemma A.2. Let X be a topological space. The following assertions are true.

(1) Every finite union of closed sets is closed.

(2) Every intersection of closed sets is closed.

(3) $\emptyset$ and X are both closed.

Proof. We show (1): Let $A_1, A_2, \ldots, A_n$ be closed sets, i.e., the $X \setminus A_j$ are open. With use of De Morgans law,

$$X \setminus \bigcup_{j=1}^{n} A_j = \bigcap_{j=1}^{n} X \setminus A_j \tag{A.30}$$

which is open by defining property (O2).

Part (2) is proved just as easy. Let A_i be an arbitrary family of closed set indexed by $i \in I$. Again, De Morgans law yields

$$X \setminus \bigcap_{i \in I} A_i = \bigcup_{i \in I} X \setminus A_i \tag{A.31}$$

and property (O1) tells that we are done.

The sets $\emptyset$ and X are closed since they are open and each other's complement. ■

Note the duality of these assertions and the condition on open sets in the definition of a topology. It is possible to define the notion of a topology by defining what sets should be closed, as opposed to defining what sets should be open as in Definition A.5. The properties that a family of sets has to fulfill such that it appears as the family of closed sets of some topology are exactly the assertions of Lemma A.2.

The question whether the notion of closure is actually well-defined for any set $B \subset X$ follows from the fact every intersection of closed sets is again closed. More precisely,

$$\overline{B} = \bigcap \{A \subset X \mid A \text{ is closed and } B \subset A\}, \tag{A.32}$$

i.e., the closure of B is equal to the intersection of all closed sets A that contain B. The set on right-hand side of (A.32) is closed by (2) of Lemma A.2 and is of course contained in every other closed set by definition of intersection. Hence we constructed $\overline{B}$ for every $B \subset X$.

For spaces in which sequences suffice to build a proper notion of convergence, in particular in AA2 spaces (see Section A.2.1), the following important characterization of closure holds:

Lemma A.3. Let X be an AA2 space and $B \subset X$. Then the closure of B is equal to the set of limit points that sequences in B have, i.e.,

$$\overline{B} = \left\{x \in X \mid \text{there exists a sequence } (x_k)_{k\in\mathbb{N}} \in B^{\mathbb{N}} \text{ with } x_k \to x\right\}. \tag{A.33}$$

Proof. We first prove that $\overline{B}$ is contained in the right-hand side R. It suffices to show that the right-hand side is closed and contains B as a subset. The latter claim is clear since every $x \in B$ is limit of the constant sequence $(x)_{k\in\mathbb{N}}$. We show closedness by contradiction. Suppose that $X \setminus R$ is not open. Then, by definition, there exists some $x \in X \setminus R$ such that for every neighborhood N of x, we have $N \cap R \neq \emptyset$. We will construct a sequence of points in B that converges to x, deriving the desired contradiction.

In a first step, we will show that there exists a sequence of points *in* R that converges to x and then show how this implies the claim. By hypothesis, there exists some countable basis $\mathscr{B}$ of the topology. Let $(N_k)_{k\in\mathbb{N}}$ denote the family of basis sets that contain x. It is easy to show that for every neighborhood N of x, there exists a $k \in \mathbb{N}$ such that $N_k \subset N$. Now, for every $k \in \mathbb{N}$, let x_k be an arbitrary point in $R \cap \bigcap_{i=1}^k N_i$ (which is non-empty, see above). This is a sequence of points in R that converges to x. For let $N \in \mathscr{N}(x)$, then there exists some $K \in \mathbb{N}$ such that $N_K \subset N$. Hence for any $k \geqslant K$, we have $x_k \in \bigcap_{i=1}^k N_i \subset N_K \subset N$ which shows $x_k \to x$.

