FAKULTÄT FÜR !NFORMATIK
Faculty of Informatics

# Characterization of Consensus Solvability under Message Adversaries

DISSERTATION

submitted in partial fulfillment of the requirements for the degree of

**Doktor der Technischen Wissenschaften**

by

**Kyrill Winkler**
Registration Number 00201623

to the Faculty of Informatics

at the TU Wien

Advisor: Univ.-Prof. Dr. Ulrich Schmid

The dissertation has been reviewed by:

Sergio Rajsbaum

Stefan Schmid

Vienna, 5th August, 2019

Kyrill Winkler

Technische Universität Wien
A-1040 Wien ▪ Karlsplatz 13 ▪ Tel. +43-1-58801-0 ▪ www.tuwien.ac.at

# Declaration of Authorship

Kyrill Winkler

I hereby declare that I have written this Doctoral Thesis independently, that I have completely specified the utilized sources and resources and that I have definitely marked all parts of the work - including tables, maps and figures - which belong to other works or to the internet, literally or extracted, by referencing the source as borrowed.

Vienna, 5th August, 2019

Kyrill Winkler

This thesis is dedicated to my grandparents.

# Acknowledgements

Undoubtedly the most crucial agent in the causal chain of events that led to this thesis[1] and whom I want to thank first and foremost is Ulrich Schmid, who kindled my interest in scientific work and introduced me to the field of distributed algorithms. I am truly impressed by his extraordinary patience and empathy that enables him to be an outstanding teacher and by his willingness to spend uncountable hours to discuss scientific and sometimes even more important topics with me, despite his vast amount of different responsibilities. It is fair to say that, without him, none of this would exist.

Much of the thesis would not be in its current form without the people who collaborated with me and with whom I discussed and worked over the past years. I would like to thank all of them and in particular Bernadette Charron-Bost, Matthias Függer, Thomas Nowak, Hugo Rincón Galeana, and Manfred Schwarz. I also want to thank the group secretary Traude for taking care of my administrative needs, often without me even noticing, and the technicians Thomas and Heinz, who provided the infrastructure necessary for my work.

Furthermore, I want to thank all the friends and colleagues that I had the privilege of getting to know during my work at TU Wien and who were always good for comic relief from work, giving the institute a truly cozy atmosphere, as well as being responsible for countless inspiring and entertaining lunch breaks. Among them are Alex, Andreas, Josef, Krisztina, Laura, Martin Z., Robert, Roman, and many others. Special credits go to Martin P. for being available when things did not go as planned, as well as to Flo and Jürgen for listening with understanding and sympathy.

Finally, I want to thank my family and friends for their unwavering support and at times much needed outside perspective. Their contribution cannot be overstated, as it was really them who supplied the motivation and energy needed for my work on this thesis. In this regard, I especially want to thank my parents, Julia and Gebhard, and Angela, my significant other, and her family.

[1] This work was supported by the Austrian Science Fund (FWF) under project ADynNet (P28182) and RiSE/SHiNE (S11405).

# Kurzfassung

Diese Arbeit beschäftigt sich mit verschiedenen Charakterisierungen für die Lösbarkeit von Consensus in dynamischen Netzen, welche sich unter der Kontrolle eines allwissenden Message Adversary befinden. Das Message Adversary Modell beschreibt ein System von $n$ verteilten Agenten, meist "Prozesse" genannt, die einen verteilten Algorithmus ausführen und über den Austausch von Nachrichten in synchronen Runden miteinander kommunizieren. In jeder Runde bestimmt der Message Adversary welche Nachrichten ihr Ziel erreichen und welche verloren gehen, was üblicherweise mit Hilfe eines Kommunikationsgraphen dargestellt wird. Die allgemeinste Darstellung eines Message Adversary gelingt als Menge von unendlichen Sequenzen solcher Kommunikationsgraphen, wobei jeder Kommunikationsgraph in dieser Sequenz die Kommunikation einer Runde abbildet.

Vermutlich das wichtigste theoretische Resultat dieser Arbeit ist eine präzise topologische Charakterisierung der Lösbarkeit von Consensus unter allgemeinen Message Adversaries, und die deutlich weniger abstrakte kombinatorische Charakterisierung für die wichtige Teilklasse der abgeschlossenen Message Adversaries. Mit Hilfe dieser Resultate können wir zum ersten Mal präzise und rigoros die Kerneigenschaft eines beliebigen Message Adversary formulieren, die einen Lösungsalgorithmus für Consensus erlaubt. Das wahrscheinlich wichtigste praktische Resultat ist ein optimaler Algorithmus für dynamische Netze, welche nur für kurze Zeit stabil werden, zusammen mit einem Korrektheitsbeweis und einem Beweis der Optimalität. Der Algorithmus ist relativ einfach und kann daher als Grundlage für die Lösung von Consensus in realen Systemen dienen, die einem massiven, transienten Verlust von Nachrichten ausgesetzt sind.

Grob zusammengefasst enthält diese Arbeit eine Schar von Unmöglichkeitsbeweisen sowie Lösungsalgorithmen für immer stärkere Message Adversaries:

Zuerst beschäftigen wir uns mit oblivious Message Adversaries, die über eine Menge an erlaubten Kommunikationsgraphen definiert sind; die Sequenzen, die ein solcher Message Adversary generieren kann, sind sämtliche Kombinationen der Kommunikationsgraphen aus dieser Menge. Um triviale Unmöglichkeiten zu vermeiden, nehmen wir an, dass jeder Graph aus dieser Menge rooted ist, also einen Wurzelspannbaum als Teilgraph

beinhaltet. Für diese Klasse von Message Adversaries zeigen wir eine obere Schranke für den dynamischen Radius, welcher sich direkt auf die Lösbarkeit von Consensus auswirkt. Als nächstes betrachten wir die Klasse der abgeschlossenen Message Adversaries, welche auch die oblivious Message Adversaries inkludiert. Hierfür entwickeln wir eine präzise kombinatorische Charakterisierung für die Lösbarkeit von Consensus, welche sich auch als praktikabel herausstellt, da sie nur endliche Sequenzen von Kommunikationsgraphen benötigt.

Wir wenden uns dann der wichtigen Klasse der allgemeinen Message Adversaries zu, die "gute" Kommunikationsgraphen beliebig spät generieren dürfen und deswegen nicht abgeschlossen sind. Wir präsentieren einige scharfe untere Schranken mit passenden Lösungsalgorithmen für Message Adversaries, die auf Vertex-Stable Root Components basieren, einer speziellen Instanz von Stabilisierung. Wir entwickeln einen Consensus Algorithmus für einen Message Adversary mit rooted Kommunikationsgraphen und beweisen die Korrektheit dieses Algorithmus sowie seine Optimalität im Bezug auf die Dauer der Vertex-Stability. Danach untersuchen wir die Konsequenzen der Annahme, dass alle Kommunikationsgraphen rooted sind, und entwickeln einige damit in Zusammenhang stehende untere Schranken für das $k$-set agreement Problem, eine Verallgemeinerung von Consensus.

Schlussendlich richten wir unsere Aufmerksamkeit auf abstraktere Charakterisierungen der Lösbarkeit von Consensus für allgemeine Message Adversaries. Zunächst etablieren wir zu diesem Zweck Message Adversary Simulationen, welche einige Gemeinsamkeiten mit dem Failure Detector Formalismus aufweisen. Im letzten Kapitel etablieren wir eine vollständige Charakterisierung für die Lösbarkeit von Consensus unter allgemeinen Message Adversaries basierend auf Punktmengentopologie. Zu diesem Zweck konstruieren wir einen topologischen Raum, basierend auf der Ununterscheidbarkeit der Kommunikationsgraphsequenzen bzw. den korrespondierenden Executions eines Consensus Algorithmus. Wir zeigen, dass Consensus dann und nur dann lösbar ist, wenn dieser Raum in verschiedene Komponenten $PS(v)$ partitionierbar ist, welche aus den Executions, in denen $v$ entschieden wird, bestehen. Es stellt sich heraus, dass für abgeschlossene Message Adversaries diese Mengen kompakt und deshalb auch abgeschlossen sind, während sie für nicht abgeschlossene Message Adversaries nur relativ kompakt sind. In beiden Fällen zeigt sich jedoch, dass die Existenz dieser Partitionierung an die Broadcastability jeder einzelnen Komponente von $PS(.)$ geknüpft ist.

# Abstract

We study characterizations of consensus solvability in dynamic networks controlled by an omniscient message adversary. This model assumes a system of $n$ distributed agents, called processes, that execute a distributed algorithm and communicate by message passing in lock-step synchronous rounds. Communication in each round is subject to a message adversary, which determines which messages are successfully delivered and which are lost, encoded via a directed communication graph. In its most general form, the message adversary can be represented by the set of infinite sequences of communication graphs, one for each round, that it may generate.

Perhaps our most important theoretical result are a precise topological characterization of consensus solvability for general message adversaries and a considerably less abstract combinatorial characterization for the important class of closed message adversaries, which encompasses most of the message adversary classes for which consensus solvability characterizations existed so far. Thanks to this, we are able to state for the first time in a precise and rigorous way exactly what is the crucial property of an arbitrary message adversary that permits a consensus solution algorithm. Our arguably most important practical result is an optimal algorithm for dynamic networks that guarantee only brief stability phases along with its correctness and optimality proof. This algorithm is relatively simple and might be used as a template for solving consensus in real systems that exhibit massive transient message loss. In more detail, we present a host of impossibility results and solution algorithms for message adversaries of increasing generality:

First, we concern ourselves with oblivious adversaries, which are defined by a set of allowed communication graphs; the sequences it may generate are all possible combinations of communication graphs from this set. In order to circumvent trivial impossibility results, we assume that every graph from the set of allowed graphs is rooted, that is, contains a rooted spanning tree. For this class of message adversaries, we provide an upper bound for the dynamic radius, which has direct implications on the solvability of consensus. We then proceed to the larger class of closed message adversaries, which also contains oblivious message adversaries. We develop a precise combinatorial characterization of

consensus solvability, which is also practical in that it focuses on finite communication graph sequences only.

We proceed to an important class of general message adversaries, which supports a notion of "eventually something good will happen" and are hence non-closed. We present tight lower bounds and solution algorithms for message adversaries based on eventually vertex-stable root components, a particular form of eventual stabilization. We develop a consensus algorithm for message adversaries that generates rooted graphs only, which is optimal in terms of the duration of vertex-stability, along with its correctness and optimality proof. In addition, we study the consequences of dropping the requirement that all communication graphs are rooted, and derive some related lower bounds for the more general $k$-set agreement problem, a relaxed variant of consensus.

Finally, we turn our attention to more abstract characterizations of consensus solvability for general message adversaries. We first provide a notion, called a message adversary simulation, that shares some of the merits and generality of the failure detector formalism.

At last, we provide a complete characterization of consensus solvability for general message adversaries based on the topology of the execution space. For this purpose, we develop a topological space based on the indistinguishability of the executions and show that consensus is solvable if and only this space partitions into multiple components $PS(v)$ consisting of the executions with decision value $v$. For closed message adversaries, these sets are shown to be compact and hence closed, for non-closed MAs, they are only relative compact. In both cases, however, it turns out that the existence of this partitioning is tied to the broadcastability of every individual $PS(v)$, by some common process.

# Contents

Chapter 1

# Introduction

## 1.1 Dynamic Networks Overview

We study models, problems, and solutions for dynamic networks of distributed computing systems. Such dynamic networks arise when participants communicate by exchanging messages that may potentially get lost. We tend to think of these participants as small computing devices, which are aptly called "processes", however, this is not the point: The core issues that we investigate are those of local uncertainty due to incomplete information, which arise in every distributed environment, irrespective of the exact nature of the "processes". A related field is parallel computing, however there the focus is on how to distribute some computing task to multiple computing entities, typically coupled via shared memory, such as multiple cores of a processor, in order to perform the computation faster and more efficient. In contrast, distributed computing typically studies what can still be solved in spite of, and, what is impossible due to, faults occurring in a network-coupled system.

One classic failure model in distributed computing are process crashes, where, at some unknown point in time, a process may simply cease to operate. Crashes are usually permanent and are particularly insidious because of two reasons: First, a process may crash after sending a message only to a subset of the other processes, thereby creating local uncertainty of who still received its last message. Secondly, in an asynchronous system, where there are no bounds on the message delays or the time it may take a process to complete its computation, there is no way to locally distinguish a crashed process from a painfully slow one. In contrast, dynamic networks usually consider transient communication faults, where links may recover after being faulty. In this sense, crashes are almost a special case of a dynamic network, where at some point in time a process ceases to send any messages. We study this relation in more detail in Section 9.3.

Our focal point here is a special notion of dynamic networks, where communication is controlled by an entity, called message adversary. This paradigm conceptually assumes that every single message delivery is under the control of an adversary that tries to foil

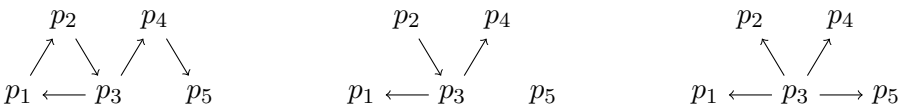


Figure 1.1: Communication graphs $G_1, G_2, G_3$ for 3 rounds.

the effort of the processes to solve a distributed computing task. If the adversary has a winning strategy that renders the task unsolvable, irrespective of the algorithm employed by the processes, we say that the task is impossible under the adversary. If, on the other hand, there exists an algorithmic winning strategy for the processes, we say the problem is solvable under the adversary. The main topic of this thesis are the circumstances under which a given distributed computing task is solvable under a message adversary and under which it is impossible. Clearly, every truly distributed task is impossible if the message adversary is not restricted in any way, since it may simply suppress all messages. Thus, we will usually assume an adequately restricted message adversary that, ideally, allows us to precisely pin down the relevant parameters for which solvability and impossibility can be established. A concrete example for this can be seen in Figure 7.2.

Most of the time, we will assume that the computation occurs in lock-step synchronous rounds (we will touch on asynchronous systems briefly in Sections 4.5 and 9.3). Each round consists of a phase of communication where every process attempts to broadcast its message to all other processes. It is here that the message adversary determines which messages are delivered and which are lost forever. After the communication phase ends, the processes perform some computation, which may depend on the messages that they just received as well as their current state. The rules of this computation are given as a deterministic algorithm, one for each process, which we will specify in pseudo-code for convenience. The communication in a given round, controlled by the message adversary, can be concisely expressed as a directed graph $G_r$, called the communication graph, where the set of vertices consists of the processes and an edge from $p$ to $q$ in the graph means that $q$ has received the message from $p$ in round $r$. As we shall see, an essential quality of a communication graph is whether it is *rooted*, i.e., whether it contains a rooted spanning tree. A given run, dictated by the message adversary, effectively consists of a sequence of communication graphs $(G_r)_{r>0}$, one for each round, called a communication pattern. It thus makes sense to describe a message adversary as a set of communication patterns, called the *admissible communication patterns*. Perhaps the most important insight about running deterministic algorithms in these kinds of systems is that the states of the processes in a given round are completely determined by the initial states of the process and the communication pattern generated by the message adversary.

An example for the first three rounds of a communication pattern can be seen in Figure 1.1. Even though there is no direct connection from $p_1$ to $p_5$ and $p_5$ is even disconnected most of the time, there is still an information flow from $p_1$ to $p_5$ in the following sense: If every process $p_i$ initially holds some piece of information, e.g. an integer $x_i$, and executes a

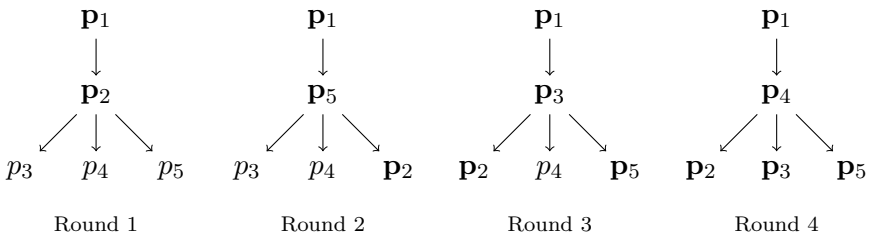


Figure 1.2: A communication pattern where it takes $O(n)$ rounds until some node broadcasts, despite a constant hop distance in every single graph (bold nodes represent processes that heard from $p_1$ after the depicted communication graph was applied).

simple store and forward algorithm, i.e., it stores and forwards all integers received so far in some set $S_i$, we have $x_1 \in S_5$ at the end of round 3. Clearly, $x_1$ is forwarded to $p_2$ in round 1, which in turn forwards it to $p_3$ in round 2. Finally, $p_5$ receives $x_1$ from $p_3$ in round 3.

It is important to realize that local properties of the communication graphs do not necessarily have direct implications on the communication pattern, even though this might seem plausible at first glance. For example, one may be tempted to conjecture that a maximum radius of 2 in every communication graph guarantees that all processes have heard from some process after a constant number of rounds. This is not the case, however: Consider, for example, the three-round communication pattern for processes $p_1, \dots, p_5$ shown in Figure 1.2. Herein, $G_1$ is a directed tree of height 3, with a single root node $p_1$ and a single node in the second level. In the following rounds, this second level node switches places with a new node from the third level. In this scenario, $p_4$ has not heard from $p_1$ by the end of round 3, even though the length of the path from $p_1$ to any other process is $\leqslant 2$ in every $G_r$. It is not hard to generalize this example to a set of $n$ processes $p_1, \dots, p_n$ to see that it takes actually $O(n)$ rounds until every process heard from $p_1$ even though every communication graph has a constant radius of 2.

Throughout this thesis, we focus on the classic distributed consensus problem, which we introduce in more detail later in Section 3.4. For now, it is interesting to note that in these problems each process is assigned an input value and, given an algorithm, the only variance in the initial states of the processes comes from different input value assignments. Therefore, each run of a deterministic distributed algorithm is actually completely determined by this input value assignment, together with the communication pattern. When investigating the execution space of a given message adversary, it therefore makes sense to consider a message adversary as the set of all possible input assignments combined with the admissible communication patterns.

## 1.2 Motivation

The human ability to invent models that explain our surroundings is perhaps the most distinguishing feature that sets us apart from the remaining known material world. This modeling can take many forms, most notably perhaps enabling our ancestors to survive in a deadly environment by outsmarting both predator and prey. More recently, the scientific method has led to astounding progress in human advancement, by providing ever more precise models for almost all aspects of our surroundings. Curiously, a significant amount of this effort is spent on better understanding the fundamental underpinnings of creations of our own design, a particular one of which, distributed computing systems, is the focus of this thesis. In this field, as in most of computer science, the focus is hardly on computing devices themselves but rather on the principles that guide their operation. Furthermore, even though the original motivation was to study engineered objects, the insights gained are sometimes profound enough to find applications well outside of the domain of computational and even artificial systems (see e.g. [LLT13, NBJ14, FH07, FK13]).

This thesis attempts to find a suitable model for distributed computing systems where communication is highly unreliable and investigates questions regarding the possibility/impossibility to still achieve agreement among the participants. We proceed by providing characterizations for ever more refined models, founding our understanding at a basic level and steadily increasing to more abstract and more encompassing models (see Section 1.3 for more details). The type of agreement that we investigate are the well-known consensus problem and one of its generalizations, $k$-set agreement. The ability to achieve agreement in a distributed computing system where communication is highly unreliable is interesting both from a theoretical and a practical perspective. For example, consensus is required for ensuring consistency in replicated databases, which are distributed across a number of servers, or for consistent mode switches e.g. in distributed reconfiguration. The interest in solving consensus in dynamic networks comes from the increasing pervasiveness of wireless and even mobile devices, which are subject to unreliable communication and energy constraints. A rigorous theoretical understanding of what can and cannot be computed in dynamic networks is, to some degree, necessary for their design and optimization.

**Mobility** Mobile ad-hoc networks [KM07] consist of highly mobile devices that move in a fashion that is hard to predict. This may incur that two participants that could communicate flawlessly with each other at one point in time due to their close proximity may not be able to directly communicate at all soon after if they have moved apart too far. Dynamic networks excel in expressing this issue, as the respective edge may simply disappear from one round to the next in the communication graph. In fact, the network topology may change completely from round to round.

**Directed Communication** A common paradigm in computer networks is to assume that links are bidirectional, in the sense that if $p$ hears from $q$ at some point in time,

then also $q$ hears from $p$. While this is certainly justified in wired "landline" networks, where the connection is present in both directions or not at all, ensuring this in a wireless setting comes at a price: localized fading or interference phenomena [SBB12, GKFS10] such as the capture effect or near-far problems [WJCD00] lead to communication that is inherently directed. Guaranteeing bidirectional communication here necessitates spending additional resources, which may be undesirable in many cases, especially if solutions can also be achieved with unidirectional forms of communication. Furthermore, there are instances such as disaster relief applications [LHSP11] where strong communication guarantees may simply not be achievable at all.

**Message Adversaries** A message adversary that can adapt to varying operating conditions and even to the behavior of some given algorithm is a very strong assumption, which offers several advantages: First and foremost, it allows to model systems that suffer from uncoordinated boot-up sequences or systems that must recover from massive transient faults: Wireless network connectivity can be expected to improve over time here, e.g., due to improving clock synchronization quality. Assuming a message adversary that behaves differently during stable and unstable periods is obviously advantageous in terms of coverage. A solution algorithm that can cope with such an adaptive adversary succeeds in every run, which makes the solution also suitable for safety-critical applications where failures would be catastrophic. Furthermore, such solutions also work if message delivery is vulnerable from a security point of view, i.e., if message loss may, at least to some extent, be in fact caused by a malicious entity that tries to attack the system.

**Synchronous Execution** Thanks to modern communication technology [SCS04], assuming the ability to implement lock-step synchronous execution of all processes is not unreasonable nowadays. As synchronized clocks are typically required for basic communication in wireless systems anyway, e.g., for transmission scheduling and sender/receiver synchronization, global synchrony is reasonably easy to achieve: It can be integrated directly at low system levels as in 802.11 MAC+PHY [IEE07], provided by GPS receivers, or implemented by means of network time synchronization protocols like IEEE 1588 or FTSP [MKSL04].

**Consensus** The consensus problem is a fundamental distributed computing problem that was proposed originally in [PSL80] and has since been studied thoroughly under many facets and in a great variety of models. One of the reasons why it is of such interest for computer science is state machine replication [Sch90] for fault-tolerance: the replicas' ability to solve consensus corresponds to their ability to reliably produce the same output. For instance, when requesting a withdrawal from a bank account, it makes sense that a query is sent to a database that is replicated across a number of servers in order to tolerate server failures. The replicas should hence reach consensus on whether to grant or deny the request, based on the state of their databases. Understanding the solvability of consensus in a given system is also beneficial for applications that require weaker versions

of the problem, such as approximate consensus, asymptotic consensus, $k$-set agreement or stabilizing consensus.

We hope that the above justifications for message adversaries have sufficiently piqued the curiosity of the reader to dive into the results presented in the following sections. The algorithms given may stimulate the development of more practical solutions for a real system that can be approximated with satisfactory accuracy by a given message adversary, and the characterization theorems allow answering the question whether distributed consensus can be solved in a dynamic network with certain communication guarantees at all.

## 1.3 Outline

This thesis is centered around the quest to find a characterization for consensus solvability under message adversaries that is as general as possible while still being fine-grained enough to be able to express a grand variety of constraints for the message adversary. As described in more detail below, we proceed by studying specific models and getting ever more general and abstract over the course of the thesis, thereby successively broadening our understanding of this question. We start by investigating the dynamic radius of a message adversary, which is a fundamental characteristic for consensus. Subsequently, we exploit this dynamic radius in a characterization for the special class of closed message adversaries, which have a nice combinatorial structure. After this, we continue to the structurally much more involved non-closed message adversaries by studying message adversaries based on the notion of eventual stability, instantiated by the notion of vertex-stable root components. Having obtained a deeper understanding of the intricacies of the non-closed message adversaries, we then attempt to replicate, in our model, the success of the failure detector formalism to express consensus solvability characterizations. Finding that the granularity of this approach is not very satisfying, we find at last a suitable characterization of general message adversaries, expressed as a condition on a topology of the execution space of the message adversary.

We now proceed with a more detailed description of the individual chapters of the thesis, an extended abstract of which appeared in

- [WS19] Kyrill Winkler and Ulrich Schmid. An overview of recent results for consensus in directed dynamic networks. *Bulletin of the European Association for Theoretical Computer Science EATCS*, (128):41–72, June 2019.

Most of this thesis is a result of various collaborations, as will be detailed below. Thus, throughout most of the thesis, I write “we” to mean myself as well as the respective collaborators that were instrumental in obtaining the results in their current form. Note that one of my collaborators, Manfred Schwarz, was also pursuing his doctoral degree at the time and thus some of our jointly obtained results that appear here, are presented also

in his thesis [Sch18]. A paragraph at the end of this section states the precise connection of our theses.

**Chapter 2: State of the Art** First, we give an overview of existing related work that is focussed on similar challenges as the ones posed throughout the thesis. Even though this list is hardly complete, it should give the reader an idea of our main sources of inspiration and provide a solid starting point for further exploration of the topic in general. Note that this chapter represents merely a broad overview of closely related literature on dynamic networks and characterizations for consensus solvability and that we give more specific connections to existing work at the beginning of each chapter.

**Chapter 3: Modelling Message Adversaries** In this chapter, we provide the formal model for dynamic networks and message adversaries that we employ throughout the remaining thesis. We present here only the most general aspects and notations of our model that are shared across the entire thesis, like rooted graphs. To avoid clutter, we do not introduce specific notation that is only required for contained subparts of the thesis but do so only in the relevant sections later on.

**Chapter 4: Dynamic Radius of Nonsplit Graphs** We commence our study with the fundamental task of information dissemination in Chapter 4. This represents a problem that is much more fundamental than consensus itself, yet both problems share a crucial connection that we will rely on throughout the entire remaining thesis. Information dissemination has been studied in several flavors in the literature and usually amounts to the question of how long it takes until every processes has heard from all the processes in the system. Since all-to-all communication is impossible in rooted dynamic networks in general, we restrict ourselves to the study of one-to-all communication, which serves as a lower bound for other distributed computing tasks. This analysis connects to an existing result that provides an interesting reduction of sequences of rooted communication graphs to sequences of nonsplit communication graphs, where each pair of nodes $p$, $q$ has a common incoming neighbor. We exploit this connection by deriving some new bounds regarding sequences of nonsplit graphs, leading to a new upper bound for guaranteed one-to-all communication. This chapter is based on the report

- [FNW19] Matthias Függer, Thomas Nowak, and Kyrill Winkler. On the radius of nonsplit graphs and information dissemination in dynamic networks. *CoRR*, abs/1901.06824, 2019.

**Chapter 5: Closed Message Adversaries** We then proceed to look at the class of closed message adversaries, which have a nice structure that makes them amenable to a concise and comparably easy consensus solvability characterization. We start by describing an existing characterization of consensus solvability from the literature, which shows, for an important subclass of closed message adversaries, under which conditions consensus is solvable. We then provide a characterization on our own for general closed

message adversaries, which relies on a notion of the kernel of a communication pattern, which shares a crucial connection to the dynamic radius from the previous chapter, as is apparent from the definition of the latter. This chapter is based on not yet published work.

**Chapter 6: Vertex-Stable Root Components** Searching for an even more general characterization, we leave the familiar terrain of closed message adversaries: Here, we consider message adversaries that guarantee that something good happens only eventually. These message adversaries are particularly interesting if one wants to express eventually stabilizing behavior, one instance of which, vertex-stable root components, is explored in depth. We show how this message adversary can be parameterized in such a way that allows us to formulate an exact boundary between solvability and impossibility of consensus in the case where all generated graphs are rooted. This entails algorithms that are optimal in the sense that they allow us to solve consensus for the least amount of necessary stability. This chapter is based mostly on the paper

- [WSS19] Kyrill Winkler, Manfred Schwarz, and Ulrich Schmid. Consensus in rooted dynamic networks with short-lived stability. *Distributed Computing*, Feb 2019

In addition to this, we present a tight lower bound on the duration of a vertex-stable root component that is required for solving consensus in Theorem 6.3.5, which (along with the related technical lemmas) is taken from

- [BRS+18] Martin Biely, Peter Robinson, Ulrich Schmid, Manfred Schwarz, and Kyrill Winkler. Gracefully degrading consensus and k-set agreement in directed dynamic networks. *Theoretical Computer Science*, 726:41–77, 2018.

**Chapter 7: Beyond Rootedness and Consecutive Stability** We proceed with message adversaries that do not generated rooted communication graphs in every round. We study different extents of partitioning of the communication graphs, such as only a constant fraction of the graphs being partitioned, a dominant connected component that eventually influences everything else, and even relax consensus to $k$-set agreement. With respect to the latter, we show that a straight-forward partitioning that does not look too bad at first glance already makes solving $k$-set agreement impossible. In order to do so, we use an impossibility proof technique from [BRS11] that reduces the impossibility of $k$-set agreement to a combination of a partitioning argument and an impossibility of consensus. While the first section of this chapter (Section 7.2) is new, the following sections are essentially a rewrite of

- [SWS16] Manfred Schwarz, Kyrill Winkler, and Ulrich Schmid. Fast consensus under eventually stabilizing message adversaries. In *Proceedings of the 17th International Conference on Distributed Computing and Networking*, ICDCN '16, pages 7:1–7:10, New York, NY, USA, 2016. ACM.

In addition to this, the last section (Section 7.6) on $k$-set agreement appeared in

- [BRS+18] Martin Biely, Peter Robinson, Ulrich Schmid, Manfred Schwarz, and Kyrill Winkler. Gracefully degrading consensus and k-set agreement in directed dynamic networks. *Theoretical Computer Science*, 726:41–77, 2018.

Preliminary versions of this result also appeared in

- [SWS+14] Manfred Schwarz, Kyrill Winkler, Ulrich Schmid, Martin Biely, and Peter Robinson. Brief announcement: Gracefully degrading consensus and $k$-set agreement under dynamic link failures. In *Proceedings of the 33th ACM SIGACT-SIGOPS Symposium on Principles of Distributed Computing*, PODC '14, pages 341–343, New York, NY, USA, 2014. ACM,
- [BRS+15] Martin Biely, Peter Robinson, Ulrich Schmid, Manfred Schwarz, and Kyrill Winkler. Gracefully degrading consensus and $k$-set agreement in directed dynamic networks. In *Revised selected papers Third International Conference on Networked Systems (NETYS'15)*, Springer LNCS 9466, pages 109–124, Agadir, Morocco, 2015. Springer International Publishing.

Some key ideas of Chapter 7, in particular, of Sections 7.3 to 7.5 also appeared in [Sch18], although they were presented there in a different fashion. We adapted them to match the framework of this thesis, and extended the results by Sections 7.2 and 7.6 and Lemma 7.3.4.

**Chapter 8: Faster and Simpler Consensus Algorithms** Until now, we were not concerned with the performance of our algorithms. Since the termination time is a crucial performance criterion for any consensus algorithm, however, we proceed to investigate this aspect in particular and develop a simple algorithmic technique to solve consensus under the message adversary from Chapter 6 with an algorithm that has an asymptotically optimal termination time. We show that, surprisingly, most of the computation can actually performed in a centralized, de facto *a priori* manner, reducing the complexity of the algorithm significantly. This chapter is based on not yet published work.

**Chapter 9: Message Adversary Simulations** Getting back on track for our search for a more general consensus characterization, we investigate how to apply failure detectors to our model. Chapter 9, thus investigates the connections of our solvability characterizations of consensus and $k$-set agreement to the well-established failure detector formalism. We show that an analogous notion, which we call message adversary simulations, can be defined for message adversaries. While it provides us with the interesting ability to reduce every message adversary that permits a consensus or $k$-set agreement algorithm to a much simpler adversary, regarding a solvability characterization, this notion is somewhat coarse and lacks the finer granularity of the characterizations from

the previous chapters. We did a first investigation of this topic already in [BRS+18], which was later refined in

- [SSW18] Ulrich Schmid, Manfred Schwarz, and Kyrill Winkler. On the strongest message adversary for consensus in directed dynamic networks. In Zvi Lotker and Boaz Patt-Shamir, editors, *Structural Information and Communication Complexity*, pages 102–120, Cham, 2018. Springer International Publishing.

Most results of Chapter 9 appear also in [Sch18]. Our presentation here differs from the one in [Sch18] in that it represents a thorough revision of [SSW18] with more clear and concise definitions and proofs. Furthermore, we strengthened the original result by showing the equivalence of binary consensus and multi-valued consensus in our model.

**Chapter 10: The Execution Space of a Message Adversary** Finally, we present a precise consensus solvability characterization for general message adversaries that encompasses closed as well as non-closed message adversaries. We accomplish this by looking at the topology of the execution space of a message adversary. For this purpose, we develop a pseudo-metric and a pseudo-semi-metric that are based on the indistinguishability of two executions (respectively their underlying process-time graphs) for some processes. The resulting topological space leads us to a characterization of consensus solvability in terms of the connectedness of the subspaces corresponding to conflicting valences. We show that this property, for compact as well as non-compact message adversaries, equivalently means that all $v$-valent connected components must be broadcastable, i.e., have a finite dynamic radius. This chapter is based on

- Thomas Nowak, Ulrich Schmid, and Kyrill Winkler. Topological characterization of consensus under general message adversaries. In *Proceedings of the 2019 ACM Symposium on Principles of Distributed Computing*, PODC '19, pages 218–227, New York, NY, USA, 2019. ACM

**Chapter 11: Concluding Remarks** Finally, we conclude with a collection of remarks that summarize our main accomplishments in a more technical manner and present a collection of open questions.

**Summary of relation to [Sch18]:** Note that a preliminary version of parts of Sections 7.3 to 7.5, Chapter 9, and Theorem 6.3.1, as well as elements from the respective models, appeared already in the thesis of Manfred Schwarz [Sch18], with whom I collaborated on these questions. Below, I summarize the results that we both included for the sake of completeness and convenience of the reader. In addition, I will explicitly mark the according results where they appear in the later chapters.

While Theorem 6.3.1 (and its proof) is essentially the same as [Sch18, Theorem 88], Chapter 7 has been extended by Section 7.2, a lower bound (Lemma 7.3.4), and completely

rewritten in the framework of the remaining thesis. Chapter 9 of the present thesis is a thoroughly revised, streamlined version of [Sch18, Chapter 7], which was published in [SSW18]: Aside from a major rewrite that lead to more clear and concise definitions and proofs, it presents a proof for the equivalence of binary consensus and multi-valued consensus in the model, which leads to a stronger and more general result.

## 1.4 Structure of the Thesis

The thesis is presented in a bottom-up fashion, which starts from low-level concepts and simple message adversaries, steadily getting more abstract and general. Chapter 4 investigates how long it takes until the kernel of a communication pattern, a fundamental key characteristic for consensus solvability for every message adversary that we will encounter frequently throughout the thesis, becomes non-empty. Chapter 5 subsequently studies consensus solvability under closed message adversaries, a restricted class of message adversaries that are amenable to a detailed combinatorial analysis. We go beyond closed message adversaries in Chapter 6, where we study an important class of stabilizing message adversaries that are not closed and give an optimal algorithm. After taking a detour in Chapter 7, where we look at what happens when rootedness, an assumption commonly made for message adversaries, is dropped and in Chapter 8, where we take a look at a more efficient algorithm, we arrive at our first characterization of consensus solvability using message adversary simulations in Chapter 9. Finally, the thesis culminates in Chapter 10, which gives a general and complete topological characterization of consensus solvability for arbitrary message adversaries.

Each of these chapters contains a short outline, followed by an overview of specific related work and the basic definitions, notations, and terminology required for the technical parts of the chapter. The algorithms, theorems and proofs constitute the main body of each chapter. All the above chapters depend on the model introduced in Chapter 3 and can be read independently of each other, except perhaps for Chapter 7, which uses variants of the message adversaries introduced in Section 6.2 and is based in part on the algorithms of Section 6.4. Furthermore Chapter 8 contains an algorithm for a message adversary that was already introduced in Section 6.2. We have taken care, however, to make this chapter reasonably self-contained, thus it should be reasonably readable on its own as well.

Chapter 2

# State of the Art

We now give an overview of general related work for this thesis. In addition to this, we provide a brief section at the beginning of each chapter that mentions more specific related work that is relevant in the context of the respective chapter and we found to be too specialized to be mentioned already here.

## 2.1 Consensus Lower Bounds

Consensus solvability characterizations for distributed computing systems have existed for a long time, at least since the seminal work [PSL80], where it was revealed that reaching agreement in a synchronous system with $n$ byzantine agents requires at least $3n + 1$ agents in general, respectively $2n + 1$ if digital signatures that are sufficient to protect against random failures, are available. Later on, the highly influential result [FLP85] showed the impossibility of consensus in an asynchronous system even if just a single crash failure may occur. In more detail, it was shown there that every potential solution algorithm has an admissible, forever undecided run. The way this was proven is of particular relevance to this thesis, as it introduced the now-standard bivalence proof technique, which essentially shows that there is a bivalent initial configuration and each bivalent configuration has a bivalent successor configuration. We will use this formalism heavily for our own impossibility proofs, albeit in a slightly less subtle fashion. That is to say, the proof of [FLP85] provides a very strong induction step, which essentially shows that the bivalent successor configuration can be reached even if an arbitrary receive event is scheduled last. This implies that even a simple round-robin schedule contains a perpetually bivalent run and thus ensures that the resulting forever undecided run is indeed admissible.

In addition to this, it is shown in [FLP85] that a majority of correct processes suffices to solve consensus in an asynchronous system if processes are either correct or initially dead. The proposed solution algorithm employs a communication graph approximation, in which an initial clique, without any incoming edges from outside of the clique, can be

found. Much of our employed methodology from Chapter 6 is closely related to this idea and may even be seen as a translation of this principle to the message adversary model.

The impossibility of asynchronous consensus with a single faulty process from [FLP85] has been generalized in several directions. In [BMZ90], the technique from [FLP85] was translated from consensus to decision tasks in general, for which a precise characterization was given. This characterization rests on the notion of output vectors and their graph. Briefly summarized and with glossing over many of the details, an input, resp. output, vector is an $n$-dimensional vector where the $i$-th component represents the input, resp. output, of $p_i$. The graph of the output vectors is the graph that results when two output vectors that differ in precisely one component are considered adjacent. A partial input/output vector is a vector that is missing one component and the extension of a partial vector replaces this missing component in such a way that the result is a valid input/output vector according to the task. Finally, a partial output vector covers a partial input vector if every extension of the input vector has corresponding extension of this output vector. The core result from [BMZ90] is that a decision task is solvable despite a single crash failure if, and only if, the graph of the decision vectors is connected and, for every partial input vector, a covering decision vector exists and can be computed.

In [LM95], it was shown that bivalence arguments can be readily applied to closed schedulers, a special case of asynchronous distributed computing systems that has a nice structure and is amenable to a clean analysis. One of the crucial properties of closed schedulers that is identified in [LM95] is the following: given a predicate that, for every admissible schedule, holds in a finite prefix of said schedule, we can find a bound $M_c$, such that this predicate holds in every prefix (of an admissible schedule) of length $M_c$. The authors proceed to show that several consensus impossibility results for established distributed computing models can actually also be derived in a closed scheduler. The underlying principle of closed schedulers is highly relevant to several parts of this thesis, in particular to Chapter 5, where we identify an analogous paradigm for message adversaries.

The above work was generalized in [MR02], where layerings are defined in terms of the very general *runs and systems model* of [FMHV03]. Roughly speaking, a layering is a highly structured run that adheres to the system's fairness constraints, respectively its admissibility requirements, but violates the consensus specification. The general applicability of almost the same layering reveals striking similarities between the source of consensus impossibilities in ostensibly quite different system models. This leads to two kinds of bivalence-style impossibility proofs: The first is based on a permutation layering, which generalizes the argument from [FLP85], yet does not rely on the notion of a critical state. The second constructs a synchronic layering that imitates the behavior of a synchronous model where consensus is impossible.

## 2.2 Consensus Algorithms

A large part of this thesis (in particular, most of Chapters 6 to 9) is dedicated to message adversaries that support a notion of "eventually something good happens". This is diametrically opposed to a closed scheduler (and even to the first mention of message adversaries in [AG13]) in that, by construction, not everything in the closure is admissible: "Eventually" means something happens in finite but unbounded time, i.e., the "something good" may be delayed arbitrarily long, but not forever. Consensus in systems that adhere to this (usually) more complex paradigm has been studied extensively in the past. Some classical results in this area include [DLS88, CT96, Lam98], where processes may exhibit byzantine behavior and operate in a partially synchronous environment.

[DLS88] provides consensus algorithms and lower bounds under a great variety of combinations of failure and synchrony models, such as eventually bounded message delivery (where this bound itself may be known or unknown to the processes), with processes prone to byzantine behavior or crashes.

In [CT96], the partially synchrony assumptions are encapsulated into a single abstract object, called the eventually strong failure detector $\Diamond\mathcal{S}$. This failure detector is an oracle, whose output is a list of processes that are suspected to have crashed, such that, eventually, $\Diamond\mathcal{S}$ suspects all faulty processes and, furthermore, there exists a correct process that is eventually not suspected by $\Diamond\mathcal{S}$ anymore. One of the algorithms presented in [CT96] shows that a rotating coordinator algorithm can be employed to solve consensus with $\Diamond\mathcal{S}$.

In [Lam98], the PAXOS algorithm is presented, which shows how consensus can be solved in spite of timing violations and process crashes, as long as a fault-free period happens eventually. It even tolerates process restarts, provided that the processes have access to a persistent storage. PAXOS is optimized for the fault-free case, motivated by practical systems where failures are rare.

All of the above solution rely on a majority of alive processes, resp., an overwhelming majority of non-byzantine processes.

The fundamental difference of these works to the message adversary model is that a message adversaries is specified directly with respect to the communication graphs and the resulting dynamically evolving network topology. This allows for a (much) more fine-grained formulation of the message delivery throughout the run and, in particular, the "something good" period, which, in [DLS88, Lam98], is assumed to be an execution fragment where no failures or timing violations occur.

## 2.3 Information Dissemination

Information dissemination among an ensemble of $n$ participants is a fundamental question that has been studied in a grand variety of settings and flavors (see [FL94, HLPR13, HHL88, HKMP96] for various reviews on the topic). While traditional approaches

usually assume a static underlying network topology, with the rise of pervasive wireless devices, more recently, the focus has shifted to dynamically changing network topologies [KLO10]. A useful way of viewing the distribution of information is to denote the pieces of information that should be shared among the participants as tokens. For instance, the all-to-all token dissemination problem investigates the complete dissemination of $n$ initially distributed tokens. This problem was studied in [KLO10] with a focus on bounds for the time complexity of the problem, i.e., how long it takes at least, resp. at most, until $n$ tokens have been received by everyone. Here, the participants employed a token-forwarding algorithm mechanism, where tokens are stored and forwarded but not altered.

## 2.4 Message adversaries

Studying consensus in synchronous message passing systems subject to mobile link failures dates back at least to the seminal paper [SW89] by Santoro and Widmayer; generalizations have been provided in [SWK09, CBS09, BSW11, CGP15, CBFN15]. In all these papers, consensus, resp. variants thereof, are solved in systems where, in each round, a digraph is picked from a set of possible communication graphs. The term *message adversary* was coined by Afek and Gafni in [AG13] for this abstraction, which was later refined to *oblivious message adversary* in [CGP15].

A different approach for modeling dynamic networks has been proposed in [KLO10]: $T$-interval connectivity guarantees a common subgraph in the communication graphs of every $T$ consecutive rounds. [KOM11] studies agreement problems in this setting. Note that solving consensus is relatively easy here, since the model assumes bidirectional and always connected communication graphs. In particular, 1-interval-connectivity, the weakest form of $T$-interval connectivity, implies that all nodes are able to reach all the other nodes in the system.

Researchers have also developed several "round-by-round" frameworks, which allow to relate models of computation with different degrees of synchrony and failures. Examples are round-by-round fault detectors by Gafni [Gaf98], the GIRAF framework by Keidar and Shraer [KS06], the HO model by Charron-Bost and Schiper [CBS09], the time-varying graph framework by Casteigts et al. [CFQS12], and the hybrid failure model by Biely et al. [BSW11].

### 2.4.1 Oblivious message adversaries

Perhaps one of the earliest characterizations of consensus solvability in distributed computing systems prone to communication errors is [SW89], where it is shown that consensus is impossible if up to $n-1$ messages may be lost each round. The reason for this was identified to be that communication patterns in this case are *adjacency-preserving* and *continuous*, the combination of which allows an inductive construction of a perpetually bivalent execution, i.e., an execution where consensus can never be solved, independent

of the solution algorithm employed. This proof is in the spirit of the classic consensus impossibility proof from [FLP85], even though in the latter, the construction of a forever bivalent execution that is also admissible is harder and necessitates somewhat even more subtle arguments.