To show that the existence of a sequence in R converging to x implies the existence of a sequence in B converging to x, we take for every point $x_k \in R$ from the above construction a sequence $(\xi_{k,j})_{j\in\mathbb{N}} \in B^{\mathbb{N}}$ converging to x_k. These sequences exist by the definition of R. We claim that the following sequence y_k converges to x. The set

$\bigcap_{i=1}^{k} N_i$ containing x_k and being open, there exists some index $j_k \in \mathbb{N}$ such that for all $j \geqslant j_k$, $\xi_{k,j} \in \bigcap_{i=1}^{k} N_i$. We define

$$y_k = \xi_{k,j_k} \in B \cap \bigcap_{i=1}^{k} N_i \tag{A.34}$$

which obviously converges to x. This is a contradiction and we have shown $\overline{B} \subset R$.

The converse direction $R \subset \overline{B}$ is much easier. Suppose that there exists some $x \in R \setminus \overline{B}$. Because x lies in the open set $X \setminus \overline{B}$, there exists some neighborhood N of x such that

$$N \subset X \setminus \overline{B} \subset X \setminus B. \tag{A.35}$$

But this relation denies the existence of a sequence in B converging to x, which contradicts the assumption. ■

It is also possible to define a topological space in terms of its *closure operator*, i.e., the map $P(X) \to P(X)$ that takes a set $A \subset X$ to its closure. We can retain the topology from the closure operator, because a set $A \subset X$ is closed if and only if $\overline{A} = A$. The axioms needed to define the family of closed sets of a topology by the above procedure are the following:

Lemma A.4. Let X be a topological space and let $C : P(X) \to P(X)$ be its closure operator. The following assertions are true for any $A, B \subset X$:

(1) $C(\emptyset) = \emptyset$

(2) $A \subset C(A)$

(3) $C(C(A)) = C(A)$

(4) $C(A \cup B) = C(A) \cup C(B)$

Proof. (1) is clear since $\emptyset$ is a closed set. (2) and (3) are immediate consequences of the definition of closure. We will now prove (4). The inclusion $C(A \cup B) \subset C(A) \cup C(B)$ is a consequence of the fact that $C(A) \cup C(B)$ is closed by Lemma A.2(1) and of course contains both A and B as subsets. For the other inclusion, we note that it is sufficient to show $C(A) \subset C(A \cup B)$ which is clear since $A \subset A \cup B$. ■

Dual to the notion of closure is that of interior, as defined next.

Definition A.9 (Interior)**.** Let X be a topological space and $A \subset X$. We call the (with respect to set inclusion) greatest open set that is contained in A the *interior of* A, denoted by A°. □

As with closure, we have to check that this is a well-defined notion, i.e., that A° exists for all $A \subset X$. This follows from the formula

$$A^\circ = \bigcup \{B \subset X \mid B \text{ is open and } B \subset A\}\,. \tag{A.36}$$

A set $A \subset X$ is open if and only if $A^\circ = A$. The important properties of the *interior operator* $P(X) \to P(X)$, $A \mapsto A^\circ$ are:

Lemma A.5. Let X be a topological space and let $I : P(X) \to P(X)$ be its interior operator. The following assertions are true for any $A, B \subset X$:

(1) $I(X) = X$

(2) $I(A) \subset A$

(3) $I(I(A)) = I(A)$

(4) $I(A \cap B) = I(A) \cap I(B)$

Proof. (1) holds because the set X is open by definition, (2) and (3) are obvious. To prove (4), we show both set inclusions. The inclusion from $I(A \cap B) \subset I(A) \cap I(B)$ is true, because $A \cap B \subset A$ and $A \cap B \subset B$. We now show $I(A) \cap I(B) \subset I(A \cap B)$. The set $I(A) \cap I(B)$ is open and contained in both A and B, hence contained in $A \cap B$. The claim now follows from the definition of interior. ■

The notions of closure and interior are connected by the following formula

$$A^\circ = \left(\overline{A^c}\right)^c, \overline{A} = ((A^c)^\circ)^c \tag{A.37}$$

where $B^c = X \setminus B$ denotes the complement of $B \subset X$.

Example A.6 (Intervals)**.** We will demonstrate the use of the notions interior and closure with real intervals. The interior and the closure of an interval do not depend on whether the boundary points belong to the interval or not. More precisely, the interiors of (a, b), $(a, b]$, $[a, b)$ and $[a, b]$ are all equal to (a, b) and their closures are all equal to $[a, b]$. It follows from the fact that (a, b) is open, $[a, b]$ is closed and neither $(a, b]$ nor $[a, b)$ are open or closed. □

We may generalize this situation and identify for any set $A \subset X$ a set of points for which it does not matter whether they are added or removed when considering interior and closure.

Definition A.10 (Boundary)**.** Let X be a topological space and $A \subset X$. We call $\partial A = \overline{A} \setminus A^\circ$ the *boundary of* A. □

We state the following observations.

Lemma A.6. Let X be a topological space and let $A \subset X$. The following statements are true.

(1) $\partial A = \{x \in X \mid \text{for all } N \in \mathscr{N}(x) \text{ it is } N \cap A \neq \emptyset \text{ and } N \cap A^c \neq \emptyset\}$

(2) $\partial A = \partial(A^c)$

(3) $(A \setminus \partial A)^\circ = A^\circ$

(4) $\overline{A \cup \partial A} = \overline{A}$

(5) ∂A is closed

(6) $\overline{A} = A \cup \partial A$. In particular, A is closed if and only if $\partial A \subset A$.

(7) $A^\circ = A \setminus \partial A$. In particular, A is open if and only if $\partial A \cap A = \emptyset$.

Proof. (1): From the definition and equation (A.37), we deduce $\partial A = A^{coc} \setminus A^\circ = A^{coc} \cap A^{\circ c}$. It hence suffices to show

$$A^{\circ c} = \{x \in X \mid \text{for all } N \in \mathscr{N}(x) \text{ it is } N \cap A^c \neq \emptyset\}. \tag{A.38}$$

But this is trivially equivalent (by taking complements) to

$$A^\circ = \{x \in X \mid \text{there exists some } N \in \mathscr{N}(x) \text{ such that } N \cap A^c = \emptyset\}. \tag{A.39}$$

which is true, because the relation $N \cap A^c = \emptyset$ is the same as the relation $N \subset A$.

(2) is a trivial consequence of (1).

(3): After a simple calculation involving De Morgan's law and $R \setminus S = R \cap S^c$, we arrive at the equation

$$(A \setminus \partial A)^\circ = \big(A \cap (\overline{A}^c \cup A^\circ)\big)^\circ \tag{A.40}$$

which by Lemma A.5(4) is equal to

$$A^\circ \cap \big(\overline{A}^c \cup A^\circ\big)^\circ \tag{A.41}$$

Since $A^\circ \subset \overline{A}^c \cup A^\circ$ and $A^{\circ\circ} = A^\circ$, we get

$$(A \setminus \partial A)^\circ \supset A^\circ \cap A^\circ = A^\circ \tag{A.42}$$

and we are done, the other inclusion being trivial.

(4) follows from (2), (3) and (A.37) as the following calculation shows:

$$\overline{A \cup \partial A} = (A \cup \partial A)^{coc} = (A^c \cap (\partial A)^c)^{\circ c} \tag{A.43}$$

$$= (A^c \setminus \partial\,(A^c))^{\circ c} = A^{c\circ\circ c} = A^{coc} = \overline{A} \tag{A.44}$$

(5) is clear since $\partial A = \overline{A} \cap A^{\circ c}$ is an intersection of two closed sets.

(6) and (7) are simple calculations. ■

We now turn to a different notion that is derived from the notion of closure, namely *density*. Informally, we will call a set dense if every point in the space is arbitrarily close to a point of the dense set. The formal definition follows now.

Definition A.11 (Density)**.** Let X be a topological space and $A \subset X$. We call A *dense in* X if the closure of A in X is equal to X, i.e., $\overline{A} = X$. □

Equivalent statements are summarized in the next

Lemma A.7. Let X be a topological space and $A \subset X$. The following statements are equivalent:

(1) A is dense in X.

(2) For every non-empty open set $O \subset X$ it follows that $A \cap O \neq \emptyset$.

(3) For every neighborhood N of a point $x \in X$ it follows that $A \cap N \neq \emptyset$.