The term "oblivious message adversary" for a message adversary that can choose the communication graphs in every round from a fixed set of graphs was coined in [CGP15]. In this terminology, the adversary from [SW89] may, each round, pick any graph from the set that contains all communication graphs where $n-1$ or fewer edges are missing. In [CGP15], a property of an equivalence relation on the sets of communication graphs was identified, which captures exactly the source of consensus impossibility in the oblivious setting. It was shown how this property can be used to find a necessary and sufficient condition for solving consensus. While this result was certainly a major inspiration for our work, to the best of our knowledge there is no way to generalize it directly to non-oblivious message adversaries. Nevertheless, it provides an exact characterization for consensus solvability under oblivious message adversaries, thereby answering this question and providing a significant generalization of the classic result from [SW89]. As stated above, the latter can be seen as a special case of [CGP15], even though we should note that the agreement problem considered in [SW89] was more general than consensus. Due to its close relation to this thesis, in particular to Chapter 5, we describe [CGP15] in more detail in Section 5.2.

### 2.4.2 Non-oblivious message adversaries

In the world of oblivious message adversaries, as well as in the original notion of message adversaries from [AG13], there is no notion of eventually stabilizing behavior of dynamic networks. One instance where such stabilizing behavior is described, is the message adversary that guarantees eventually stable root components, considered in [BRS12].

For the special case of $n = 2$, [FG11] provides a complete characterization of consensus solvability for message adversaries that are not oblivious: Using a bivalence argument, it is shown that certain graph sequences (a "fair sequence" or a special pair of "unfair sequences") must be inadmissible to render consensus solvable, and provided a universal algorithm for this case. However, to the best of our knowledge, a complete characterization of consensus solvability for arbitrary system sizes and general message adversaries did not exist.

## 2.5 Failure Detectors

There is a large body of existing work on relations between different distributed computing models. The implementability of failure detectors in timing-based models of computation has already been addressed in Chandra and Toueg's seminal work [CT96], and received quite some attention in the chase for the weakest system model for implementing $\Omega$ [ADGFT01, ADGFT03, ADGFT04, MR99, AFM$^+$04, MOZ05, HMSZ09, FR10].

There are also several papers that establish more abstract relations between various synchronous models and asynchronous models with failure detectors. For example, Charron-Bost, Guerraoui and Schiper showed that the partially synchronous model [DLS88] with $\Phi = 1$ and $\Delta \geqslant 1$ is not equivalent to the asynchronous model with perfect failure detectors in terms of problem solvability. Rajsbaum, Raynal and Travers showed in [RRT08] that failure detectors do not increase the solution power of the iterated immediate snapshot model over asynchronous read/write shared memory. A more general relation between various eventually synchronous models and asynchronous models with stabilizing failure detectors, which also considers efficiency of the algorithmic transformations, has been established by Biely et. al. in [BHDPW07]: Among other results, it establishes that $\Omega$ is essentially equivalent to models with an eventually timely source, as well as to eventual lock-step rounds. [JT08, CBHW10] shed some light on the limitations of the failure detector abstraction in timing-based models of computation, which are relevant in our context.

## 2.6 Point-set topology

Regarding topological methods, one has to distinguish point-set topology, as introduced in [AS85], and combinatorial topology, as used for proving theorems characterizing solvable tasks in wait-free asynchronous shared memory systems with process crashes[1] [HS99] or under fair adversaries [KRH18], for example. Combinatorial topology studies the topology of the reachable states of admissible executions, captured by simplicial complexes as shown in Figure 2.1 (left), and has been developed into a widely applicable tool for the analysis of distributed systems [HKR13]. By contrast, the primary objects of point-set topology are infinite executions, as shown in Figure 2.1 (right), where closed and dense sets precisely characterize safety and liveness properties, respectively.

[1] Note that message adversaries do not fall into this category of models.

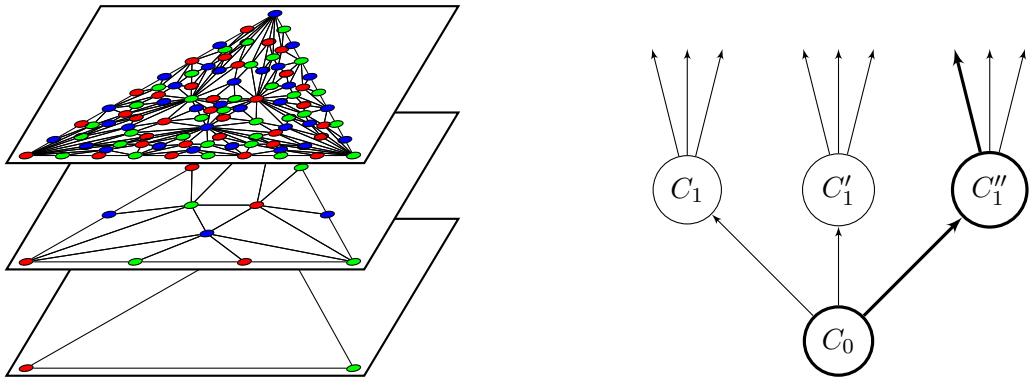


Figure 2.1: Comparison of the combinatorial topology approach and the point-set topology approach: The combinatorial topology approach (left) studies sequences of increasingly refined spaces in which the objects of interest are simplices (corresponding to configurations). The point-set topology approach (right) studies a single space in which the objects of interest are executions (i.e., infinite sequences of configurations).

Chapter 3

# Modelling Message Adversaries

We now provide an overview over the different basic models and formalisms that are employed throughout the thesis. Whereas specialized formalisms that remain relevant mostly throughout a single chapter are introduced at the start of that respective chapter, we give here a general overview of the underlying formal model that is common to all chapters. Our presentation includes a variety of different notations that have been introduced in the literature and, where appropriate, we elaborate how they are connected to each other.

The granularity of the formalism that we choose to model a distributed computing system in this thesis follows perhaps most closely the spirit of the books [Lyn96, AW04]. That is, we use a model for the operation of an ensemble of distributed processes with a clear focus on the uncertainty that results from the incomplete information available at a single process. As the topic of this thesis are message adversaries or, more generally speaking, the issues that arise due to unreliable message delivery, we usually assume that the computation of the processes happens instantaneously and that the processes have an unbounded amount of memory and may send arbitrarily large messages. We will see that often, even with such (unrealistically) powerful machines, there is no hope of succeeding against a powerful message adversary. With respect to our solution algorithms, albeit that our focus is usually on their existence and correctness and not on their efficiency, we will discuss how they can be optimized when appropriate (see in particular Chapter 8).

The mathematical tools that we employ throughout the thesis are mostly consistent across all chapters. We will use graph theory, however we will require only elemental knowledge of the topic, like connectedness, directed paths, and induced subgraphs, as we introduce all specialized terminology later in this chapter. We will employ some basic combinatorial concepts such as the pigeonhole principle and use only elementary set theory (except perhaps in Chapter 10) and fundamental rules from predicate logic, such as contraposition. Our proofs will most often proceed by contradiction or by induction. In order to prove the correctness of an algorithm, we will usually define invariants and, using an inductive argument, prove that they hold for an arbitrary round. In order to

show the impossibility to solve consensus under a message adversary, we have to show that no correct solution algorithm exists. In this respect, we are often inspired by the now-standard methodology from [FLP85], in particular the application of bivalence proofs or similar techniques, which we often establish by means of a partitioning argument. In Chapter 5 we go slightly beyond this by looking at the infinitely long limit executions directly. Finally, we will make use of point-set topology in Chapter 10, where we take care to (briefly) introduce all the employed notions and concepts at the beginning of the chapter.

**Chapter organization:** We will start our exposition with the fundamental computational model in Section 3.1. There, we will describe the operation of a distributed algorithm in the message adversary model and formalize the basic terminology of communication graphs, communication patterns, and message adversaries. Subsequently, in Section 3.2, we introduce the specialized graph-theoretic terminology that we will encounter throughout the entire thesis. It is here that we first encounter the notion of a root component, which, as we will see in the subsequent chapters, plays a crucial role when solving the consensus problem. In Section 3.3, we describe our notion for information propagation in communication patterns. This leads to the pivotal concept of indistinguishability of communication patterns, which will be essential for our impossibility proofs throughout the remaining thesis. The $k$-set agreement problem and the consensus problem as its special case, along with the formalization of the executions of a distribute algorithm and their indistinguishability, will be introduced in Section 3.4. Finally, we describe process-time graphs, a very powerful object that encapsulates all the properties of interest of a concrete consensus algorithm execution in the message adversary model in Section 3.5.

## 3.1 Distributed Computation

We consider an ensemble of deterministic state machines, called *processes*, which communicate via message passing over unreliable point-to-point links. Processes have unique identifiers and are typically denoted by $p, q, p', q'$, etc. The operation proceeds in lock-step synchronous rounds $r = 1, 2, 3, \dots$ consisting of a phase of message exchange between the processes, which is followed by a phase of local computations. Similar to, e.g., [KOM11], we use the convention that all operations of round $r$ take place strictly within time $r - 1$ and time $r$, which results in well-defined and stable states of all processes between the rounds: The state of a process at time $r$ is its initial state (specifying the starting values for each variable) for $r = 0$, respectively the state at the end of its round-$r$ computation (describing the content of all variables as well as the messages to be sent) for $r > 0$.

A *dynamic graph* is a mapping $\sigma : \mathbb{N}^* \to G$ of each round $r$ to a directed graph[1] $G_r = \langle \Pi, E^r \rangle$, called the round-$r$ communication graph. Each node of $\Pi$ represents a

[1] We sometimes write $p \in G_r$ instead of $p \in \Pi$ to stress that $p$ is a vertex of $G_r$, and sloppily write $(p, q) \in G_r$ instead of $(p, q) \in E^r$.

process, and an edge $(p, q)$ in $G_r$ represents that the round-$r$ message of $p$ sent to $q$ is received by $q$ in the same round. Since every process $p$ always successfully receives from itself, all graphs $G_r$ are reflexive, i.e., they contain an edge $(p, p)$ for every process $p \in \Pi$. The *in-neighborhood* of $p$ in $G_r$, $\mathrm{In}_p(G_r) = \{q \mid (q, p) \in G_r\}$, hence represents the processes whose message to $p$ in round $r$ is not lost. Similarly, the *out-neighborhood* of $p$ in $G_r$, $\mathrm{Out}_p(G_r) = \{q \mid (p, q) \in G_r\}$, are those processes that did receive a message sent by $p$ in round $r$. We stress that the vertex set $\Pi$ of a given dynamic graph is fixed (but usually not known to the processes) and only the set of edges may vary from round to round and assume that every $p \in \Pi$ has a unique identifier from the set $[1, |\Pi|]$. Most of the time, we will interpret a dynamic graph $\sigma(r)$ as an infinite sequence $\sigma$ of consecutive communication graphs and denote its vertex set by $\Pi_\sigma$. When describing a continuous subsequence $\sigma'$ of $\sigma$, ranging from round $a$ to round $b$, we denote this as $\sigma' = (G_r)_{r=a}^{b}$, where $|\sigma'| = b - a + 1$, with $b = \infty$ for infinite subsequences. We slightly overload notation and write $\sigma' \subseteq \sigma$ if $\sigma'$ is a continuous subsequence of $\sigma$ and write the concatenation of two successive subsequences $\sigma'$, $\sigma''$ as $\sigma' \circ \sigma''$. We call the special subsequence of $\sigma$ that consists of its first $r$ communication graphs the round $r$ prefix of $\sigma$ and denote it using $\sigma|_r$ and call $\sigma$ an extension of $\sigma|_r$.

A *message adversary* $\mathsf{MA}$ that may suppress certain messages in an attempt to foil the collaboration of the processes is at the core of our model. Formally, it is a set of dynamic graphs, or, equivalently, sequences of communication graphs, which are called the *admissible communication patterns*. Sometimes it will be convenient to denote explicitly restrictions on the size of the vertex set of the dynamic graphs of a message adversary as the first index of $\mathsf{MA}$. For example, $\mathsf{MA}_n$ states that every dynamic graph of $\mathsf{MA}_n$ has a vertex set of size exactly $n$, while $\mathsf{MA}_{\leqslant n}$ denotes that this size is at most $n$. Conceptually,[2] we assume that processes know *a priori* the specification of the message adversary, hence an algorithm that succeeds under $\mathsf{MA}$ must be able to cope with the size restrictions of $\mathsf{MA}$. Since a message adversary is a set of dynamic graphs, we can compare different message adversaries via set inclusion.

## 3.2 Dynamic Graph Structures

First, we introduce the pivotal notion of a *root component* $R$, often called root for brevity, which denotes the vertex set of a strongly connected component of a graph where there is no edge from a process outside of $R$ to a process in $R$.

**Definition 3.2.1** (Root Component)**.** *$R \neq \emptyset$ is a root (component) of graph $G$, if it is the set of vertices of a strongly connected component $\mathcal{R}$ of $G$ and $\forall p \in G, q \in R : (p, q) \in G \Rightarrow p \in R$.*

It is easy to see that every graph has at least one root component. A graph $G$ that has a *single* root component is called *rooted*; its root component is denoted by $\mathrm{Root}(G)$.

[2] As we will see e.g. in Section 6.4, the processes often require only knowledge of some key parameters.

Clearly, a graph $G$ is rooted if and only if contracting its strongly connected components to single vertices yields a rooted tree. Hence, $G$ is weakly connected and contains a directed path from every node of $\mathrm{Root}(G)$ to every other node of $G$.

Conceptually, root components have already been employed for solving consensus a long time ago: The asynchronous consensus algorithm for initially dead processes introduced in the classic FLP paper [FLP85] relies on a suitably constructed initial clique, which is just a special case of a root component.

In order to model stability, we rely on root components that are present in every member of a (sub)sequence of communication graphs. We call such a root component the *vertex-stable root component* of a (sub)sequence and stress that, although the set of processes remains the same, the interconnection topology between the processes of the root component as well as the connection to the processes outside may vary arbitrarily from round to round.

**Definition 3.2.2** (Vertex-Stable Root Component)**.** *We say that a non-empty sequence $(G_r)_{r\in I}$ of graphs has a vertex-stable root component $R$, if and only if each $G_r$ of the sequence is rooted and $\forall i, j \in I : \mathrm{Root}(G_i) = \mathrm{Root}(G_j) = R$. We call such a sequence an $R$-rooted sequence.*

We would like to clarify that while "rooted" describes a graph property, "$R$-rooted" describes a property of a sequence of graphs.

As is easily checked, Definition 3.2.1 applies also to graphs that are not rooted: here $\mathrm{Root}(G)$ yields a collection of sets. In order to stress that the underlying graph is not rooted in this case, we call the members of this collection the *source components* of $G$, written as $\mathrm{Sources}(G)$. It is not hard to see that each node in $G$ has a path from at least one source component. Given some arbitrary ordering on the source components of $G$, it is hence possible to assign to each node $p$ of $G$ the unique source component that can reach $p$ that is the smallest wrt. this ordering. In this sense, every node can be mapped to a unique source component and the graph $G$ can hence be seen as partitioned into multiple connected components, one for each source component. This partitioning of a communication graph $G$ will prove useful when talking about communication graphs that are not rooted in Chapter 7.

**Definition 3.2.3.** *Let $G$ be a communication graph and $R_1, \ldots, R_k$ the source components of $G$. The partitioning $P = P_1, \ldots, P_k$ of $G$ wrt. its source components is defined as the induced subgraph $P_i = G[V_i]$ where $V_i = \{v \in G \setminus \bigcup_{j=1}^{i-1} P_i : v \text{ is reachable from } R_i\}$.*

Below we show that the partition from Definition 3.2.3 does make sense for communication graphs with multiple source components. In essence, we see that each set of the partition is determined by a source component of the communication graph.

**Corollary 3.2.4.** *The partitioning $P$ of $G$ has the same source components as $G$ and two distinct source components do not occur in the same connected component in $P$.*

*Proof.* For the first property, we observe that $P$ is a subgraph of $G$. By Definition 3.2.1, the nodes of a source component are reachable only by other nodes of the same source component. It follows all the source components of $G$ occur also in $P$. For the converse, suppose there is a source component $R$ in $P$ that is not in $G$. Let $j = \min\{i : R \text{ is reachable from } R_i \text{ in } G\}$. It follows from the minimality of $j$ that there is a path in $G$ such that every node on this path is reachable from $R_j$ and no $R_k$ with $k < j$. But then $R$ is reachable from $R_j$ in $P$, and $R$ is thus no source component in $P$ by Definition 3.2.1.

For the second property, suppose that a node $v$ in $P$ is reachable from two distinct source components $R, R'$ of $P$. Let $j = \min\{i : v \text{ is reachable from } R_i \text{ in } G\}$. By the minimality of $j$ and by Definition 3.2.3, $v$ is reachable in $P$ from the source component $R_j$ and no other source component, a contradiction to the initial supposition. □

## 3.3 Information Propagation

Given two graphs $G = \langle V, E\rangle$, $G' = \langle V, E'\rangle$ with the same vertex set $V$, let the *product graph* $G \circ G' := \langle V, E''\rangle$ where $(p, q) \in E''$ if and only if there exists a $p' \in V$ such that $(p, p') \in E$ and $(p', q) \in E'$.

In order to model information propagation in the network, we use a notion of *causal past*: Intuitively, a process $q$ is in $p$'s causal past, denoted $q \in \mathrm{CP}_p^r(r')$ if, at time $r$, $p$ holds information (sent either directly or transitively, via intermediate messages) that $q$ sent after time $r'$. This is closely related to various concepts that have been introduced in the literature (cf. for example [CFQS11] and the references therein), such as the heard-of sets from [CBS09], the temporal distance from [XFJ03], or the past set from [KOM11].

**Definition 3.3.1** (Causal past)**.** *Given a sequence $\sigma$ of communication graphs that contains rounds $a$ and $b$, the causal past of process $p$ from time $b$ down to time $a$ is $\mathrm{CP}_p^b(a) = \emptyset$ if $a \geqslant b$ and $\mathrm{CP}_p^b(a) = \mathrm{In}_p(G_{a+1} \circ \cdots \circ G_b)$ if $a < b$.*

A useful fact about the causal past is that in full-information protocols, where processes exchange their entire state history in every round, we have $q \in \mathrm{CP}_p^r(s)$ if and only if, at time $r$ (and hence thereafter), $p$ knows already the state of $q \neq p$ at time $s$.

To familiarize the reader with our notation, we introduce the following technical Theorem 3.3.2, which shows a structural property of our model using similar basic arguments as in [KLO10]. It describes the information propagation in a graph sequence consisting of a sequence $G = (G_{r_i})_{i=1}^n$ of not necessarily consecutive communication graphs on the same vertex set $\Pi$, where all $G_{r_i}, G_{r_j} \in G$ are rooted but $\mathrm{Root}(G_{r_i})$ is not necessarily the same as $\mathrm{Root}(G_{r_j})$. As we have mentioned earlier, every $G_{r_i} \in G$ contains a rooted spanning tree and is therefore weakly connected. In essence, the lemma shows that, by time $r_n$, each process $p$ (except one process that is in the root component of $G_{r_n}$) transitively received a message from a process $q$ that was sent after $q$ was member of a root component of a graph of $G$.

**Theorem 3.3.2.** *Let $G = (G_{r_i})_{i=1}^{n}$ be a sequence of not necessarily consecutive, rooted communication graphs on the same vertex set $\Pi$ where $|\Pi| = n > 1$. Pick an arbitrary mapping $f\colon [1,n] \to \Pi$ s.t. $f(i) \in \mathrm{Root}(G_{r_i})$. Then $\forall p \in \Pi \setminus \{f(n)\}$, $\exists i \in [1, n-1]$: $f(i) \in \mathrm{CP}_p^{r_n}(r_i)$.*

*Proof.* Let $S(i) = \{p \in \Pi \mid \exists j \in [1,i]\colon p = f(j) \vee f(j) \in \mathrm{CP}_p^{r_i}(r_j)\}$ be the set of nodes that, by time $r_i$, received the state of $f(j)$ at time $r_j$ or are equal to $f(j)$. We show by induction that $|S(n)| \geqslant n$.

The base $|S(1)| \geqslant 1$, follows because $f(1) \in S(1)$.

For the step from $i$ to $i+1$, we have the hypothesis $|S(i)| \geqslant i$. Since $S(i) \subseteq S(i+1)$, we only need to consider the case $|S(i)| = i < n$. Let $p = f(i+1)$. If $p \notin S(i)$, the claim is immediate, so assume $p \in S(i)$. As $p \in \mathrm{Root}(G_{r_{i+1}})$, there is a path from $p$ to every $q \in \Pi$ in $G_{r_{i+1}}$. Because we assumed $|S(i)| = i < n$, there is an edge $(u,v)$ on this path such that $u \in S(i)$ and $v \in \Pi \setminus S(i)$. By construction of $S(i)$ and Definition 3.3.1, $v \in S(i+1)$.

It remains to be shown that $|S(n)| \geqslant n$ implies the theorem. By construction of $S(i)$, $S(n) \setminus \{f(n)\}$ contains only processes $p$ for which the claim holds directly or which satisfy $p = f(j)$ for a $j \in [1, n-1]$. In case of the latter, since we assume self-loops in every communication graph, $p \in \mathrm{CP}_p^{r_n}(r_j)$ also holds. □

An alternative way to express information propagation is by introducing a relation between the process-time pairs of $\Pi \times \mathbf{N}^*$ in the following way (c.f. also [BZM14]): Fix a communication pattern $\sigma = G_1, G_2, \ldots$. We say process $q$ at time $r'$ influences $p$ at time $r$, written as $(q,r') \rightsquigarrow_\sigma (p,r)$, if there is a chain of messages, starting at $q$ no earlier than time $r'$ and ending at $p$ no later than time $r$. Formally, influence is the transitive closure of the relation $(q,r) \to_\sigma (p, r+1)$ which holds if the edge $(q,p) \in G_{r+1}$. We will often simply write $\rightsquigarrow$ instead of $\rightsquigarrow_\sigma$ if $\sigma$ is understood. The *view* of a process in round $r$ for communication pattern $\sigma$ is then the graph on the process-time pairs that have influenced it, where an edge between two process time pairs represents that an edge exists in the according communication graph:

$$\mathsf{view}_\sigma^r(p) := \langle V, E\rangle \quad \text{where} \quad V = \{(q,r') \mid (q,r') \rightsquigarrow_\sigma (p,r)\} \quad \text{and} \quad ((p,r'),(q,r'')) \in E \Leftrightarrow (r \geq r'' = r'+1) \wedge (p,q) \in \sigma(r'')$$

In Section 3.5, we will elaborate this further to the concept of process-time graphs. We say that two round $r$ prefixes $\sigma|_r$, $\sigma'|_r$ are indistinguishable for $p_i$, written as $\sigma|_r \sim_{p_i} \sigma'|_r$, if $\mathsf{view}^r_{\sigma|_r}(p_i) = \mathsf{view}^r_{\sigma'|_r}(p_i)$. We write $\sigma \sim \sigma'$ if there exists a process $p$ such that $\sigma \sim_p \sigma'$ and sometimes use $\sigma \sim_P \sigma'$ if $\sigma \sim_p \sigma'$ for all $p \in P$. We use $\sigma|_r \sim^* \sigma'|_r$ to denote the transitive closure of the above relation over all processes and sequences, i.e., given a message adversary $\mathsf{MA}$, $\sigma|_r \sim^* \sigma'|_r$ means for some multiset of processes $\{p_{i_1}, \ldots, p_{i_k}\}$ of $\Pi$ and some set of sequences $\{\sigma_1, \ldots, \sigma_{k-1}\} \subseteq \mathsf{MA}$, we have

$\sigma|_r \sim_{p_{i_1}} \sigma_1|_r \sim_{p_{i_2}} \ldots \sim_{p_{i_{k-1}}} \sigma_{k-1}|_r \sim_{p_{i_k}} \sigma'|_r$. We use $\sim_p^*$ and $\sim^*$ to also relate infinite sequences that satisfy the relation for each of their finite prefixes.

**Definition 3.3.3** (Indistinguishability)**.** *For two communication patterns $\rho, \sigma$, we write $\rho \sim_p \sigma$ if for all rounds $r$: $\mathsf{view}_\rho^r(p) = \mathsf{view}_\sigma^r(p)$. We write $\rho \sim \sigma$ if $\rho \sim_p \sigma$ for some $p$ and denote by $\sim^*$ the transitive closure of $\sim$.*

The *kernel* of a prefix is the set of those processes that reach every other process, directly or via transitive messages by the end of round $r$. For an infinite communication pattern $\sigma$, we say $K = \mathrm{Ker}(\sigma)$ if there is a prefix $\sigma|_r$ s.t. $K = \mathrm{Ker}(\sigma|_r)$ and for all rounds $r'$ it holds that if $k \in \mathrm{Ker}(\sigma_{r'})$ then $k \in K$.

**Definition 3.3.4** (Kernel)**.** *For communication pattern $\sigma$ and round $r$,* $\mathrm{Ker}(\sigma|_r) := \{p_i \in \Pi \mid \forall p_j \in \Pi\colon (p_i, 0) \rightsquigarrow_{\sigma|_r} (p_j, r)\}$. *Furthermore,* $\mathrm{Ker}(\sigma) := \bigcup_{r>0} \mathrm{Ker}(\sigma|_r)$.

The causal past is particularly useful when arguing about subsets of $\Pi$ that have influenced a certain process $p$, and, in this sense, it is in the spirit of the heard-of sets from [CBS09]. In contrast, the influence relation is more in the spirit of the process-time graphs of [KOM11] and is perhaps more concise when expressing the existence of some crucial influence, originating at a single process at a certain point in time. Note that, quite different from the causal past $\mathrm{CP}_p^r(r')$, the view $\mathsf{view}_\sigma^r(p)$ encodes not only the identities of the processes that influenced $p$ from round $r$ down to round $r'$ but rather collects the identifiers of all processes that influenced $p$ by round $r$, along with the time that their respective influence began. Throughout this thesis, we will strive to use the more appropriate notation for any given circumstance, however we would like to stress that they are all equivalent in the following sense:

$$(q, r') \rightsquigarrow (p, r) \Leftrightarrow (q, r') \in \mathsf{view}^r(p) \Leftrightarrow q \in \mathrm{CP}_p^r(r')$$

## 3.4 Problems and Executions

We consider the agreement problems consensus and $k$-set agreement. In the $k$-set agreement problem, each process $p$ starts with input value $x_p \in \mathbb{N}$ and has a dedicated write-once output variable $y_p$, where $y_p = \perp$ initially; eventually, every process needs to irrevocably decide, i.e., assign a so-called output value to $y_p$ (*termination*) such that there are at most $k$ different output values system-wide at every process (*$k$-agreement*) and these outputs were the input of some process (*validity*). Consensus is 1-set agreement (for the 1-*agreement* property, we use the shorthand *agreement*).

The content of all variables as well as the messages to be sent by process $p$ at time $t$ constitute its *state*, which we shall represent by the process-time pair $(p, t)$ for simplicity. A collection $C^t = \{(p, t)\colon p \in \Pi\}$ of the time $t$ states of all processes represents a configuration. We let $(p, 0)$ represent the initial state of $p$ (in the consensus problem, respectively the $k$-set agreement problem this consists mostly of the input value assigned

to $p$) and, accordingly, $C^0$ the initial configuration. Given a message adversary MA, a deterministic consensus algorithm $\mathcal{A}$ and a $\sigma \in \mathsf{MA}$, an *admissible execution* or *run* $\varepsilon = \langle C^0, \sigma \rangle$ is a sequence of configurations $C^0, C^1 \dots$ where for $r > 0$, $C^r$ is the result of exchanging the messages to be sent according to $C^{r-1}$ and $G_r$, and applying the resulting state transitions specified by $\mathcal{A}$. Since $\mathcal{A}$ is deterministic, the execution $\varepsilon$ is uniquely determined by an admissible graph sequence $\sigma \in \mathsf{MA}$ and an initial configuration $C^0$. Algorithm $\mathcal{A}$ solves consensus under message adversary MA if, for every $\sigma \in \mathsf{MA}$ and every input assignment in $C^0$, validity, agreement and termination are all satisfied in the execution $\langle C^0, \sigma \rangle$ of $\mathcal{A}$. We will see that, in some cases, the size of the set of processes $\Pi$ may be different in selected dynamic graphs of MA and the processes must cope with this and the fact that they cannot reliably compute the size of $\Pi$. We call a consensus algorithm *uniform* (c.f. [AW04]) for MA if it solves consensus under MA and MA consists of dynamic graphs of arbitrary size.

As usual, we write $\varepsilon \sim_p \varepsilon'$ if the finite or infinite executions $\varepsilon$ and $\varepsilon'$ are *indistinguishable* to $p$ (i.e., the state of $p$ at time $r \geq 0$ is the same in both executions) until $p$ decides. When establishing our lower bounds, we will often exploit that, as outlined above, the configuration at time $r$ is uniquely determined by the initial configuration $C^0$ and the communication pattern until round $r$.

As some of our impossibility proofs rely on a bivalence argument, we briefly rephrase the terminology from [FLP85]: Consider an algorithm $\mathcal{A}$ that solves the binary consensus problem, where, for every process $p$, the initial value satisfies $x_p \in \{0, 1\}$. Given a message adversary MA, we call a configuration $C = \langle C^0, \sigma \rangle$ of $\mathcal{A}$ *univalent* or, more specifically, *$v$-valent*, if all processes decide $v$ in $\langle C, \sigma' \rangle$ for all $\sigma'$ satisfying that the concatenated sequence $\sigma \circ \sigma' \in \mathsf{MA}$. We call $C$ *bivalent*, if it is not univalent. Note that we will often state our impossibility results for binary consensus as this is usually easier to show and implies the impossibility of consensus with arbitrary input values.

## 3.5 Process-Time Graphs

*Process-time graphs* [BZM14] condense the essentials of the execution of a consensus algorithm under a message adversary into a single graph. They offer a concise formalism to provide definitions of indistinguishability, influence, views etc. that subsume the traditional definitions given in the previous sections.

For every graph sequence $\mathbf{G} = (G_t)_{t \geqslant 1}$ and every assignment of initial values $x \in \mathcal{V}_I^n$ to the $n$ processes, we inductively construct the following sequence of process-time graphs $PT^t$:

- The process-time graph $PT^0$ at time 0 contains the nodes $(p, 0, x_p)$ for all processes $p \in [n]$ and no edges.

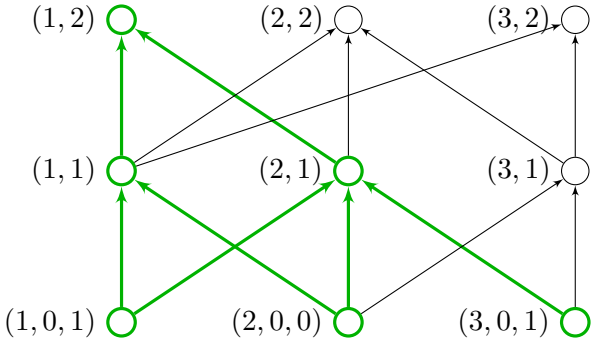


Figure 3.1: Example of a process-time graph $PT^2$ at time $t = 2$ with $n = 3$ processes and initial values $x = (1, 0, 1)$. Process 1's view $V_{\{1\}}(PT^2)$ is highlighted in bold green.

- The process-time graph $PT^1$ at time 1 contains the nodes $(p, 0, x_p)$ and $(p, 1)$ for all processes $p \in [n]$. It contains an edge from $(p, 0, x_p)$ to $(q, 1)$ if and only if $(p, q) \in G_1$.
- For $t \geqslant 2$, the process-time graph $PT^t$ at time $t$ contains the nodes of $PT^{t-1}$ and the nodes $(p, t)$ for all processes $p \in [n]$. It contains an edge from $(p, t-1)$ to $(q, t)$ if and only if $(p, q) \in G_t$.

Figure 3.1 shows an example of a process-time graph at time 2.

Let $\mathcal{PT}^t$ be the set of all possible process-time graphs at time $t \geqslant 0$, which is finite for any $t \geqslant 0$. Furthermore, let $\mathcal{PT}^\omega = \mathcal{PT}^0 \times \mathcal{PT}^1 \times \dots$ be the set of all infinite sequences of possible process-time graphs.[3]

Given an input domain $\mathcal{V}_I$, every message adversary uniquely corresponds to a subset $PS$ of $\mathcal{PT}^\omega$ generated by the graph sequences admissible under the message adversary.

Note that process-time graphs are *independent* of the particular algorithm used, but do depend on the input values. For conciseness, we will use the term process-time graphs for the elements of $\mathcal{PT}^\omega$ as well.

Regarding the equivalence of process-time graphs with our previous notations, we observe that all of our terms, defined atop of influence and view, could also be defined in terms of process-time graphs. This is not hard, as an *influence* of the form $(p, r') \rightsquigarrow_\sigma (q, r'')$ exists if there is a path from $(p, r')$ to $(q, r'')$ in $\mathcal{PT}_\sigma$, the process-time graph of communication pattern $\sigma$. In addition, the *view* $\mathsf{view}^r_\sigma(p)$ of a process $p$ in $\sigma$ can be defined as the subgraph of $\mathcal{PT}_\sigma$ that is induced by the vertex-set $\{(q, r') \colon (q, r') \rightsquigarrow_\sigma (p, r)\}$.

[3] Please note that we slightly abuse the notation $\mathcal{PT}^\omega$ here, which normally represents $\mathcal{PT} \times \mathcal{PT} \times \dots$.

Chapter 4

# Dynamic Radius of Nonsplit Graphs

Given a graph $G$ and denoting by $d(u, v)$ the length of the shortest path in $G$ from node $u$ to node $v$, the radius of $G$ is formally defined as:

**Definition 4.0.1** (Graph radius)**.** *The radius of graph $G$ is* $\min_{u \in G} \max_{v \in G} d(u, v)$.

We can interpret the radius of communication graph $G$ as the number of rounds until all processes have (transitively) received a message from a common process in the communication pattern $(G)_{r>0}$ where $G$ is repeated forever.

Its value thus poses a lower bound on the number of rounds until information, originating at a single process, can be spread over the entire network. Related applications are from disease spreading and opinion dynamics, where the radius is a lower bound on the rounds it takes to spread a disease or an opinion that originates at a single agent.

In the context of message adversaries, we generalize the radius of a communication graph $G$ to the dynamic radius of a sequence of varying communication graphs $\sigma = G_1, G_2, \ldots$. The *dynamic radius of the sequence* $\sigma$ is the number of rounds until all processes have (transitively) received a message from a common process. Recalling Definition 3.3.4, this can be formally defined as follows:

**Definition 4.0.2** (Dynamic radius)**.** *The dynamic radius of a communication pattern $\sigma$ is* $\min\{r : \mathrm{Ker}(\sigma|_r) \neq \emptyset\}$.

The *dynamic radius of a message adversary* is defined as the supremum over all dynamic radii of its communication patterns, capturing the worst-case of information dissemination within this message adversary.

## Radius of Nonsplit Digraphs

A nonsplit digraph is a directed graph where each pair of nodes has *at least one* common incoming neighbor.

**Definition 4.0.3.** *A communication graph $G = (\Pi, E)$ is* nonsplit *if*

$$\forall i, j \in [n]\ \exists k \in [n] \colon (k, i) \in E \wedge (k, j) \in E\ .$$

We call a message adversary whose communication patterns consist exclusively of nonsplit communication graphs a *nonsplit message adversary*.

In the undirected case, the radius is trivially bounded by the diameter of the graph, which is 2 in the case of nonsplit graphs. Undirected graphs where each pair of nodes has *exactly one* common neighbor have been studied by Erdős et al. [ERS66], who showed that they are exactly the windmill graphs, consisting of triangles that share a common node. Thus, their radius is 1.

As demonstrated by the example in Figure 4.1 with radius 3, these bounds do not hold for nonsplit digraphs. In Section 4.3, will prove the following upper bound:

**Theorem 4.0.1.** *The radius of a nonsplit digraph with $n$ nodes is in $O(\log \log n)$.*

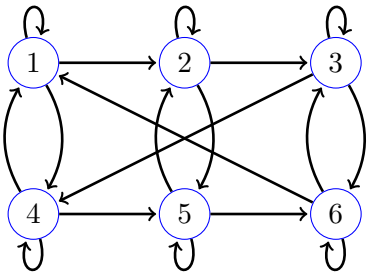


Figure 4.1: Nonsplit digraph with radius 3. For example, node 1 and 6 have common incoming neighbor 6, while nodes 1 and 5 have node 4 as common incoming neighbor.

## Communication over Nonsplit Digraphs

Nonsplit digraphs naturally occur as communication graphs in classical fault-models and as models for message adversaries.

In fact, it was shown in [CBFN15] that, in the more general case where all communication graphs are rooted, i.e., are required to contain a rooted spanning tree, one can in fact simulate nonsplit communication graphs. Since, conversely, every nonsplit communication graph can easily shown to be rooted as well, nonsplit communication graphs are a convenient and concise abstraction for the technically more cumbersome rooted communication graphs. Furthermore, several classical fault-models were shown to lead to nonsplit communication graphs [CBS09], among them link failures, as considered in [SW89], and asynchronous message passing systems with crash failures [Lyn96]. Nonsplit digraphs thus represent a convenient abstraction to these classical fault-models as well. We will show in Section 4.5 that nonsplit digraphs arising from the classical model of asynchronous messages and crashes have dynamic radius at most 2.

The study of nonsplit digraphs is also motivated by the study of agreement problems in general: For several problems, e.g., distributed control, clock synchronization, load balancing, etc., it is sufficient to asymptotically converge to the same value (asymptotic consensus), or decide on values not too far from each other (approximate consensus). Charron-Bost et al. [CBFN15] showed that both problems are solvable efficiently in the case of communication graphs that may vary arbitrarily, but are required to be nonsplit.

Motivated by this work on varying communication graphs, we will prove in Section 4.3 the following generalization of Theorem 4.0.1:

**Theorem 4.0.2.** *The dynamic radius of a nonsplit message adversary, consisting of $n$ processes, is $O(\log \log n)$.*

Combining the above Theorem 4.0.2 with the result from [CBFN15] that established a $O(n)$ simulation of nonsplit message adversaries in rooted message adversaries, i.e., message adversaries where every communication graph contains a rooted spanning tree, yields an improvement of the best previously known upper bound for the dynamic radius of message adversaries with rooted communication graphs from $O(n \log n)$ to $O(n \log \log n)$:

**Theorem 4.0.3.** *The dynamic radius of a message adversary whose communication graphs are rooted is $O(n \log \log n)$.*

Traditionally, information dissemination is studied w.r.t. either all-to-all message relaying or the time it takes for a fixed process to broadcast its message to everyone [HLPR13, HHL88]. Under a nonsplit message adversary, however, such strong forms of information dissemination are impossible. This can easily be seen by constructing appropriate sequences of star graphs (with self-loops), which are a degenerate case of nonsplit graphs with radius 1.

One-to-all broadcast of *some* process, on the other hand, is readily achieved under such message adversaries, which is why we focus on this characteristic here. While it is certainly not as universal as the previously mentioned primitives, in Chapter 6, it will turn out that this type of information dissemination is crucial for the termination time of Algorithm 6.2, a consensus algorithms based on vertex-stable root components. Furthermore, we will prove the following theorem, relating the dynamic radius and the termination time of arbitrary consensus algorithms:

**Theorem 4.0.4.** *If the dynamic radius of a communication pattern is $k$, then, in every deterministic consensus algorithm, not all processes can have terminated before time $k$.*

Finally, we note that the dynamic radius is also an upper bound for the number of rounds until a single process aggregates the data of all other processes, when we use the dual interpretation of an edge $(i, j)$ in a communication graph as a message sent by $j$ and received by $i$. Even though this might not be the desired form of data aggregation

in a standard setting, in a scenario where the communication is so constrained that aggregation by an *a priori* selected process is simply unobtainable, such a weak form might still be useful to transmit the collected data to a dedicated sink at regular intervals, for example.

**Chapter organization:** After a brief overview of related work in the next section, we proceed to introduce a few terms and notations in Section 4.2. We will then prove our main result, i.e., the dynamic radius of a nonsplit message adversary in Section 4.3 and show the implications this has on the solvability of the consensus problem in Section 4.4. Finally, we show an interesting connection between the classic asynchronous model of distributed computing and a nonsplit message adversary in Section 4.5.

## 4.1 Related Work

In the model of [KLO10], it was assumed that the communication graphs are connected and undirected. For this, a lower bound of $\Omega(n \log n)$ and an upper bound of $O(n^2)$ for all-to-all token dissemination was established in the case where $n$ is unknown to the participants, they have to terminate when the broadcast is finished, and the system is 1-interval connected, i.e., the communication graphs are completely independent of each other. In contrast, if the communication graphs are connected, directed, and rooted, in the worst case only one of the tokens may ever be delivered to all participants. This can be seen, for example, when considering a dynamic graph that produces the same line graph for every round. We note that this example also provides a trivial lower bound of $\Omega(n)$ rounds until one token is received by everyone for the first time. As far as we are aware, the best such lower bound was established in [ZSS19, Theorem 4.3] to be $\lceil (3n-1)/2 - 2 \rceil$ rounds.

In [CBS09], it was shown that the dynamic radius of a sequence of arbitrary nonsplit communication graphs is $O(\log n)$. Later, it was shown in [CBFN15] that the product of any $n-1$ rooted communication graphs is nonsplit. Put together, this means that the dynamic radius of a sequence of arbitrary rooted graphs is $O(n \log n)$. More recently, [ZSS19] provided an alternative proof for this fact that does not rely on the reduction to nonsplit graphs but instead uses a notion of influence sets similar to [KLO10, Lemma 3.2.(b)]. In addition to this, [ZSS19] provided linear $O(n)$ bounds in sequences of rooted trees with a constant number of leaves or inner nodes, established a dependency on the size of certain subtrees in sequences of rooted trees where the root remains the same, and investigated sequences of undirected trees.

## 4.2 Model and Definitions

We start with some definitions motivated by the study of information dissemination under a message adversary that controls the communication within a distributed system of $n$ processes. For ease of notation, throughout this chapter, we will assume that the

set of processes $\Pi$ is represented by the set $[n] = \{1, \dots, n\}$. Starting with information being available only locally to each process, processes broadcast and receive information tokens in every round. We are interested in the earliest round where all processes have received an information token from a common process.

A node $i \in G$ is called a *broadcaster in* $G$ if it has an edge to all nodes, i.e., $\forall j \in [n]\colon (i,j) \in E$.

With the product graph defined as in Section 3.2, we let the empty product be equal to the communication graph $([n], E_\perp)$ which contains the self-loops $(i,i)$ for all nodes $i$ and no other edges. The usage of the graph product here is motivated by information dissemination within distributed systems of processes that continuously relay information tokens that they received: if $k$ received $i$'s information token in a round, and $j$ received $k$'s information token in the next round, then $j$ received $i$'s information token in the macro-round formed by these two successive rounds.

For every node $i \in [n]$ and every communication pattern $\sigma$, define the *broadcast time* $T_i(\sigma)$ *of node* $i$ *in* $\sigma$ as the minimum $t$ such that $i$ is a broadcaster in the product of the first $t$ communication graphs of $\sigma$, i.e.,

$$T_i(\sigma) = \min\left\{t\colon \bigcap_{i\in[n]} \mathrm{CP}_i^t(0) \neq \emptyset\right\} \ .$$

If no such $t$ exists, then $T_i(\sigma) = \infty$. The dynamic radius $T(\sigma)$ of $\sigma$ is the minimal broadcast time of its nodes, i.e., $T(\sigma) = \min_{i\in[n]} T_i(\sigma)$. Note that $T(\sigma)$ is the earliest time, in terms of rounds, until that all nodes have received an information token from a common node, given that the communication pattern is $\sigma$.

## 4.3 The Dynamic Radius of Nonsplit Message Adversaries

In this section we show an upper bound on the dynamic radius of nonsplit message adversaries.

During this section, let $\sigma = (G_1, G_2, G_3, \dots)$ be a communication pattern on $n$ nodes in which every communication graph $G_t$ is nonsplit.

In order to prove an upper bound on the dynamic radius of $\sigma$, we will prove the existence of a relatively small set of nodes that "infects" all other nodes within only $O(\log\log n)$ rounds. Iteratively going back in time, it remains to be shown that any such set is itself "infected" by an exponentially smaller set within $O(\log\log n)$ rounds, until we reach a single node. It follows that this single node has "infected" all nodes with its information token after $O(\log\log n)$ rounds.

Note that the strategy to follow "infection" back in time rather than considering the evolution of infected sets over time is essential in our proofs: it may very well be that a

certain set of infected nodes cannot infect other nodes from some time on, since it only has incoming edges from nodes not in the set in all successive communication graphs. Going back in time prevents us to run into such dead-ends of infection.

For that purpose we define: Let $U, W \subseteq [n]$ be sets of nodes. We say that $U$ *covers* $W$ *in communication graph* $G = ([n], E)$ if for every $j \in W$ there is some $i \in U$ that has an edge to $j$, i.e., $\forall j \in W\ \exists i \in U \colon (i, j) \in E$.