Proof. The equivalence (2)⇔(3) is trivial. Let A be dense and suppose that (2) does not hold. Then there exists a non-empty open set O with $A \subset O^c$ where O^c denotes $X \backslash O$. But since $O^c \neq X$ and O^c is closed, we deduce $\overline{A} \neq X$ which is a contradiction. Conversely, let (2) hold and suppose that $\overline{A} = C \neq X$. But then the complement of C is non-empty, open, and has trivial intersection with A. Contradiction. ■

Example A.7. The set $\mathbb{Q} \subset \mathbb{R}$ is dense: Let $N \subset \mathbb{R}$ be a neighborhood of some $x \in \mathbb{R}$. Then, by definition, there exists some $\varepsilon > 0$ such that $B_\varepsilon(x) \subset N$. The decimal expansion of x yields a sequence (q_k) that converges to x. This implies that there exists some $K \in \mathbb{N}$ such that $q_K \in B_\varepsilon(x) \subset N$. The claim follows because, by construction, $q_K \in \mathbb{Q}$. □

A.2.3. Continuity

In the previous sections, we studied the objects "topological spaces". It is the purpose of this section to deal with "morphisms" of such objects, i.e., functions between topological spaces that preserve the topological structure. We will then have laid the ground to study the *category* of topological spaces (Herrlich and Strecker 1973).

Definition A.12 (Continuity)**.** Let X and Y be topological spaces. Furthermore, let $f : X \to Y$ be a function. We call f *continuous* if for every open set $O \subset Y$, it follows that its inverse image $f^{-1}[O]$ is open in X. □

By taking complements and recalling $f^{-1}[Y \setminus A] = X \setminus f^{-1}[A]$ for all $A \subset Y$, we arrive at the insight that f is continuous if and only if $f^{-1}[C]$ is closed in X for every set C that is closed in Y.

Example A.8. Let X be a set equipped with the discrete topology. Then *every* map $f : X \to Y$ is continuous, because every subset of X is open. Conversely, if Y is equipped with the trivial topology, i.e., only $\emptyset$ and Y are open, then again every map $f : X \to Y$ is continuous, because $f^{-1}[\emptyset] = \emptyset$ and $f^{-1}[Y] = X$ which are in any case open in X. □

If Y a topological space and X is any non-empty set, we may ask ourselves with which topology we must equip X such that a given mapping $f : X \to Y$ becomes continuous. Of course, we want to do this in the most general fashion, i.e., we do not want to add too many open sets to the to be defined topology on X, just enough to make f continuous. We are obliged to have sets of the form $f^{-1}[O]$ where $O \subset Y$ is open as open sets in X. But by recalling all those useful properties of the inverse image, it also turns out that these sets are enough: The sets of the above form $f^{-1}[O]$ form a topology on X with respect to which f is continuous. We call this topology on X the *initial topology* with respect to $f : X \to Y$.

Example A.9 (Subspace topology)**.** If $X \subset Y$ and Y is equipped with a topology, we may consider the inclusion map $\iota : X \hookrightarrow Y$, i.e., $\iota(x) = x$ for all $x \in X$, and equip X with the initial topology with respect to ι. We call this topology on X the *subspace topology* inherited from Y. The open sets of X are exactly the sets $X \cap O$ where $O \subset Y$ is open. □

We may well go the opposite direction and ask, given a mapping $f : X \to Y$ where X is a topological space and Y is an arbitrary non-empty set, which topology on Y makes f continuous and has the least number of open sets. This time, it turns out that indeed the sets $A \subset Y$ such that $f^{-1}[A] \subset X$ is open form a topology on Y. This topology on Y is called the *final topology* with respect to $f : X \to Y$.

We have already seen a notion called "continuity" in Definition A.3 where we defined it for maps $U \to \mathbb{R}$ where $U \subset \mathbb{R}$. It would be embarrassing if those notions would not agree for maps $U \to \mathbb{R}$. Luckily, the following lemma holds.