Now let $0 < t_1 \leqslant t_2$. We say that $U$ *at time* $t_1$ *covers* $W$ *at time* $t_2$ if $U$ covers $W$ in the product graph $G_{t_1} \circ G_{t_1+1} \circ \cdots \circ G_{t_2-1}$.

Note that $U$ at time $t$ covers $U$ at time $t$ for all sets $U \subseteq [n]$ and all $t \geqslant 1$, by definition of the empty product as the digraph with only self-loops.

We first show that the notion of covering is transitive:

**Lemma 4.3.1.** *Let $0 < t_1 \leqslant t_2 \leqslant t_3$ and let $U, W, X \subseteq [n]$. If $U$ at time $t_1$ covers $W$ at time $t_2$, and $W$ at time $t_2$ covers $X$ at time $t_3$, then $U$ at time $t_1$ covers $X$ at time $t_3$.*

*Proof.* By definition, for all $k \in W$ there is some $i \in U$ such that $(i, k)$ is an edge of the product graph $G_{t_1} \circ \cdots \circ G_{t_2-1}$. Also, for all $j \in X$ there is some $k \in W$ such that $(k, j)$ is an edge of the product graph $G_{t_2} \circ \cdots \circ G_{t_3-1}$.

But, by the associativity of the graph product, this means that for all $j \in X$ there exists some $i \in U$ such that $(i, j)$ is an edge in the product graph

$$(G_{t_1} \circ \cdots \circ G_{t_2-1}) \circ (G_{t_2} \circ \cdots \circ G_{t_3-1}) = G_{t_1} \circ \cdots \circ G_{t_3-1} \ .$$

That is, $U$ at time $t_1$ covers $X$ at time $t_3$. □

We continue with some basic technical lemmas that we prove here for completeness.

**Lemma 4.3.2.** *For all $x \geq 1$ we have $\lceil \log_2 x \rceil = \lceil \log_2 \lceil x \rceil \rceil$.*

*Proof.* We have $\lceil \log_2 x \rceil = \min\{k \in \mathbb{Z} \mid x \leqslant 2^k\}$ if $x \geq 1$. Now, noting that the inequality $x \leqslant p$ is equivalent to $\lceil x \rceil \leqslant p$ whenever $p$ is an integer concludes the proof. □

**Lemma 4.3.3.** *Let $m$ and $n$ be positive integers such that $|m - n| \leqslant 1$. Then $\lceil \log_2(m + n) \rceil \geqslant \lceil \log_2 m \rceil + 1$.*

*Proof.* By assumption we have $n \geqslant m - 1$. We distinguish between two cases for the positive integer $m$:

(i) If $m = 1$ then $n \in \{1, 2\}$, and we immediately obtain the lemma from $\lceil \log_2 2 \rceil = \lceil \log_2 1 \rceil + 1$ and $\lceil \log_2 3 \rceil = \log_2 4 = \lceil \log_2 2 \rceil + 1$.

(ii) Otherwise, $m \geq 2$. From $n \geqslant m-1$ we deduce $m+n \geqslant 2m-1$. This implies

$$\lceil \log_2(m+n) \rceil \geqslant \lceil \log_2(2m-1) \rceil = \left\lceil \log_2 \left( m - \frac{1}{2} \right) \right\rceil + 1 \ .$$

We are hence done if we can show $\lceil \log_2(m-\frac{1}{2}) \rceil = \lceil \log_2 m \rceil$. But this is just Lemma 4.3.2 with $x = m - \frac{1}{2} \geq 1$. □

**Lemma 4.3.4.** *Let $n$ and $m$ be positive integers such that $n \geq m$. Then there exist positive integers $n_1, n_2, \ldots, n_m$ such that $n = n_1 + \cdots + n_m$ and $\lceil \log_2 \frac{n}{m} \rceil \geqslant \lceil \log_2 n_i \rceil$ for all $1 \leqslant i \leqslant m$.*

*Proof.* Let $n = km + r$ with $k, r \in \mathbb{Z}$ and $0 \leqslant r < m$ be the integer division of $n$ by $m$. Set $n_1 = n_2 = \cdots = n_r = k+1$ and $n_{r+1} = n_{r+2} = \cdots = n_m = k$.

By Lemma 4.3.2, we have

$$\lceil \log_2 n_i \rceil \leqslant \left\lceil \log_2 \left( k + \left\lceil \frac{r}{m} \right\rceil \right) \right\rceil = \left\lceil \log_2 \left( k + \frac{r}{m} \right) \right\rceil = \left\lceil \log_2 \frac{n}{m} \right\rceil$$

for all $1 \leqslant i \leqslant m$. □

We proceed with a generalization of a result by Charron-Bost and Schiper [CBS09]. In particular ($m = 1$), it shows that any set of nodes can be "infected" by a single node in such a way that the set of infected nodes grows exponentially in size per round.

**Lemma 4.3.5.** *Let $W \subseteq [n]$ be nonempty and $m$ be a positive integer. If $t_2 - t_1 \geqslant \log_2 \frac{|W|}{m}$, then there exists some $U \subseteq [n]$ with $|U| \leqslant m$ such that $U$ at time $t_1$ covers $W$ at time $t_2$.*

*Proof.* Using Lemma 4.3.4, we can assume without loss of generality that $m = 1$: For $m > 1$, we would need to show that for every $i \in U$, there are $n_i$ distinct processes covered by $i$, given that $t_2 - t_1 \geq \log_2 n_i$ and $\sum_{i \in U} n_i = |W|$. This, however, is equivalent to the claim of the lemma for $m = 1$. We proceed by induction on $t_2 - t_1 \geq 0$.

Base case: If $t_2 - t_1 = 0$, i.e., $t_1 = t_2$, then $|W| = 1$ and the statement is trivially true since we can choose $U = W$.

Inductive step: Now let $t_2 - t_1 \geqslant 1$. Let $W = W_1 \cup W_2$ such that $W_1, W_2 \neq \emptyset$ and $||W_1| - |W_2|| \leqslant 1$. Using Lemma 4.3.3, we see that $t_2 - (t_1+1) \geqslant \lceil \log_2 |W_s| \rceil$ for $s \in \{1, 2\}$. By the induction hypothesis, there hence exist nodes $j_1$ and $j_2$ that at time $t_1+1$ cover $W_1$ and $W_2$, respectively. But now, using the nonsplit property of communication graph $G_{t_1}$, we see that there exists a node $i$ that covers $\{j_1, j_2\}$ in $G_{t_1}$. An application of Lemma 4.3.1 concludes the proof. □

Note that Lemma 4.3.5, by choosing $W = [n]$ and $m = 1$, immediately provides an upper bound on the dynamic radius of $O(\log n)$. To show an upper bound of $O(\log \log n)$, we will apply this lemma only for the early infection phase of $O(\log \log n)$ rounds, and use a different technique, by the next two lemmas, for the late phase. Note that, for a set $N$, we use $\binom{N}{k}$ to denote all subsets of $N$ with cardinality $k$ and for an integer $n$, we use $\binom{n}{k}$ to denote the binomial coefficient $n$ choose $k$.

**Lemma 4.3.6.** *Let $U$ and $W$ be finite sets with $|U| = k$, $|W| = n$, and $f : \binom{U}{\lfloor \log n \rfloor} \to W$. If $n \geqslant 8$, then there exists some $w \in W$ such that $\left|\bigcup f^{-1}[\{w\}]\right| \geqslant k/e^4$.*

*Proof.* By the pigeonhole principle and Stirling's formula, we get the existence of some $w \in W$ with

$$\left|f^{-1}[\{w\}]\right| \geqslant \frac{\binom{k}{\lfloor \log n \rfloor}}{n} \geqslant \frac{k^{\lfloor \log n \rfloor}}{n \lfloor \log n \rfloor^{\lfloor \log n \rfloor}} \quad . \tag{4.1}$$

Write $M = \bigcup f^{-1}[\{w\}]$ and $m = |M|$. Since $S \in \binom{M}{\lfloor \log n \rfloor}$ for all $S \in f^{-1}[\{w\}]$, we have

$$|f^{-1}[\{w\}]| \leqslant \binom{m}{\lfloor \log n \rfloor} \leqslant \frac{m^{\lfloor \log n \rfloor} e^{\lfloor \log n \rfloor}}{\lfloor \log n \rfloor^{\lfloor \log n \rfloor}} \leqslant \frac{m^{\lfloor \log n \rfloor} n}{\lfloor \log n \rfloor^{\lfloor \log n \rfloor}} \quad . \tag{4.2}$$

Combining (4.1) and (4.2), we get

$$\begin{aligned} m &\geqslant \frac{k}{n^{2/\lfloor \log n \rfloor}} = \frac{k}{e^{2 \log n / \lfloor \log n \rfloor}} \geqslant \frac{k}{e^{2 \log n / (\log n - 1)}} \\ &= \frac{k}{e^{2/(1 - 1/\log n)}} \geqslant \frac{k}{e^{2/(1-1/2)}} = \frac{k}{e^4} \end{aligned} \tag{4.3}$$

where we used $\log n \geqslant \log 8 \geqslant 2$. This concludes the proof. □

The next lemma shows that nodes are infected fast in the late phase:

**Lemma 4.3.7.** *There exists some $C > 0$ such that for all $t \geqslant 1$ there exists a set of at most $C \log n$ nodes that at time $t$ covers the set $[n]$ of all nodes at time $t + \lceil \log_2 \log n \rceil$.*

*Proof.* For every set $A \in \binom{[n]}{\lfloor \log n \rfloor}$ of $\lfloor \log n \rfloor$ nodes, let $f(A) \in [n]$ be a node that at time $t$ covers $A$ at time $t + \lceil \log_2 \log n \rceil$, which exists by Lemma 4.3.5.

We recursively define the following sequence of nodes $v_i$, $\geqslant 1$ and sets of nodes $V_i$, $i \geqslant 0$:

- $V_0 = [n]$
- For $i \geqslant 1$, we choose $v_i$ such that $\left|\bigcup f^{-1}[\{v_i\}]\right| \geqslant |V_{i-1}|/e^4$, which exists by Lemma 4.3.6, and $V_i = V_{i-1} \setminus \bigcup f^{-1}[\{v_i\}]$.

Note that, setting $r = 1 + \log n / \log \frac{e^4}{e^4-1}$, we have $V_r = \emptyset$. Hence the set $\{v_1, \dots, v_r\}$ at time $t$ covers all nodes at time $t + \lceil \log_2 \log n \rceil$. Noting $r = O(\log n)$ concludes the proof. □

We are now ready to combine Lemma 4.3.5 for the early phase and Lemma 4.3.7 for the late phase to prove the main result of this section, Theorem 4.0.2.

**Theorem 4.0.2.** *The dynamic radius of a nonsplit message adversary, consisting of $n$ processes, is $O(\log \log n)$.*

*Proof.* Let $t = \lceil \log_2(C \log n) \rceil$ where $C$ is the constant from Lemma 4.3.7.

By Lemma 4.3.7, there is a set $A$ of nodes with $|A| \leqslant C \log n$ that at time $t$ covers all nodes at time $t + \lceil \log_2 \log n \rceil$. By Lemma 4.3.5, a single node at time 1 covers $A$ at time $t$.

Combining both results via Lemma 4.3.1 shows that a single node at time 1 covers all nodes at time $\lceil \log_2(C \log n) \rceil + \lceil \log_2 \log n \rceil = O(\log \log n)$. □

As we have mentioned in the beginning of the chapter, the radius of a communication graph is a lower bound for the dynamic radius. This allows us to conclude from Theorem 4.0.2 the following bound for the radius of nonsplit graphs:

**Theorem 4.0.1.** *The radius of a nonsplit digraph with $n$ nodes is in $O(\log \log n)$.*

## 4.4 A Lower Bound for Consensus in Dynamic Networks

We now show that the dynamic radius of a message adversary represents a lower bound on the time complexity of a consensus algorithm for this message adversary. In Chapter 5, we provide a slightly rephrased version of Theorem 4.0.4 as Theorem 5.3.2, which we will use in Lemma 5.4.6 to further generalize this result.

**Theorem 4.0.4.** *If the dynamic radius of a communication pattern is $k$, then, in every deterministic consensus algorithm, not all processes can have terminated before time $k$.*

*Proof.* Let $\sigma$ be a communication pattern with dynamic radius $k$, which occurs in the message adversary by assumption. Suppose, in some deterministic consensus algorithm $A$, all $i \in [n]$ have terminated by time $k-1$ in every execution based on $\sigma$. Let $C_0$ be the input assignment where $x_i = 0$ for all $i \in [n]$ and $C_1$ be the input assignment where $x_i = 1$ for all $i \in [n]$. By validity, when running $A$ under $\sigma$ and starting from $C_0$, all $i \in [n]$ have $y_i = 0$ by time $k-1$ and when starting from $C_1$, they have $y_i = 1$. Thus, there are input assignments $C, C'$ that differ only in the input assignment $x_j$ of a single process $j$ and, for all $i \in [n]$, at time $k-1$, $y_i = 0$ when applying $A$ under $\sigma$ when starting from $C$ and $y_i = 1$ when starting from $C'$. Since there is no broadcaster in $\sigma$ before round $k$, there is some process $i'$ that did not receive a (transitive) message from

$j$ and thus $i'$ is in the same state in both executions. Therefore, $i'$ decides on the same value in both executions, which is a contradiction and concludes the proof. □

## 4.5 Nonsplit Message Adversaries from Asynchronous Rounds

We finally show that in an important special case of nonsplit message adversaries, namely those evolving from distributed algorithms that establish a round structure over asynchronous message passing in the presence of crashes, the dynamic radius is at most 2.

In the classic asynchronous message passing model with crashes, it is assumed that all messages sent have an unbounded but finite delay until they are delivered. Furthermore, processes do not operate in lock-step but may perform their computations at arbitrary times relative to each other. In addition, some processes may be faulty in the sense that they are prone to crashes, i.e., they may cease to perform computations at an arbitrary point in time.

This means that in a system where up to $f$ processes may be faulty, in order to make progress in a distributed algorithm, a process may wait until it received a message from $n - f$ different processes but no more: If a process waits for a message from $> n - f$ different processes, but there were in fact $f$ crashes, this process will wait forever. For this reason, algorithms for this asynchronous model often employ the concept of *asynchronous rounds*, sometimes realized as a local round counter variable $r_i$, which is held by each process $i \in [n]$ and appended to every message. A process $i$ increments $r_i$ only if it received a message containing a round counter $\geq r_i$ from $n - f$ different processes.

One may now ask how fast information can spread in this distributed computing model.

For this purpose, we consider a network whose communication patterns are induced by $n$ processes communicating in asynchronous rounds. Here, an edge $(i, j)$ in the communication graph $G_t$ represents that $j$ received a message from $i$ and the round counter, appended to this message, was $r_i \geqslant t$. When deriving the communication graph $G_t$ of such an asynchronous round $t$, we get a digraph where each correct process has at least $n - f$ incoming neighbors. In fact, when we slightly abuse notation and define the arrival of a message at a crashed process as a (virtual) reception that is represented in the communication graph $G_t$ as well, we get that $G_t$ is actually a digraph where all processes have at least $n - f$ incoming neighbors.

We restrict our attention to the case where $n > 2f$, i.e., a majority of the processes is correct. This implies that the sets of incoming neighbors of any two processes in a communication graph have a non-empty intersection, which means that the communication graph is nonsplit. Note that if $n \leqslant 2f$, then the network is not necessarily nonsplit. In fact, it can be disconnected into two disjoint sets of processes that do not receive messages from each other until termination of the algorithm. Theorem 4.0.1 establishes a constant upper bound on the dynamic radius of this important class of nonsplit graphs.

**Theorem 4.5.1.** *Let $f \geq 0$, $n > 2f$, and $(G_r)_{r\geq 1}$ be a communication pattern with $\mathrm{In}_i(G_r) \geq n - f$ for all $r$ and all $i$. The dynamic radius of $(G_r)_{r\geq 1}$ is at most 2.*

*Proof.* To show the bound on the radius we prove that there exists a center node $m$ that realizes the dynamic radius, i.e.,

$$\exists m \in [n]\ \forall i \in [n]\ \exists j \in [n] : m \in \mathrm{In}_j(G_1) \wedge j \in \mathrm{In}_i(G_2)\,. \tag{4.4}$$

Equation (4.4) now follows from

$$\exists m \in [n] : |\mathrm{Out}_m(G_1)| \geq f + 1\,, \tag{4.5}$$

by the following arguments: Equation (4.5) states that the information at $m$ has been transmitted to at least $f + 1$ nodes. By assumption, $\mathrm{In}_i(G_2) \geq n - f$ for all $i \in [n]$. Thus, each $i$ must have an incoming neighbor $j$ in digraph $G_2$ such that $j \in \mathrm{Out}_m(G_1)$; Equation (4.4) follows.

It remains to show (4.5). Suppose that the equation does not hold, i.e.,

$$\forall j \in [n] : |\,\mathrm{Out}_j(G_1)| \leq f\,. \tag{4.6}$$

By assumption on digraph $G_1$, we have

$$\sum_{i\in[n]} |\,\mathrm{In}_i(G_1)| \geq n(n - f) \tag{4.7}$$

By the handshake lemma,

$$\sum_{i\in[n]} |\,\mathrm{In}_i(G_1)| = \sum_{j\in[n]} |\mathrm{Out}_j(G_1)|$$

and, using (4.6),

$$\sum_{i\in[n]} |\,\mathrm{In}_i(G_1)| \leq nf\,.$$

Together with (4.7), we have $n(n - f) \leq nf$; a contradiction to the assumption that $n > 2f$. □

Chapter 5

# Closed Message Adversaries

One important and well-studied class of message adversaries are *oblivious* ones [CGP15], where the MA may pick every $G_r$ arbitrarily, i.e., without restrictions, from a given set $\mathbf{D}$ of candidate graphs. We can hence write $\mathsf{MA} = \mathbf{D}^\omega$ for an oblivious message adversary. A prominent example is the message adversary that allows a certain number of mobile link failures per round, which was studied in the seminal work by Santoro and Widmayer [SW89]. Consensus has been shown to be impossible if $\mathbf{D}$ contains all communication graphs where $\leqslant n-1$ edges are missing (excluding self-loops).

Oblivious message adversaries do not allow to change the set of candidate graphs over time, however, which makes them unsuitable to model transient performance variations in real-world dynamic networks. In wireless mobile ad-hoc networks, for example, there may be substantial periods of time where some nodes are outside the communication range of others, e.g. in disaster-relief applications [LHSP11] or under strong interference by other radio transmitters [KM07]. Another source of time-varying communication conditions are mode switches, caused by the nodes themselves, due to reasons such as energy-saving, boot-up completion or fault recovery.

In this chapter, we first provide a summary of the characterization from [CGP15]. This characterization is based on the broadcastability of the $\beta$-classes, an equivalence class of communication graphs, resulting from the $\beta$-relation. The $\beta$-classes are the equivalence classes of the $\beta$-relation, a refinement of the transitive closure of the standard indistinguishability relation between the graphs of $\mathbf{D}$. Here, broadcastability means that there exists a process $p$ such that each communication graph of the $\beta$-class is rooted and $p$ is a member of its root component (c.f. Definition 3.2.1).

After this brief summary, we focus on the class of *limit-closed* message adversaries (subsequently called *closed*[1] MAs for brevity), which are a proper superset of oblivious

[1] In the topological framework of Chapter 10, which follows-up on [AS85, NSW19], a closed message adversary corresponds to a compact space, which is why, in a topological context, the term "compact message adversary" is preferable. Because we do not use topological reasoning in this chapter, we stick to the terminology of [LM95] in this chapter and use the term "closed message adversary" instead.

message adversaries. Their characterizing property is that every sequence $(\sigma_i)_{i\geq 1}$, such that each $\sigma_i$ is the *prefix* of an admissible communication pattern and each $\sigma_i$ is a prefix of $\sigma_{i+1}$, has a limit $\sigma \in \mathsf{MA}$. Equivalently, for every communication pattern $\sigma$ that is inadmissible under a closed message adversary, there exists a round $r$ such that all extensions of the round $r$ prefix of $\sigma$ are inadmissible as well. Note that this definition makes the set of admissible sequences of the MA closed and hence a safety property in the spirit of [AS85].

It follows directly from its definition that an oblivious message adversary is closed, but there are also closed message adversaries that are not oblivious. An important example are MAs that ensure *bounded instability*. Such message adversaries guarantee some "global stabilization round" $r_{GST}$, which must be bounded (with some known bound), such that that the communication graphs become "nice" from round $r_{GST}$ onwards. The latter could occur in a myriad of ways, however: For example, starting from $r_{GST}$, a benign dynamic graph structure (like the vertex-stable source component from Chapter 6 that persists for sufficiently many rounds) could appear. Alternatively, the number of rounds it takes for some processes to reach all other processes could be bounded, analogous to partially synchronous systems [DLS88].[2] A different example are message adversaries that assemble communication patterns by concatenating finite subsequences of graphs, picked from some finite set of such subsequences with a fixed maximal length. Its choice can either be unrestricted, similar to oblivious message adversaries, or even subject to bounded instability.

On the other hand, relaxing the above requirement of $r_{GST}$ being bounded to being unbounded but finite provides an illustrative example for a *general* message adversary. A MA that guarantees stability only eventually, i.e., after *unbounded*[3] *instability*, is not closed. In fact, the communication pattern where stability *never* occurs does not have a finite prefix such that all extensions are inadmissible, as any such finite prefix could still be made admissible by attaching a stability phase. We do not aim at a combinatorial characterization of consensus solvability for non-closed MAs in this chapter; a topological characterization can be found in Chapter 10, however.

### An example for $n = 2$

In order to illustrate the different message adversary classes, we consider the case where the set of processes is $\Pi = \{\circ, \bullet\}$, i.e., $n = 2$, and the communication graphs are $\circ\leftarrow\bullet$, $\circ\leftrightarrow\bullet$, $\circ\rightarrow\bullet$, and $\circ \quad \bullet$ (in the last graph, there is no edge between the processes).

In this example, an oblivious message adversary is represented by a subset of these communication graphs and considers all sequences admissible that consist exclusively

[2] In contrast to [DLS88], message adversaries allow us to restrict precisely which processes may communicate with each other and which may not.

[3] Note that finite but unbounded $r_{GST}$ is equivalent in terms of task solvability to $r_{GST}$ being bounded with an unknown bound: The code of a solution algorithm $\mathcal{A}$ for the latter cannot depend on any bound on $r_{GST}$, yet must work for every finite value of $r_{GST}$.

of graphs from this subset. Thus, every oblivious message adversary that permits the communication graph $\circ \quad \bullet$ makes solving consensus trivially impossible. Furthermore, the oblivious message adversary that permits the communication graphs $\circ\!\leftarrow\!\bullet$, $\circ\!\leftrightarrow\!\bullet$, and $\circ\!\rightarrow\!\bullet$ is known to make consensus impossible at least since [SW89]. However, removing only one graph from this set of possible communication graphs already makes consensus solvable: If the message adversary permits $\circ\!\leftarrow\!\bullet$ and $\circ\!\leftrightarrow\!\bullet$, both processes may decide the input of $\bullet$ after the first round. If it permits $\circ\!\leftarrow\!\bullet$ and $\circ\!\rightarrow\!\bullet$, a process may decide on the other's input if it received the other's message in the first round and on its own input otherwise.

We get an example of a closed message adversary by allowing the communication graphs $\circ\!\leftarrow\!\bullet$, $\circ\!\leftrightarrow\!\bullet$, and $\circ\!\rightarrow\!\bullet$, provided there is some known round $r_{GST}$ by which it is guaranteed that the same communication graph has occurred consecutively, in rounds $r$ and $r+1$. This message adversary is closed, because every limit of a sequence $(\sigma_i)_{i\geq 1}$ of admissible prefixes, s.t. $\sigma_i$ is a prefix of $\sigma_{i+1}$ is admissible here: Intuitively, the reason is that every prefix in the sequence that is longer than $r_{GST}$ rounds, in order to be admissible, must satisfy the property of the message adversary as described above and every continuation of such a sequence is also admissible. We can hence use Algorithm 6.2 introduced in Section 5.5 for solving consensus.

The same message adversary, however with the property that $r_{GST}$ is finite but unbounded, provides an example of a non-closed message adversary. Consensus is solvable even under this message adversary; a suitable algorithm has been provided in [WSS19]. Still, our Algorithm 6.2 does not work anymore, since the MA is not closed: Since the repetition of the same graph twice in a row may occur arbitrarily late, there is a limit sequence $(\sigma_i)_{i\geq 1}$ that consists entirely of admissible prefixes where it did not occur yet. This is an inadmissible communication pattern for this message adversary, however.

*Chapter organization:* In this chapter, we provide a complete combinatorial characterization of consensus solvability under closed message adversaries. Compared to the topological characterization provided in Chapter 10, it is considerably less abstract and, more importantly, also fully operational: it utilizes an easy to check property of (finitely many) *prefixes* of admissible communication patterns, rather than properties of (uncountably many) infinite communication patterns. Compared to the combinatorial characterization for oblivious message adversaries developed in [CGP15], which we summarize in Section 5.2, our characterization applies to the larger class of closed MAs, and relies on checking the simple dynamic graph property "non-empty kernel intersection" (see Theorem 5.0.1 below), rather than on an involved algorithm that exploits certain properties of the so-called "$\beta$-classes".

In more detail, we present a condition on the admissible communication patterns of a message adversary, which we prove to be both necessary and sufficient for solving consensus. The condition, given in Theorem 5.0.1 below, rests on two main ingredients:

(i) An equivalence relation $\sigma|_r \sim \rho|_r$ on the $r$-round prefixes of the communication patterns $\sigma, \rho$, which is the transitive closure of the per-process equivalence relation

$\sigma|_r \sim_{p_i} \rho|_r$; the latter holds if process $p_i$ cannot distinguish $\sigma|_r$ from $\rho|_r$ in every round $r$.

(ii) The set of processes (called the *kernel* $\mathrm{Ker}(\sigma|_r)$) that influence every process in the system within $\sigma|_r$. Note that the processes in $\mathrm{Ker}(\sigma|_r)$ are the ones that manage to broadcast their initial value to all processes within $\sigma|_r$, and are hence sometimes called *broadcasters*.

Whereas it is not too difficult to prove (see Theorem 5.3.2) that solving consensus within $r$ rounds under the communication pattern $\sigma$ requires the existence of a broadcaster in $\sigma|_r$, i.e., $\mathrm{Ker}(\sigma|_r) \neq 0$, this is only a *necessary* condition. What needs to be added to also make it sufficient is that actually *all* transitively indistinguishable prefixes $\rho|_r$ must have some *common* element(s) in their kernels $\mathrm{Ker}(\rho|_r)$: With $[\sigma|_r]$ denoting the equivalence class w.r.t. $\sim$ containing $\sigma|_r$, which can be computed from the message adversary specification, our main result is the following:

**Theorem 5.0.1.** *Consensus is solvable under a closed message adversary* $\mathsf{MA}$ *if and only if for each* $\sigma \in \mathsf{MA}$ *there is a round* $r$ *such that* $\bigcap_{x\in[\sigma|_r]} \mathrm{Ker}(x) \neq \emptyset$.

Theorem 5.0.1 is not only interesting from a theoretical point of view, but may also have practical implications in that it allows to avoid attempts on the development of consensus algorithms for dynamic networks that actually do not allow any solution. After briefly summarizing the existing characterization using the $\beta$-classes by [CGP15] in Section 5.2, the remainder of this chapter is devoted to the proof of this theorem and is organized as follows: In Section 5.3, we present our system and computation model, the definition of our message adversaries and their properties, and the specification of the consensus problem. In Section 5.4, we use indistinguishability arguments and König's Infinity Lemma to prove that consensus is impossible if the condition in Theorem 5.0.1 does not hold. In Section 5.5, we prove the sufficiency of our condition, by specifying an algorithm that solves consensus under a message adversary that satisfies Theorem 5.0.1.

## 5.1 Related Work

A refinement of the classic consensus solvability characterization of [SW89] was provided in [SWK09] and, more recently, by Coulouma et al. in [CGP15], where a property of an equivalence relation on the sets of communication graphs was found that captures exactly the source of consensus impossibility under an oblivious message adversary. The authors also showed how this property can be exploited in order to develop a generic consensus algorithm.

The first characterization of consensus solvability under general message adversaries was provided in [FG11], albeit only for systems that consist of two processes. A bivalence argument has been used to show that certain communication patterns, namely, a fair or a special pair of unfair communication patterns must be excluded by $\mathsf{MA}$ for consensus

to become solvable. In [NSW19], which we will present in Chapter 10, Nowak et al. provided a complete topological characterization for systems of arbitrary size, which relies on non-trivial extensions of the seminal work by Alpern and Schneider [AS85]. It focuses on the space of communication patterns (actually, the corresponding infinite sequences of process-time graphs resp. configurations), and defines topologies based on variants of the well-known common-prefix metric. The authors show that consensus is solvable for a given MA, if and only if this space partitions into multiple components $PS(v)$ consisting of the executions with decision value $v$. For closed message adversaries, these sets are shown to be compact and hence closed, for non-closed MAs, they are only relative compact. In both cases, however, it turns out that the existence of this partitioning is tied to the broadcastability of every individual $PS(v)$, by the some common process. Note that this requirement is obviously enforced by the condition given in Theorem 5.0.1 as well.

Regarding the study of closed message adversaries, the seminal works by Dolev et al. [DDS87] and Dwork et al. [DLS88] on partially synchronous systems introduced important abstractions like eventual stabilization and eventually bounded message delays, and provided a characterization of consensus solvability under various combinations of synchrony and failure models (including byzantine-faulty processes).

Unified consensus impossibilities in asynchronous distributed systems have been provided by Lubitch and Moran in [LM95]. The authors focused on closed subsets of runs of asynchronous distributed algorithms, defined by the property that every path in the corresponding configuration tree corresponds to an admissible execution (which is equivalent to limit-closure). Using a model-independent construction of closed schedulers that generate closed sets of runs, they provided a unified impossibility proof for consensus using a bivalence argument. The simplicity of their approach rests on the fact that the so constructed forever bivalent run lies in the closed set of runs, and is hence admissible. This convenient property of closed sets of communication patterns is also used in this chapter, albeit we use König's Infinity Lemma rather than a bivalence argument for constructing a non-terminating run. We briefly explain at the end of Section 5.4 why our model is not suitable for a bivalence argument like the one in [LM95].

## 5.2 Broadcastable $\beta$-Classes by Coulouma, Godard, and Peters

We now provide an overview of the paper "A characterization of oblivious message adversaries for which consensus is solvable" by Coulouma, Godard, and Peters [CGP15]. To the best of our knowledge, this (resp. its preliminary version, [CG13]) is the first paper that provides a complete characterization of oblivious message adversaries based on the property of the set of possible graphs $\mathbf{D}$. Recall that, in this model, the message adversary may choose a communication graph from the set $\mathbf{D}$ in each round without any further restrictions (such as eventually ...). In this sense, the message adversary that corresponds to $\mathbf{D}$ can be expressed as $\mathbf{D}^\omega$, i.e., all infinite sequences of graphs of $\mathbf{D}$.

At the core of their consensus characterization lies a host of relations between communication graphs. Given a graph $G = \langle V, E\rangle$, let $\mathcal{S}(G) = \text{Root}(G) \subseteq V$ denote its root component (see Definition 3.2.1; this was called the *set of sources* in [CGP15]), i.e., for each $p \in \mathcal{S}(G)$ there is a directed path in $G$ from $p$ to every $q$ of $V$. For $X \subseteq V$, let $\text{In}^e_X(G) = \{(u, v) \in E \mid v \in X\}$ denote the in-edges of the vertices of $X$ in $G$. For two graphs $G, H \in \mathbf{D}$ we write $G\ \alpha_K\ H$ if $\text{In}^e_{\mathcal{S}(K)}(G) = \text{In}^e_{\mathcal{S}(K)}(H)$ for some $K \in \mathbf{D}$, i.e., the sources of $K$ cannot (on their own) distinguish between $G$ and $H$. Furthermore, we use $G\ \alpha\ H$ if and only if there is some $K \in \mathbf{D}$ s.t. $G\ \alpha_K\ H$ and write $\alpha^*$ for the transitive closure of $\alpha$, taken over $K \in \mathbf{D}$.

The central relation is the $\beta$-relation $G\ \beta\ H$. It is the refinement of $\alpha^*$ that has the following additional requirement: we can find, $G_0, \ldots, G_s$ and $K_1, \ldots, K_s$ in $\mathbf{D}$ such that $G_0 = G$, $G_s = H$ and for all $j \in \{1, \ldots, s\}$ we have both, $G_{j-1}\ \alpha_{K_j}\ G_j$ and $G\ \beta\ K_j$ as well as $G\ \beta\ G_j$. This means that for a graph $H$ of some equivalence class $[G]_\beta$ of $\beta$, called $\beta$-class, we have a chain of $\alpha_K$ relations, starting at $G$ and ending at $H$, such that all graphs along this chain, as well as all $K$ used to establish the $\alpha_K$ relations, are are in $[G]_\beta$.

The core theorem of the paper is that consensus is solvable for $\mathbf{D}^\omega$ if and only if the $\beta$-classes of $\mathbf{D}$ are broadcastable, which means that for any $\beta$-class $[G]_\beta$ of $\mathbf{D}$, we have $\bigcap_{H \in [G]_\beta} \mathcal{S}(H) \neq \emptyset$.

Since processes know $\mathbf{D}$, if a process detects that some members of a $\beta$-class occurred $|V|$ times during an execution (we will say that the $\beta$-class occured $|V|$ times for brevity), it knows that the input of any process of the above intersection reached everyone in the system as a direct consequence of Theorem 3.3.2.

### 5.2.1 Impossibility

We now sketch why, if some $\beta$-class is not broadcastable, consensus is impossible. For the impossibility, it suffices to consider only graphs from the non-broadcastable $\beta$-class $[A]_\beta$, hence, for now, consider the restricted message adversary $([A]_\beta)^\omega$.

Here, a bivalent initial configuration exists because, following the argument about the validity condition and configurations where all processes start with the same input, used in the proof of Theorem 4.0.4, there are two configurations $C, C'$ that differ in valence and only in the input of one process $p$. Since $[A]_\beta$ is not broadcastable, the adversary can choose an execution where $p$ never reaches some process $q$. Here, $q$ never learns the difference between $C$ and $C'$ and thus $C, C'$ can not be univalent and of different valence.

Assuming that we arrived in a bivalent configuration $C$, we now show that not all successor configurations can be univalent. If all were univalent, by exploiting the bivalence of $C$, we have a 0-valent configuration $C'$ and a 1-valent configuration $C''$ where $C'$ is reached by applying $G' \in [A]_\beta$ to $C$ and $C''$ is reached by applying $G'' \in [A]_\beta$ to $C$ and $G' \alpha_K G''$. By the definition of $\alpha$, $C'$ and $C''$ are indistinguishable for the processes of $\mathcal{S}(K)$. By the definition of $\beta$, $K \in [A]_\beta$, hence the adversary may subsequently schedule $K$ forever.

Then, however, the processes of $\mathcal{S}(K)$ will never be able to tell $C'$ and $C''$ apart, which contradicts their stipulated differing univalence.

### 5.2.2 Possibility

For the full-information algorithm proposed by the authors, one requires three essential ingredients.

**The $\beta_i$ relation.** The first ingredient is a relation $\beta_i$ that is a step-wise refinement of $\alpha^*$. It is defined as follows: $G\ \beta_0\ H \Leftrightarrow G\ \alpha^*\ H$ and $G\ \beta_{i+1}\ H$ if $G\ \beta_i\ H$ and $\exists G_0, \dots, G_s \in [G]_{\beta_i}, \exists K_1, \dots, K_s \in [G]_{\beta_i}$ such that $G_0 = G, G_s = H$ and $\forall j \in \{1, \dots, s\}$ we have $G_{j-1}\ \alpha_{K_j}\ G_j$.

Essentially, in step $i+1$, we refine an equivalence class $[A]_{\beta_i}$ of the previous step, if no chain of $G_j, K_j$ graphs can be found in $[A]_{\beta_i}$ that relates any two graphs of $[A]_{\beta_i}$ via the $\alpha_{K_j}$-relation. It is clear, that, since we only consider finite $\mathbf{D}$, there is some integer $t$ such that $\beta_t = \beta$ and furthermore, since $\mathbf{D}$ is known to the processes, the processes are aware of $t$. We note that multiple refinements may be required to reach $\beta_t$ since in each step it can occur that a graph $H$ remains in the refined relation due to a sequence of $\alpha_{K_j}$ relations while at the same time one of the $K_j$ is removed.

**A notion of compatibility with local approximation.** The second ingredient is a notion of compatibility $\mathrm{Comp}(F)$, which collects all graphs some graph $F$ is "compatible with", intuitively in the sense that the processes that receive a message in communication graph $F$ receive the same messages in every graph $G$ of $\mathrm{Comp}(F)$. In this sense, $G$ is compatible with $F$ for these processes, because, as far as they are concerned, $G$ could have been the actual communication graph. This is defined as $F \subseteq G$ and if $(u, v) \in F$ and $(w, v) \in G$ then $(w, v) \in F$. Note that this definition by itself is not very useful, since, under the assumption that the $\beta$-classes are broadcastable, every node has at least one incoming edge in every communication graph (it is not hard to see that a communication graph $G$ where a node has no in-edge has $\mathcal{S}(G) = \emptyset$), and thus in this case $\mathrm{Comp}(F) = \{F\}$. It becomes useful with the parametrization

$$\mathrm{Comp}_K(F) = \mathrm{Comp}(\mathrm{In}^e_{\mathcal{S}(K)}(F)) \ ,$$

i.e., when $F$ is just the specific set of in-edges of $\mathcal{S}(K)$. This notation describes those graphs that are, for the nodes of $\mathcal{S}(K)$, compatible with $G$. To see this, consider that in a broadcastable, i.e., rooted graph, the set $\mathrm{In}^e_{\mathcal{S}(K)}(G)$ is exactly the subgraph of $G$ induced by the vertex set of $\mathcal{S}(K)$ (c.f. Definition 3.2.1). It is not hard to see that $\mathrm{Comp}(F) \subseteq \mathrm{Comp}_K(F)$.

We can combine this notion with the $\beta_i$ relation to state the crucial observation that $G\ \beta_i\ K$ implies that $\mathrm{Comp}_K(G) \subseteq [G]_{\beta_{i+1}}$ (and hence $\mathrm{Comp}(G) \subseteq [G]_{\beta_{i+1}}$) since for any $H \in \mathrm{Comp}_K(G)$, we have $H\ \alpha_K\ G$ (essentially by definition; see Lemma 6.3 of [CGP15] for details). Clearly, we also have $\mathrm{Comp}_K(G) \subseteq [G]_{\alpha^*}$

**Bounds, suitable for pigeon-hole arguments.** The third ingredient is a collection of bounds, both for the counting of the occurrence of $\beta_i$-classes and the round $r_{\max}$ in which the algorithm has accumulated enough information to terminate. For counting $\beta_i$-classes, we use $k_i = |\mathbf{D}|k_{i+1} + 2|V||\mathbf{D}|$, where $k_t = |V|$. Here, $t$ again is s.t. $\beta_t = \beta$. We set $r_{\max} = |\mathbf{D}|k_0 + 2|V||\mathbf{D}|$.

Note that by round $r_{\max}$, by the pigeon-hole principle, some graph $G \in \mathbf{D}$ occurred in at least $k_0 + 2|V|$ different rounds. This leads to an inductive reasoning about the algorithm, the base case being that all $\alpha^*$ classes of the first $|\mathbf{D}|k_0$ rounds, i.e., the $\alpha^*$-class, the communication graph $G_r$ of round $r$ belongs to, can be discovered by any process, the induction step that from the detection of the first $k_i$ occurrences of $\beta_i$, we may derive the detection of the first $k_{i+1}$ occurrences of $\beta_{i+1}$ for $i \in [0, t-1]$.

We sketch the details of how to show this below, for now, observe that this would indeed imply the existence of an algorithm. Since any $p$ detects all $\alpha^*$-classes of the first $|\mathbf{D}|k_0$ rounds, and there can be at most $|\mathbf{D}|$ $\alpha^*$-classes, $p$ detects the first $\alpha^*$-class that occurs for $k_0$ rounds. We could then apply the inductive argument to ultimately show that each process can find the $\beta_t = \beta$ classes that occurred at least $k_t$ times for the first time. Since there are $k_t = |V|$ such occurrences, the processes can then decide on some deterministic choice on the values that were broadcast in this $\beta$-class.

**Base case:** Why we can detect the $\alpha^*$-classes of the first $|\mathbf{D}|k_0$ rounds:

It is important to recall that, from their knowledge of $\mathbf{D}$, processes can calculate the $\beta_i$-class of any graph $G \in \mathbf{D}$ a priori for all $0 \leqslant i \leqslant t$. The task of the full-information protocol is to use the result of this calculation and place a (partially) observed round graph $X$ in its correct $\beta_i$-class.

As argued before, some graph $K$ occurred at least $2|V|$ times after round $r_1 = |\mathbf{D}|k_0$. Let $r_2$ denote the round where $K$ occurred $|V|$ times and $r_3$ denote the round where $K$ occurred $2|V|$ times. By Theorem 3.3.2, in round $r_2$, any process $p$ of $\mathcal{S}(K)$ has received the round $r_1$ state of every $q$ of $\mathcal{S}(K)$, which is then transmitted to the remaining system by round $r_3$.

This already suffices to calculate the $\alpha^*$-classes: As stated above, $\mathrm{Comp}_K(G) \subseteq [G]_{\alpha^*}$. Consider some round $r \leqslant r_1$ where a process $p$ has an approximation $X$ of the actual round graph $G$. Since $p$ has the state of all members of $\mathcal{S}(K)$, $\mathrm{Comp}(X) \subseteq \mathrm{Comp}_K(G)$, and therefore $\mathrm{Comp}(X) \subseteq [G]_{\alpha^*}$ (clearly, $\mathrm{Comp}_K(G) \neq \emptyset$ since $G \in \mathrm{Comp}_K(G)$). We observe that $p$ does not necessarily have a precise knowledge of $K$.

**Induction step:** Detecting the first $k_i$ $\beta_i$-classes enables detecting the first $k_{i+1}$ $\beta_{i+1}$-classes.

Assume that we have detected the $\beta_i$-class $[A]_{\beta_i}$ that occurred $k_i$ times for the first time. Let $r_1$ denote the end of the round where $\beta_i$ occurred $|\mathbf{D}|k_{i+1}$ times. Since $k_i = |\mathbf{D}|k_{i+1} + 2|V||\mathbf{D}|$, some graph $K$ occurred $2|V|$ times after $r_1$. As above, every

process hence has the state of $S(K)$ for every round $\leqslant r_1$. Let $X$ be some approximation of a round $r \leqslant r_1$ with actual graph $G$ for which we want to determine the $\beta_{i+1}$-class. It follows that $\mathrm{Comp}_K(X) = \mathrm{Comp}_K(G)$. Since $K, G \in [A]_{\beta_i}$, as we have mentioned before, $p$ can determine $\mathrm{Comp}(X) \subseteq \mathrm{Comp}_K(X) = \mathrm{Comp}_K(G) \subseteq [X]_{\beta_{i+1}}$. We remark that, again, it is not even necessary for $p$ to know the graph $K$ itself.

## 5.3 Basic Notations and Definitions

As mentioned in the introduction, we consider closed message adversaries here, which we define below. Roughly speaking, this states that every infinite sequence of prefixes, such that each element of the sequence is the prefix of its successor as well as the prefix of an admissible communication pattern, leads to an admissible communication pattern.

**Definition 5.3.1.** *A message adversary $\mathsf{MA}$ is* limit-closed *(closed for short) if the limit $\sigma = \lim_{r\to\infty} \sigma_r$ of every infinite sequence of prefixes $(\sigma_i)_{i>0}$ where, for $i > 0$, $\sigma_i = \sigma_{i+1}|_i$ and $\sigma_i = \rho|_i$ for some $\rho \in \mathsf{MA}$ satisfies $\sigma \in \mathsf{MA}$.*

We note that every oblivious message adversary is closed, whereas stabilizing message adversaries are usually non-closed: For example, the message adversary $\mathsf{MA}_C$, which guarantees that eventually the complete graph occurs in every sequence, does not allow communication patterns where the complete graph $C$ never occurs. However, the latter is the limit $\lim_{r\to\infty} \sigma_r$, where $\sigma_r \in \mathsf{MA}_C$ is such that the complete graph occurs in $\sigma_r$ for the first time in round $r + 1$.