Lemma A.8. Let $U \subset \mathbb{R}$ and $f : U \to \mathbb{R}$. The following are equivalent:

(1) f is continuous with respect to Definition A.3.

(2) f is continuous with respect to Definition A.12 (where U is equipped with the subspace topology inherited from $\mathbb{R}$).

Proof. (1)⇒(2): Let $O \subset \mathbb{R}$ be non-empty and open. Let $x \in f^{-1}[O]$. Then there exists some $\varepsilon > 0$ such that $B_\varepsilon(f(x)) \subset O$ since O is open. But now, by (1), there exists some $\delta > 0$ such that $y \in B_\delta(x)$ implies $f(y) \in B_\varepsilon(f(x))$, i.e.,

$$B_\delta(x) \subset f^{-1}[B_\varepsilon(f(x))] \subset f^{-1}[O]. \tag{A.45}$$

(2)⇒(1): Let $x \in U$ and $\varepsilon > 0$. The set $B_\varepsilon(f(x))$ is open in $\mathbb{R}$. Hence also $f^{-1}[B_\varepsilon(f(x))]$ is open. Hence there exists some $\delta > 0$ such that

$$B_\delta(x) \subset f^{-1}[B_\varepsilon(f(x)). \tag{A.46}$$

But this implies the condition of (1). ■

A.2.4. Compactness

We briefly discussed compactness in Section A.1.2 for the case of subsets of $\mathbb{R}^n$. Since we do not have the notion of distance and hence boundedness in general topological spaces, the idea is to take the conclusion of Theorem A.1 as the definition of compactness. Below, we collect the most important facts about compact sets.

Definition A.13 (Compactness)**.** Let X be a topological space. We call X *compact* if for any collection $(A_\iota)_{\iota\in I}$ of open sets for which $X = \bigcup A_\iota$, there exists some $n \in \mathbb{N}$ and $\iota_1, \ldots, \iota_n$ such that $X = \bigcup_{k=1}^n A_{\iota_k}$. We call a subset A of a topological space *compact* if A is compact with respect to the subspace topology inherited from X. □

Example A.10. A space equipped with the discrete topology is compact if and only if it is finite. This follows easily because all singleton sets $\{x\}$ are open in discrete spaces. □

Lemma A.9. Let X be a topological space and $A \subset X$. The following assertions are true:

(1) If X is compact and A is closed, then A is compact.

(2) If X is Hausdorff and A is compact, then A is closed.

Proof. (1): For any open cover (A_ι) of A, the family (A_ι) together with the open set $X \setminus A$ is an open cover of X. Since X is compact, there exist finitely many indices $\iota_1, \ldots, \iota_n$ such that $X = \bigcup_{k=1}^n A_{\iota_k} \cup (X \setminus A)$. By intersecting both sides with A, we arrive at $A = \bigcup_{k=1}^n A_{\iota_k}$ which shows that A is compact.

(2): We show that the complement of A is open. Let $x \in X \setminus A$. By the Hausdorff property, for every $y \in A$, there exist disjoint open sets $U(y)$ and $V(y)$ such that $y \in U(y)$ and $x \in V(y)$. The family $U(y)$ where $y \in A$ is an open covering of A. Because A is compact, there exist $y_1, \ldots, y_n$ such that $A = \bigcup_{i=1}^n U(y_i)$. Setting $V = \bigcap_{i=1}^n V(y_i)$ reveals that V is an open neighborhood of x ((O2) in Definition A.5) which is disjoint to A, hence $V \subset X \setminus A$. This proves that $X \setminus A$ is open, i.e., A is closed. ■

Lemma A.10. Let $f : X \to Y$ be continuous and let X be compact. Then $f[X]$ is compact.

Proof. Follows immediately from the definitions. ■

Lemma A.11. Let X be a compact space and $f : X \to \mathbb{R}$ continuous. Then f attains its minimum, i.e., there exists some $x \in X$ such that

$$f(x) = \inf\{f(y) \mid y \in X\}. \tag{A.47}$$

Proof. By Lemma A.10, the image $f[X]$ is a compact set in $\mathbb{R}$. By Lemma A.9, this set is closed. We may deduce the result from order completeness of $\mathbb{R}$.[2] ■

[2] For every set $B \subset \mathbb{R}$ which is bounded from below, the infimum of B exists in $\mathbb{R}$.