Throughout this chapter, we will argue about the $\sim^*$ relation, i.e., the transitive closure of the indistinguishability relation introduced in Chapter 3 (called $\alpha^*$-relation in [CGP15] and the previous section). We note that $\sim^*$ is an equivalence relation and, given some set $S$ of communication patterns of the same range, we denote by $[\sigma]_{\sim^*}$, or simply $[\sigma]$ for brevity, the equivalence class of $\sigma$ wrt. $\sim^*$ on $S$. Given $S' \subseteq S$ with $\sigma \in S'$, the subclass $[\sigma]_{S'}$ of $\sigma$ is the equivalence class of $\sigma$ on $S'$; we will sloppily write $[\sigma]_{S'} \subseteq [\sigma]$ in this case. Note carefully that the transitive closure $\sim^*$ of $[\sigma]_{S'}$ runs over sequences in $S'$ only. Hence, there may be $\rho \in S'$ with $\rho \in [\sigma]$ but $\rho \notin [\sigma]_{S'}$, namely, when every path between $\sigma$ and $\rho$ contains some sequence in $S \setminus S'$.

In order to avoid confusion, we will refrain from using the shorthand notation $\{1, \dots, k\} = [k]$ in the remainder of this chapter. We recall from Chapter 3 that for an infinite sequence (or the finite prefix of an infinite sequence) $\sigma$, its kernel, $\mathrm{Ker}(\sigma)$, denotes the set of processes that are heard by everyone over the course of $\sigma$. For a set (such as an equivalence class) of infinite sequences (or a set of $r$-round prefixes) $S$, we denote by

$$\mathrm{Ker}(S) = \{\mathrm{Ker}(\sigma) : \sigma \in S\} \ .$$

the set of the kernels of its members. We denote the kernel-intersection of $S$ as

$$\mathrm{KI}(S) = \bigcap_{x\in S} \mathrm{Ker}(x) \ .$$

In order to avoid notational clutter, we will abbreviate $[\sigma]_{\sim^*} = [\sigma]$, $\mathrm{Ker}([\sigma]_{\sim^*}) = \mathrm{Ker}[\sigma]$, and $\mathrm{KI}([\sigma]_{\sim^*}) = \mathrm{KI}[\sigma]$ throughout this chapter.

We commence by rephrasing the crucial dependency of decision values and $\mathrm{Ker}(\sigma)$. Note that this is a similar but stronger statement than Theorem 4.0.4.

**Theorem 5.3.2.** *Let $\mathsf{MA}$ be an arbitrary message adversary. For an initial configuration $C_0$, for all $\sigma \in \mathsf{MA}$, for all $p_i \in \Pi$, the decision $y_i$ in execution $\varepsilon = \langle C_0, \sigma\rangle$ of any correct consensus algorithm satisfies $y_i = x_j$ for some $p_j \in \mathrm{Ker}(\sigma)$.*

*Proof.* Suppose there is an initial configuration $C_0$ and a $\sigma \in \mathsf{MA}$, such that, in some correct consensus algorithm $\mathcal{A}$, some process decides $v$ in execution $\langle C_0, \sigma\rangle$ where $v \neq x_i$ for any $p_i \in \mathrm{Ker}(\sigma)$. By validity, the set $\overline{K} = \{p_i \in \Pi \setminus \mathrm{Ker}(\sigma) \mid x_i = v\}$ is non-empty. Assuming some arbitrary ordering on $\overline{K} = \{p_{i_1}, \dots, p_{i_k}\}$, let $C_0^0 = C_0$ and for $0 < \ell \leqslant |\overline{K}| = k$, let $C_0^\ell$ be the same as $C_0^{\ell-1}$ except that $x_{i_\ell} \neq v$ in $C_0^\ell$. We show by induction that, for any $\ell$, some process decides $v$ in $\varepsilon_\ell = \langle C_0^\ell, \sigma\rangle$. But then, $\mathcal{A}$ violates validity, because in $C_0^k$, $x_i \neq v$ for every process $p_i \in \Pi$.

The induction base follows from the initial assumption. For the induction step from $\ell - 1$ to $\ell$, we observe that since $\overline{K} \subseteq \Pi \setminus \mathrm{Ker}(\sigma)$, there is some process $p_i$ such that, for any round $r$, $(p_{i_\ell}, 0) \not\rightsquigarrow_\sigma (p_i, r)$.

By construction, therefore $\varepsilon_{\ell-1} \sim_{p_i} \varepsilon_\ell$. The induction hypothesis asserts that some process decides $v$ in $\varepsilon_{\ell-1}$ and thus, by agreement, $p_i$ decides $v$ in $\varepsilon_{\ell-1}$ and hence in $\varepsilon_\ell$ as well. □

Because of Theorem 5.3.2, it makes sense to only consider message adversaries $\mathsf{MA}$ where for each $\sigma \in \mathsf{MA}$ we have $\mathrm{Ker}(\sigma) \neq \emptyset$. Perhaps not surprisingly, though, this requirement alone is insufficient for solving consensus. Investigating precisely what is required additionally is the topic of the remaining chapter.

## 5.4 Necessity of an Eventually Common Kernel

In this section, we prove the "only if"-direction of Theorem 5.0.1, showing that consensus is impossible under a message adversary that contains a communication pattern $\sigma$ with $\mathrm{KI}[\sigma|_r] = \bigcap_{x \in [\sigma|_r]} \mathrm{Ker}(x) = \emptyset$ for all rounds $r$. More specifically, we show in Lemma 5.4.5 that a particular subclass $X \subseteq [\sigma]$ of communication patterns with $\bigcap_{x \in X} \mathrm{Ker}(x) = \emptyset$ exists, which implies the impossibility of consensus as shown in Lemma 5.4.6. To prove Lemma 5.4.5, we apply König's Infinity Lemma to a tree whose nodes on level $r$ are subclasses of $[\sigma|_r]$ that consist of prefixes $\rho$ such that $\mathrm{Ker}(\rho)$ is a subset of an "associate kernel" in some fixed set $S = \{K_1, \dots, K_k\}$. The infinite path in the tree guaranteed by König's Infinity Lemma defines a non-empty subclass $X$ of $[\sigma]$, and since the associate kernels satisfy $\bigcap_{i=1}^k K_i = \emptyset$, this implies $\bigcap_{x \in X} \mathrm{Ker}(x) = \emptyset$, as required.

**Lemma 5.4.1** (König's Infinity Lemma, cf. [Die06, Chapter 6])**.** *Let $V_1, V_2, \ldots$ be an infinite sequence of disjoint non-empty finite sets, and let $G$ be a graph on their union. Assume that every vertex $v$ in a set $V_r$ with $r > 1$ has a neighbor $f(v)$ in $V_{r-1}$. Then $G$ contains an infinite path $v_1 v_2 \ldots$ with $v_r \in V_r$ for all $r$.*

We start with the definition of *kernel-restricted* subclasses $[\sigma]^K$ resp. $[\sigma]^{\subseteq K}$, where all members must have a kernel that is equal to resp. a subset of an element of the non-empty set of kernels $K = \{K_1, \ldots, K_k\}$:

**Definition 5.4.2** (Kernel-restricted subclasses)**.** *Given the equivalence class $[\sigma]_S$ of a communication pattern $\sigma$ over the set $S$, and some non-empty set of non-empty kernels $\emptyset \neq K = \{K_1, \ldots, K_k\} \subseteq \mathcal{P}(\Pi)$, the kernel-restricted subclass $[\sigma]^K \subseteq [\sigma]_S$ of $\sigma$ is the subclass $[\sigma]_{S'}$ of $\sigma$ over the set $S' = \{\rho \in S : \mathrm{Ker}(\rho) \in K\}$. Similarly, $[\sigma]^{\subseteq K} \subseteq [\sigma]$ is the subclass $[\sigma]_{S''}$ over the set $S'' = \{\rho \in S : \exists K_i \in K \text{ s.t. } \mathrm{Ker}(\rho) \subseteq K_i\}$.*

Note that $[\sigma]^K = \emptyset$ if $\mathrm{Ker}(\sigma) \notin K$ (although this does not happen in Lemmas 5.4.3 to 5.4.5 below), and that there may be $\rho \in [\sigma]$ with $\mathrm{Ker}(\rho) \in K$ with $\rho \notin [\sigma]^K$, which happens if every path connecting $\sigma \sim \rho$ in $[\sigma]$ contains at least one prefix not in $S'$.

**Lemma 5.4.3.** *Let MA be a message adversary that contains some $\sigma$ s.t., for all rounds $r$, we have $\mathrm{KI}[\sigma|_r] = \emptyset$. Then, for some finite $k > 0$, there is a non-empty set $K = \{K_1, \ldots, K_k\} \subseteq \mathcal{P}(\Pi)$ of* associate kernels*, such that $\bigcap_{i=1}^k K_i = \emptyset$ and $[\sigma|_r] = [\sigma|_r]^K$ for infinitely many rounds $r = r_1, r_2, \ldots$.*

*Proof.* For an arbitrary round $r > 0$, let $g([\sigma_r]) := \{\kappa \subseteq \Pi \colon \exists \rho \in [\sigma_r] \text{ with } \mathrm{Ker}(\rho) = \kappa\}$. Since $\Pi$ is a finite set, the power set $\mathcal{P}(\Pi)$ is a finite set as well. By the pigeonhole principle, there is some $K = \{K_1, \ldots, K_k\} \subseteq \mathcal{P}(\Pi)$ such that, for infinitely many rounds $r$, $g([\sigma_r]) = K$, hence $[\sigma|_r] = [\sigma|_r]^K$. Note that, since obviously $\sigma|_r \in [\sigma|_r]$ for every $r > 0$, we have $\mathrm{Ker}(\sigma|_r) \in K$. By the assumption that, for all $r$, $\mathrm{KI}[\sigma|_r] = \emptyset$, we also have $\bigcap_{i=1}^k K_i = \emptyset$. □

**Lemma 5.4.4.** *Let MA be a message adversary. If there exists some $\sigma \in \mathsf{MA}$, $K = \{K_1, \ldots, K_k\} \subseteq \mathcal{P}(\Pi)$ with $[\sigma|_r] = [\sigma|_r]^K$ for infinitely many rounds $r$, then there is an infinite sequence $\mathcal{V} = V_1, V_2, \ldots, V_i, \ldots$ of sets $V_i$ of kernel-restricted subclasses of $\sigma|_i$ in $[\sigma|_i]^{\subseteq K}$, such that, for all $i \geqslant 1$, each of the following holds:*

*(1)* $V_i \neq \emptyset$

*(2)* $V_i = \{\nu_1, \ldots, \nu_{m(i)}\}$ *for a finite $m(i) > 0$ s.t., for $1 \leqslant j \leqslant m(i)$: $\sigma|_i \in \nu_j \subseteq [\sigma|_i]^{\subseteq K}$*

*(3) Each $\nu \in V_{i+1}$ has a neighbor $\nu' = f_{i+1}(\nu) \in V_i$*

*Proof.* Initializing $V_i = \emptyset$ for every $i \geqslant 1$, we construct $V_i$, starting from $i = 1$, as follows: For each of the infinitely many indices $i = r$ where $[\sigma|_r] = [\sigma|_r]^K$, we set $V_i = \{[\sigma|_i]^K\}$; note that $\mathrm{Ker}(\sigma|_r) \in K$ in this case, so (1) and (2) hold for $V_i$. Moreover, for all $1 \leqslant j < i$, we add to $V_j$ the set $\{\rho|_j : \rho \in [\sigma|_i]^K\}$. As $\mathrm{Ker}(\rho|_j) \subseteq \mathrm{Ker}(\rho)$, and since $\rho \sim \tau$ implies also $\rho|_j \sim \tau|_j$, (1) and (2) continue to hold for $V_j$. Moreover, for $\nu \in V_{i+1}$, we define $f(\nu) = \nu|_i$, which secures (3). Thus, the infinite sequence of sets $\mathcal{V}$ with properties (1)–(3) exists, as claimed. □

**Lemma 5.4.5.** *Let MA be a message adversary that contains some $\sigma \in \mathsf{MA}$ such that, for all $r$, we have $\mathrm{KI}[\sigma|_r] = \emptyset$. Then, there is a set of associate kernels $K = \{K_1, \ldots, K_k\}$ with $\bigcap_{i=1}^{k} K_i = \emptyset$ and a non-empty kernel-restricted subclass $X \subseteq [\sigma]^{\subseteq K}$ with $\bigcap_{x \in X} \mathrm{Ker}(x) = \emptyset$.*

*Proof.* We take $\sigma$ and apply to it Lemma 5.4.3 and then Lemma 5.4.4. In this manner, we obtain the set of associate kernels $K$ and an infinite tree spanning the members of the infinite sequence of sets $V_1, V_2, \ldots$ via the neighbor functions $f_i : V_i \to V_{i-1}$. König's Infinity Lemma ensures that there is an infinite path in this tree, which implicitly defines the sought subclass $X$ as an element of $\lim_{i \to \infty} V_i$: For every $\rho \in X$, we have $\rho|_r \sim \sigma|_r$ for every $r \geqslant 1$, and hence $\rho \sim \sigma$. □

**Lemma 5.4.6.** *Consensus is impossible under a message adversary MA with $\sigma \in \mathsf{MA}$ such that some non-empty subclass $X \subseteq [\sigma]$ satisfies $\bigcap_{x \in X} \mathrm{Ker}(x) = \emptyset$.*

*Proof.* We show that the existence of some algorithm $\mathcal{A}$ that solves consensus under MA would lead to a contradiction. Since $X \neq \emptyset$, for some $k > 0$, there is a multiset of processes $\{p_{i_1}, \ldots, p_{i_{k-1}}\}$, which may be empty (if $k = 1$), and a non-empty multiset of communication patterns $Y = \{\sigma_1, \ldots, \sigma_k\} \subseteq X$ with $\sigma_j \in [\sigma]$ for $1 \leqslant j \leqslant k$ and $\bigcap_{\rho \in Y} \mathrm{Ker}(\rho) = \emptyset$, such that $\sigma_1 \sim_{p_{i_1}} \sigma_2 \ldots \sigma_{k-1} \sim_{p_{i_{k-1}}} \sigma_k$.

Let $C_0^v$ be the input assignment where $x_i = v$ for all processes $p_i$ and let $C_0^{\overline{v}}$ be the input assignment where $x_i = \overline{v}$ for all processes $p_i$, for some $v \neq \overline{v}$. By validity, in execution $\langle C_0^v, \sigma_1 \rangle$ every process running $\mathcal{A}$ decides $v$, while in $\langle C_0^{\overline{v}}, \sigma_1 \rangle$ they decide $\overline{v}$. Since input assignments are not restricted in any way, toggling the input values of $p_1, p_2, \ldots$ from $v$ to $\overline{v}$, one after the other, reveals that there are input assignments $C_0', C_0''$ that differ only in the input value of a single process $p_i$, yet every process running $\mathcal{A}$ decides $v$ in $\varepsilon_1^v = \langle C_0', \sigma_1 \rangle$ and $\overline{v}$ in $\varepsilon_1^{\overline{v}} = \langle C_0'', \sigma_1 \rangle$.

A simple induction shows that $v$ is decided in $\varepsilon_j^v = \langle C_0', \sigma_j \rangle$ and $\overline{v}$ is decided in $\varepsilon_j^{\overline{v}} = \langle C_0'', \sigma_j \rangle$ for $1 \leqslant j \leqslant k$. The base case, $\ell = 1$, was already shown above. For the step from $\ell$ to $\ell + 1$ with $1 \leqslant \ell < k$, we have by hypothesis that $v$ was decided in $\varepsilon_\ell^v$ and $\overline{v}$ was decided in $\varepsilon_\ell^{\overline{v}}$. Since $\sigma_\ell \sim_{p_{i_\ell}} \sigma_{\ell+1}$, we have $\varepsilon_\ell^v \sim_{p_{i_\ell}} \varepsilon_{\ell+1}^v$ and $v$ is also decided in $\varepsilon_{\ell+1}^v$. A similar argument shows that $\overline{v}$ is decided in $\varepsilon_{\ell+1}^{\overline{v}}$.

We conclude the proof by showing that $p_i \in \mathrm{Ker}(\sigma_j)$ for $1 \leqslant j \leqslant k$, and hence $p_i \in \mathrm{Ker}(\rho)$ for any $\rho \in Y$, which contradicts $\bigcap_{\rho \in Y} \mathrm{Ker}(\rho) = \emptyset$. Suppose, for some $j$, $p_i \notin \mathrm{Ker}(\sigma_j)$.

Hence there is some $p_k$ with $p_i \not\rightsquigarrow_{\sigma^j} p_k$. Since $C_0'$ and $C_0''$ are the same except for the input of $p_i$, $\varepsilon_j^v \sim_{p_k} \varepsilon_j^{\overline{v}}$, and $p_k$ decides the same in $\varepsilon_j^v$ and in $\varepsilon_j^{\overline{v}}$. This, however, contradicts our previous statement that $v$ is decided in $\varepsilon_j^v$ and $\overline{v}$ is decided in $\varepsilon_j^{\overline{v}}$ for some $v \neq \overline{v}$. □

### A note on bivalence arguments

At this point, the reader might wonder why we did not resort to a bivalence argument, as introduced in [FLP85] and used heavily in the literature (e.g. in the very closely related papers [SW89, LM95, MR02, SWK09, FG11, CGP15]) to establish our impossibility result. In a nutshell, in the case of binary consensus, bivalence proofs establish impossibility by inductively constructing a run where every reached configuration is bivalent. Reachable configurations are classified according to whether only 0-decided resp. 1-decided configurations are reachable from it, in which case the configuration is called 0-valent resp. 1-valent, or whether both a 0-decided and a 1-decided configuration are reachable from it, in which case the configuration is called bivalent. Note carefully that the agreement property implies that no process can have decided in a bivalent configuration, as a single decision, say, to 0, would make the configuration already 0-valent.

In the bivalence induction proof, it is first established that not all initial configurations can be 0-valent or 1-valent. Then, under the hypothesis that the reached round $r$ configuration is bivalent, it is shown that not all round $r+1$ configurations can be univalent. This results in a forever bivalent run, in which no process can have decided. Care must be taken, however, to also prove that the so constructed run is also admissible, as the processes must decide only in an admissible run.

The reason why we cannot use such an argument in our setting, and need to resort to König's Lemma instead, is that the induction step might lead to a dead end later on: there is no a priori guarantee that the bivalent successor chosen in some step is one that allows infinite repetition, i.e., the construction of an infinite admissible suffix. Technically, what would be needed in the induction step to ensure this is that two configurations that are currently indistinguishable for some process can be extended in a way that remains indistinguishable forever for this process. As the only thing we know about our message adversary is that it is closed and eventually guarantees a non-empty kernel intersection, however, it is not clear how to infer sufficient information on the possible communication graphs generated in all admissible suffixes to guarantee this.

## 5.5 Sufficiency of an Eventually Common Kernel

We now show the "if"-direction of Theorem 5.0.1, by introducing Algorithm 5.1. This algorithm solves consensus under any message adversary $\mathsf{MA}$ that guarantees, for every $\sigma \in \mathsf{MA}$, that there is a round $r$ and a non-empty set $K \subseteq \Pi$ such that $K = \mathrm{KI}[\sigma|_r] = \bigcap_{x \in [\sigma|_r]} \mathrm{Ker}(x)$. Note that this algorithm could stop operating immediately after decision, as the latter happens in the same round at all processes.

Essentially, each process $p_i$ executing the algorithm attempts to send a local estimation of the communication pattern, stored in array $\mathsf{E}$, along with its own input value $x[i]$ to all other processes. Here, the $k^{\text{th}}$ entry of $\mathsf{E}$, $\mathsf{E}[k]$, contains the local estimate of the round $k$ communication graph. On reception of a round $r$ message from some process $p_j$, including $p_j = p_i$, $p_i$ stores the input of $p_j$ in $x[j]$ and adds the edge $(p_j \to p_i)$ to $\mathsf{E}[r]$ since it could only have received the message if $(p_j \to p_i) \in G_r$. Note that this implies that in every round $r$, $(p_i \to p_i)$ is added to $\mathsf{E}[r]$, as we assumed that every process receives a message from itself each round. Process $p_i$ then proceeds to merge its local estimates $\mathsf{E}$ with the ones received and then calculates the set $S$ of all communication patterns $G_1, \ldots, G_r$ that it considers possible. It does this by checking which communication pattern prefixes, allowed by $\mathsf{MA}$, are in accordance with what $p_i$ observed so far. Finally, $p_i$ picks an arbitrary prefix $\rho$ of $S$ and checks whether all members of the equivalence class of $\rho$ have a non-empty intersection $K$ of their kernels. If this is the case, $p_i$ decides on the input of the process in $K$ with the largest identifier. We note that the equivalence class of $\rho$ can be computed from the specification of $\mathsf{MA}$, which is, according to the system model, known to the processes. Note that there are only finitely many communication graphs and hence finitely many round $r$ prefixes for finite $r$.

In the following proof of the correctness of Algorithm 5.1, we use $\mathsf{var}_i^r \in \mathsf{state}_r(p_i)$ to denote the value of variable $\mathsf{var}$, held by process $p_i$ at the end of its round $r$ computation. This is clearly the same as the value of $\mathsf{var}$ after it was written to the last time in round $r$, so if the last write to $\mathsf{var}$ occurs in line $\ell$, $\mathsf{var}_i^r$ is the value of $\mathsf{var}$ at process $p_i$ after $p_i$ finished line $\ell$ for the last time in round $r$.

We start with a few technical results, which essentially assert the correctness of the local estimates. Lemma 5.5.1 establishes that the set $\mathsf{E}$ approximates edges in the communication graph faithfully if there was an appropriate influence.

**Lemma 5.5.1.** $(p_j, r') \rightsquigarrow_\sigma (p_i, r) \Leftrightarrow \mathrm{In}_{G_{r'}}(p_j) \subseteq \mathsf{E}_i^r[r']$.

*Proof.* For the "$\Rightarrow$" direction, we show inductively for $r' < \ell \leqslant r$ that $(p_j, r') \rightsquigarrow_\sigma (p_k, \ell)$ implies $\mathrm{In}_{G_{r'}}(p_j) \subseteq \mathsf{E}_k^\ell[r']$.

For $\ell = r' + 1$, $(p_j, r') \rightsquigarrow_\sigma (p_k, \ell)$ implies, by definition of the $\rightsquigarrow_\sigma$ relation, $(p_j \to p_k) \in G_{r'+1}$, i.e., $p_k$ receives the round $r'+1$ message of $p_j$. By Line 10, $\mathrm{In}_{G_{r'}}(p_j) \subseteq \mathsf{E}_j^{r'}[r']$. Since $p_k$ received $\mathsf{E}_j^{r'}[r']$ from $p_j$ via its round $r'+1$ message, $p_k$ incorporates $\mathsf{E}_j^{r'}[r']$ into its own $\mathsf{E}_k^{r'+1}[r']$ in Line 13.