A.2.5. Product Spaces

In this section, we will answer the question, which topology is “natural” to equip a product space $X = \prod X_\iota$ with when all X_ι are topological spaces. We do this by considering the projection mappings $\pi_\iota : X \twoheadrightarrow X_\iota$ and equipping X with a slight generalization of the initial topology. Namely, we will have to make *all* projection mapping continuous, not only one. This is done with the following

Definition A.14. Let (X_ι) be a family of topological spaces and let $X = \prod X_\iota$ be the set-theoretic product. We call the topology induced by sets of the form $\pi_\iota^{-1}[O]$ where $O \subset X_\iota$ is open the *product topology* on X. □

A most important result that we do not prove here for space limitations is the following (Bourbaki 1989, Chapter I, §9, no. 5, Theorem 3).

Theorem A.3 (Tychonoff)**.** Let $X = \prod X_\iota$ be equipped with the product topology. The following are equivalent:

(1) X is compact.

(2) All X_ι are compact. □

Bibliography

Alpern, Bowen and Fred B. Schneider. Defining liveness. Technical Report TR85-650, Cornell University, 1985.

Attiya, Hagit and Jennifer Welch. *Distributed Computing: Fundamentals, Simulations, and Advanced Topics.* John Wiley & Sons, second edition, 2004.

Bourbaki, Nicolas (pseudonym). *General Topology, Chapters 1–4.* Elements of Mathematics. Springer, 1989.

Charron-Bost, Bernadette, Sam Toueg, and Anindya Basu. Revisiting safety and liveness in the context of failures. In *Proceedings of CONCUR 2000 — Concurrency Theory*, pages 552–565. Springer, 2000.

Dolev, Danny, Cynthia Dwork, and Larry Stockmeyer. On the minimal synchronism needed for distributed consensus. *Journal of the ACM* **34**(1):77–97, 1987.

Fich, Faith and Eric Ruppert. Hundreds of impossibility results for distributed computing. *Distributed Computing* **16**(2):121–163, 2003.

Fischer, Michael J., Nancy A. Lynch, and Michael S. Paterson. Impossibility of distributed consensus with one faulty process. *Journal of the ACM* **32**(2):374–382, 1985.

Hatcher, Allan. *Algebraic Topology.* Cambridge University Press, 2002.

Herlihy, Maurice and Sergio Rajsbaum. Algebraic spans. *Mathematical Structures in Computer Science* **10**(4):549–573, 2000.

Herlihy, Maurice and Nir Shavit. The asynchronous computability theorem for t-resilient tasks. In *Proceedings of the 25th Annual ACM Symposium on Theory of Computing*, pages 111–120. 1993.

Herrlich, Horst and George E. Strecker. *Category Theory: An Introduction.* Allyn and Bacon, 1973.

Lamport, Leslie. Proving the correctness of multiprocess programs. *IEEE Transactions on Software Engineering* **SE-3**(2):125–143, 1977.

Lubitch, Ronit and Shlomo Moran. Closed schedulers: A novel technique for analyzing asynchronous protocols. *Distributed Computing* **8**(4):203–210, 1995.

Lynch, Nancy A. *Distributed Algorithms.* Morgan Kaufmann, 1996.

Moses, Yoram and Sergio Rajsbaum. A layered analysis of consensus. *SIAM Journal on Computing* **31**(4):989–1021, 2002.

Saks, Michael and Fotios Zaharoglou. Wait-free k-set agreement is impossible: The topology of public knowledge. *SIAM Journal on Computing* **29**(5):1449–1483, 2000.

Santoro, Nicola and Peter Widmayer. Time is not a healer. In *Proceedings of the 6th Annual Symposium on Theoretical Aspects of Computer Science*, pages 304–313. Springer, 1989.

Spanier, Edwin H. *Algebraic Topology.* McGraw-Hill, 1966.