For $r' + 1 < \ell \leqslant r$, we assume that $(p_j, r') \rightsquigarrow_\sigma (p_k, \ell - 1)$ implies $\mathrm{In}_{G_{r'}}(p_j) \subseteq \mathsf{E}_k^{\ell-1}[r']$. If for some $p_m$, $(p_j, r') \rightsquigarrow_\sigma (p_m, \ell)$, by definition, $(p_k \to p_m) \in G_\ell$ for some $p_k$ with $(p_j, r') \rightsquigarrow_\sigma (p_k, \ell - 1)$. By hypothesis, $\mathrm{In}_{G_{r'}}(p_j) \subseteq \mathsf{E}_k^{\ell-1}[r']$, hence $p_m$ receives $\mathsf{E}_k^{\ell-1}[r']$ at the beginning of round $\ell$ and incorporates $\mathrm{In}_{G_{r'}}(p_j)$ into $\mathsf{E}_m^\ell[r']$ in Line 13 of its round $\ell$ computation.

The "$\Leftarrow$" direction holds trivially if $p_i = p_j$. If $p_i \neq p_j$, then, according to Line 13, $p_i$ can only learn about $\mathrm{In}_{G_{r'}}(p_j)$ if there is a chain of messages, starting at $p_j$ no earlier than

**Algorithm 5.1:** Consensus algorithm for a closed message adversary, code for process $p_i$

**Initialization**:
1 $\mathsf{x}[i] \leftarrow x_i$
2 $\mathsf{x}[j] \leftarrow \perp$ for $j \neq i$
3 $\mathsf{E}[0] \leftarrow \emptyset$
4 $r \leftarrow 1$

**Transmit round $r$ messages**:
5 Attempt to send $\langle \mathsf{E}, \mathsf{x} \rangle$ to all
6 Receive $\langle \mathsf{E}_j, \mathsf{x}_j \rangle$ from all $p_j$ with $(p_j \to p_i) \in G_r$

**Round $r$ computation**:
7 **foreach** $p_j$ *from which* $p_i$ *received a message in round* $r$ **do**
8 | **foreach** $k$ *with* $\mathsf{x}_j[k] \neq \perp$ **do**
9 | | $\mathsf{x}[k] \leftarrow \mathsf{x}_j[k]$
10 | Add $(p_j \to p_i)$ to $\mathsf{E}[r]$
11 | **if** $r > 1$ **then**
12 | | **for** $r'' \in \{1, \dots r-1\}$ **do**
13 | | | $\mathsf{E}[r''] \leftarrow \mathsf{E}[r''] \cup \mathsf{E}_j[r'']$
14 **for** $r' \in \{1, \dots r\}$ **do**
15 | $\mathsf{V}_{r'} \leftarrow \{p_j \in \Pi \colon \exists p_k \in \Pi \text{ s.t. } (p_k \to p_j) \in \mathsf{E}[r']\}$
16 Let $\mathrm{In}_{G_{r'}}(\mathsf{V}_{r'})$ denote the edges $(u \to v) \in G_{r'}$ with $v \in \mathsf{V}_{r'}$
17 $\mathsf{S} \leftarrow \{\sigma|_r = G_1, \dots, G_r \colon \sigma \in \mathsf{MA} \text{ and } \mathrm{In}_{G_{r'}}(\mathsf{V}_{r'}) = \mathsf{E}[r'] \text{ for all } 1 \leqslant r' \leqslant r\}$
18 Pick an arbitrary $\rho \in \mathsf{S}$
19 **if** $y_i = \perp$ *and there exists* $K \neq \emptyset$ *s.t.* $K = \mathrm{KI}[\rho]$ **then**
20 | $m \leftarrow \max\{j : p_j \in K\}$
21 | $y_i \leftarrow \mathsf{x}[m]$ /* decide */
22 $r \leftarrow r + 1$

the end of round $r'$ and ending at $p_i$ before its round $r$ computing step. By definition, hence $(p_j, r') \rightsquigarrow_\sigma (p_i, r)$. □

Lemma 5.5.2 shows that any $\mathsf{E}_k[r']$ is an under-approximation of $G_{r'}$, i.e., it does not contain any fabricated edges.

**Lemma 5.5.2.** *If* $(p_j \to p_i) \in \mathsf{E}^r_k[r']$ *then (1)* $(p_j \to p_i) \in G_{r'}$ *and (2)* $(p_i, r') \rightsquigarrow_\sigma (p_k, r)$.

*Proof.* (1) Suppose $(p_j \to p_i) \in \mathsf{E}^r_k[r']$ but $(p_j \to p_i) \notin G_{r'}$. Since $(p_j \to p_i)$ could only be added to $\mathsf{E}^r_k[r']$ through Line 10 or Line 13, we have $(p_j \to p_i) \in \mathsf{E}^{r'}_i[r']$. Since Line 10 is guarded by Line 7, $(p_j \to p_i) \in \mathsf{E}^{r'}_i[r']$ can only happen when $p_i$ received a message from $p_j$ in round $r'$, which implies that $(p_j \to p_i) \in G_{r'}$.

(2) If $(p_j \to p_i) \in \mathsf{E}^r_k[r']$, since $(p_j \to p_i) \in \mathrm{In}_{G_{r'}}(p_i)$, because during the loop of Line 7, every in-edge is added in Line 10, $\mathrm{In}_{G_{r'}}(p_i) \subseteq \mathsf{E}^{r'}_k[r']$. By Corollary 5.5.1 hence $(p_i, r') \rightsquigarrow_\sigma (p_k, r)$. □

The above lemmas can be combined to the following Corollary 5.5.3, which shows that if $\mathsf{E}_k^r[r']$ contains some non-empty fraction of the incoming round $r'$ edges of some process $p_i$, then it actually contains all the incoming round $r'$ edges of $p_i$ and vice-versa. Furthermore, this is equivalent to the existence of an influence from $p_i$ in round $r'$ to $p_k$ in round $r$.

**Corollary 5.5.3.** *The following are all equivalent: (1)* $(p_j \to p_i) \in \mathsf{E}_k^r[r']$*, (2)* $\mathrm{In}_{G_{r'}}(p_j) \subseteq \mathsf{E}_k^r[r']$*, (3)* $(p_i, r') \rightsquigarrow_\sigma (p_k, r)$*.*

Corollary 5.5.3 can be used to show that $\mathsf{S}_i^r$ indeed contains all execution prefixes that are indistinguishable from the current execution prefix, as expressed in Corollary 5.5.4.

**Corollary 5.5.4.** *In every round $r$ of an execution $\langle C_0, \sigma\rangle$, the following holds:* $\mathsf{S}_i^r = \{\rho|_r \colon \rho \in \mathsf{MA} \text{ and } \rho|_r \sim_{p_i} \sigma|_r\}$.

*Proof.* Since we assume self-loops in every communication graph, by Corollary 5.5.3, for any processes $p_i, p_j, p_k$, an in-edge $(p_k \to p_j)$ to $p_j$ is present in $\mathsf{E}_i^r[r']$ exactly when $(p_j, r') \rightsquigarrow_\sigma (p_i, r)$. Hence, $\mathsf{S}_i^r$ contains exactly those sequences $\rho_r = G_1, \ldots, G_r$ where each $G_\ell$ matches the in-neighborhood for those nodes $p_m$ that satisfy $(p_m, \ell) \rightsquigarrow_\sigma (p_i, r)$. By definition, these are precisely those prefixes $\rho|_r$ of $\rho \in \mathsf{MA}$ where $\mathsf{view}_{\rho|_r}(p_i) = \mathsf{view}_{\sigma|_r}(p_i)$ and thus $\rho|_r \sim_{p_i} \sigma|_r$. □

Finally, from the assumption that the specification of $\mathsf{MA}$ is known to the processes, and since the finite number of processes implies a finite number of communication graphs, which, in turn, implies a finite number of round $r$ prefixes, we have the following Corollary 5.5.5.

**Corollary 5.5.5.** *The equivalence class $[\sigma|_r]$ is computable a priori, for every round $r$.*

We are now ready to show the correctness of Algorithm 5.1.

**Lemma 5.5.6.** *Algorithm 5.1 solves consensus under every message adversary $\mathsf{MA}$ that ensures, for all $\sigma \in \mathsf{MA}$, that* $\mathrm{KI}[\sigma|_r] \neq \emptyset$ *for some round $r$.*

*Proof.* Pick an arbitrary sequence $\sigma \in \mathsf{MA}$ and fix some input assignment $C_0$. We show that Algorithm 5.1 satisfies all properties of the consensus specification in the execution $\langle C_0, \sigma\rangle$. Let $t$ be the earliest round such that $\mathrm{KI}[\sigma|_t] \neq \emptyset$, i.e., for all $t' < t$ we have $\mathrm{KI}[\sigma|_{t'}] = \emptyset$. We show that (1) no $p_i$ decides before round $t$ and (2) there is a value $v$, which is the input value of some process $p_j$, such that every undecided $p_i$ decides $v$ in round $t$.

(1) Suppose some process $p_i$ decides in some round $t' < t$. This means $p_i$ passes the guard of Line 19 in round $t'$. After executing Line 18, due to Corollary 5.5.4, at any process $p_i$, $\rho_i^{t'}$ is set to some prefix $\sigma'|_{t'}$ with $\sigma' \in \mathsf{MA}$ and $\sigma'|_{t'} \sim_{p_i} \sigma|_{t'}$. By definition, hence $\sigma'|_{t'} \in [\sigma|_{t'}]$ and, in fact, $[\sigma'|_{t'}] = [\sigma|_{t'}]$. This, taken together with Corollary 5.5.5, yields that, after performing the computations in the guard of Line 19, at every process $p_i$, we

have $K_i^{t'} = \mathrm{KI}[\sigma|_{t'}]$. If $p_i$ passes this guard it means that $K_i^{t'} \neq \emptyset$ which contradicts the assumption that $t$ is the earliest round for which the condition holds.

(2) By assumption, we have some set $K$ such that $\mathrm{KI}[\sigma|_t] = K \neq \emptyset$. Let $m$ be the maximal identifier of any process in $K$. We show the claim for $v = x_m$. Since we assumed that $p_i$ has not decided yet, a similar argument as above shows that every process $p_i$ passes the guard of Line 19 in round $t$ and decides on $\mathsf{x}^t[m]$ via Line 21. Since $p_m \in \mathrm{Ker}(\sigma|_t)$, $(p_m, 0) \in \mathsf{view}^t_{\sigma|_t}(p_i)$ and hence Lines 1 and 9 ensure that $\mathsf{x}^t[m] = x_m = v$. □

Chapter 6

# Vertex-Stable Root Components

In this chapter, we explore the solvability/impossibility border of consensus under message adversaries that support *eventual stabilization* [BRS12, BRS+15, SWS16]: In contrast to the message adversaries of Chapter 5, the set of admissible choices for $G_r$ may change with evolving round numbers $r$ here and these message adversaries are usually not limit-closed. We focus on a special type of eventual stabilization that is based on eventually occurring vertex-stable root components (see Definition 3.2.2). Even though this is only a special case, it is general enough to encompass many intuitive stronger alternatives such as eventual strong connectivity, eventual repetition of the same graph, eventual repetition of the same root component, or the eventual occurrence of a completely connected component.

Apart from being theoretically interesting, considering eventually stabilizing dynamic networks is also useful from a practical perspective: Algorithms that work correctly under eventually stabilizing message adversaries are particularly suitable for systems that suffer from uncoordinated boot-up sequences or systems that must recover from massive transient faults: Network connectivity can be expected to improve over time here, e.g., due to improving clock synchronization quality. Since it is usually difficult to determine the time when such a system has reached normal operation mode, algorithms that terminate only after a reasonably stable period has been reached are obviously advantageous. Algorithms that work correctly under short-lived stable periods are particularly interesting here, since they have higher coverage and terminate earlier in systems where longer stable periods occur only rarely or even not at all. Note that the occurrence of short-lived stability periods were confirmed experimentally in the case of a prototype wireless sensor network [PS16].

Last but not least, stabilizing algorithms require less reliable and, in our case, not inherently bidirectional communication underneath, hence work with cheaper and/or more energy-efficient network communication interfaces.

**Chapter Organization** In this chapter, we thoroughly answer the question of the minimal stability required for solving consensus under eventually stabilizing root components. After the introduction of the relevant message adversaries in Section 6.2, we establish the following results:

(1) We provide a novel algorithm in Section 6.4, along with its correctness proof, which solves consensus for a message adversary that generates graph sequences consisting of graphs that (i) are rooted (c.f. Definition 3.2.1) and (ii) contain a subsequence of $x = D + 1$ consecutive graphs whose root component is vertex-stable (Definition 3.2.2). Herein, the system parameter $D \leqslant n - 1$ is the dynamic network depth, the number of rounds required for all nodes in a vertex-stable root component to reach all nodes in the system. Thanks to (i), our algorithm is always safe in the sense that agreement is never violated; (ii) is only needed to ensure termination. Compared to all existing algorithms for this type of message adversaries like [BRS12, BRS$^+$15, SWS16], where the processes more or less wait for the stability window to occur, our algorithm employs new algorithmic techniques once a candidate stability window has been found: We show that, by "waiting for contradictory evidence", the repeated information propagation from one process to the entire system is enough for everyone to reliably determine if the candidate window was just spurious and should be discarded, or if it has to be "taken seriously", because someone is convinced that it was the guaranteed stability window.

(2) In previous work [BRS12, BRS$^+$18], it has been shown that $x = D - 1$ is a lower bound for the stability interval for all consensus algorithms working under message adversaries that guarantee a stable root component to occur eventually, and that (a bound on) $D$ must be known *a priori*.[1] In Section 6.3, we improve the lower bound to $x = D$, which reveals that the previous bound was not tight and that our algorithm is optimal. This result also shows that the mere propagation of an input value to every process during the stability window does not suffice to solve consensus in this setting.

## 6.1 Related Work

[BRS12] assumed communication graphs with a non-empty set of sources and long-living periods of stability $x = 4D + 1$. [BRS$^+$15] studies consensus under a message adversary with comparably long-lived stability, which gracefully degrades to general $k$-set agreement in case of unfavorable conditions. However, this message adversary must also guarantee a certain influence relation between subsequently existing partitions. [SWS16] established a characterization of uniform consensus solvability/impossibility for longer stability periods. In particular, it provides a consensus algorithm that works for stability periods of at least $2D+1$ but does not require graph sequences where all graphs are rooted. We will present

[1] Whereas this may seem a somewhat unrealistic (though inevitable) restriction at first sight, it must be noted that $D$ only needs to be guaranteed throughout the stability interval.

some of these ideas in more detail in Chapter 7. Note that the experimental evaluation of a wireless sensor network described in [PS16] reveals that this assumption holds true, for a properly chosen value of $D$ (in particular, $D = 4$), with a coverage close to 100% both in an indoor and outdoor environment. Whereas one cannot obviously generalize from a single, non-systematic experimental evaluation, these findings nevertheless suggest that the basic assumption of an eventually vertex-stable root component is not unreasonable in practice.

[RS13] used message adversaries that allow a notion of "eventually forever" to establish a relation to failure detectors. Although we mostly do not consider this "extremal" case in this thesis, which is instead more focused on short-lived stability, we present some insights that can be drawn from this relation in Chapter 9.

## 6.2 Message Adversaries for Stable Root Components

First, we introduce the adversary that adheres to *dynamic network depth* $D$, which gives a bound on the duration of the information propagation from a vertex-stable root component to the entire network. It was shown in [BRS+18, Cor. 1] that always $D \leqslant n-1$; *a priori* restricting $D < n-1$ also allows modelling dynamic networks where information propagation is guaranteed to be faster than in the worst case (as in expander graphs, see [BRS+18]).

**Definition 6.2.1.** $\mathsf{DEPTH}_n(D)$ *is the set of all infinite communication patterns* $\sigma$ *s.t.* $|\Pi_\sigma| = n$ *and, for all finite rounds* $r_1$*, for all subsequences* $\sigma' = (G_{r_1}, \dots, G_{r_1+D-1})$ *of* $\sigma$*, if* $\sigma'$ *is* $R$*-rooted, then* $R \subseteq \mathrm{CP}_p^{r_1+D-1}(r_1 - 1)$ *for all* $p \in \Pi_\sigma$*.*

In other words, the dynamic radius (Definition 4.0.2) of all $\sigma'$ from the above definition is at most $D$.

The following liveness property, *eventual stability*, ensures that eventually every admissible graph sequence $\sigma$ has an $R$-rooted subsequence $\sigma' \subseteq \sigma$ of length $x$. Definition 3.2.2 implies that all sequences have a vertex-stable root component that consists of the same set of processes with possibly varying interconnection topology for $x$ consecutive rounds.

**Definition 6.2.2.** $\Diamond\mathsf{GOOD}_n(x)$ *is the set of all infinite communication patterns* $\sigma$ *such that* $|\Pi_\sigma| = n$ *and there exists a set* $R \subseteq \Pi_\sigma$ *and an* $R$*-rooted* $\sigma' \subseteq \sigma$ *with* $|\sigma'| \geqslant x$*.*

For finite $x$, $\Diamond\mathsf{GOOD}_n(x)$ alone is insufficient for solving consensus: Arbitrarily long sequences of graphs that are not rooted before the stability phase occurs can fool all consensus algorithms to make wrong decisions. For this reason, we introduce a safety property in the form of the message adversary that generates only rooted graphs. As mentioned after Definition 3.2.1, this implies that every communication graph is weakly connected and there is a single root component, i.e. a non-empty set of nodes from which all nodes are reachable.

**Definition 6.2.3.** $\mathsf{ROOTED}_n$ *is the set of all infinite sequences* $\sigma$ *of* rooted *communication graphs such that* $|\Pi_\sigma| = n$.

The short-lived eventually stabilizing message adversary $\Diamond\mathsf{STABLE}_{n,D}(D+1)$ used throughout the main part of this chapter adheres to the dynamic network depth $D$, guarantees that every $G_r$ is rooted and that every sequence has a subsequence of at least $x = D+1$ consecutive communication graphs with a vertex-stable root component. Since processes are aware under which message adversary they are executing, they have common *a priori* knowledge of the dynamic network depth $D$ and the duration of the stability phase $x$. Moreover, depending on the variant actually used, they have some knowledge regarding the system size $n$.

**Definition 6.2.4.** *We call* $\Diamond\mathsf{STABLE}_{n,D}(x) = \mathsf{ROOTED}_n \cap \Diamond\mathsf{GOOD}_n(x) \cap \mathsf{DEPTH}_n(D)$ *the eventually stabilizing message adversary with stability period* $x$*. For a fixed* $D$*, we consider the following generalizations:*

- $\Diamond\mathsf{STABLE}_{<\infty,D}(x) = \bigcup_{n\in\mathbb{N}\setminus\{0,1\}} \Diamond\mathsf{STABLE}_{n,D}(x)$
- $\Diamond\mathsf{STABLE}_{\leqslant N,D}(x) = \bigcup_{n=2}^{N} \Diamond\mathsf{STABLE}_{n,D}(x)$

We observe that $\Diamond\mathsf{GOOD}_n(x) \supseteq \Diamond\mathsf{GOOD}_n(D)$ for all $1 \leqslant x \leqslant D$, hence it follows that $\Diamond\mathsf{STABLE}_{n,D}(x) \supseteq \Diamond\mathsf{STABLE}_{n,D}(D)$.

## 6.3 Impossibility Results and Lower Bounds

Even though processes know the dynamic network depth $D$, for very short stability periods, this is not enough for solving consensus. In Theorem 6.3.1, we prove that consensus is impossible under $\Diamond\mathsf{STABLE}_{<\infty,D}(2D-1)$ (recall that even if the dynamic graph has a finite set of processes $\Pi$, this set is not necessarily known to the processes). That is, if processes do not have access to an upper bound $N$ on the number of processes, solving consensus is impossible if the period $x$ of eventual stability is shorter than $2D$: Here, processes can never be quite sure whether a vertex-stable root component occurred for at least $D$ rounds, which is critical, however, since only a duration of $D$ or more rounds guarantees information propagation, according to Definition 6.2.1.

The core argument of the proof is that an arbitrary correct consensus algorithm $\mathcal{A}$ will fail when exposed to the communication patterns $\sigma, \sigma'$ from Figure 6.1. Fix the input values of processes $p_1, \ldots, p_{D+2}$ to 0 and let all other processes start with input 1. Because $\mathcal{A}$ satisfies termination, process $p_{D+1}$ eventually, by a time $\tau$, has reached a decision in an execution based on $\sigma$. Since the situation is indistinguishable for $p_{D+1}$ from the situation where everyone started with 0, it has to decide 0 by validity. Crucially, $p_{D+1}$ cannot distinguish whether the actual communication pattern is $\sigma$ or $\sigma'$, thus it decides 0 also in the latter. If $n'$ was chosen sufficiently large, however, process $p_{n'}$ never learns of an

input value other than 1. A similar argument as above shows that, by validity, $p_{n'}$ hence eventually decides 1 and thus two values were decided under the communication pattern $\sigma'$. Clearly, $\mathcal{A}$ does not satisfy agreement, a contradiction to the initial supposition that $\mathcal{A}$ is correct, as $\sigma'$ is an admissible communication pattern.

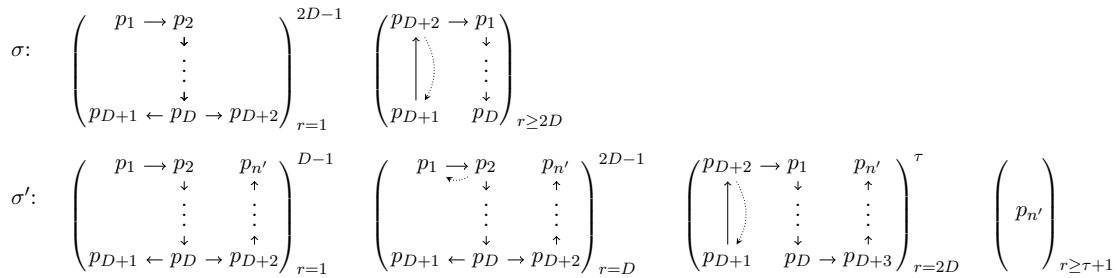


Figure 6.1: Communication patterns of Theorem 6.3.1, where $(G)_{r=a}^{b}$ denotes that $G$ is the communication graph from round $a$ until round $b$. A dotted edge represents an edge which is in $G_i$ if and only if it is not in $G_{i-1}$, starting with it being present in $G_D$. We assume that there are self-loops and there is an edge from every process depicted in the graph to every process not depicted in the graph.

**Theorem 6.3.1** (Appears as Theorem 88 in [Sch18])**.** *Under* $\Diamond\mathsf{STABLE}_{<\infty,D}(x)$ *consensus is impossible for* $0 < x < 2D$.

*Proof.* As we have $\Diamond\mathsf{GOOD}_n(x) \subset \Diamond\mathsf{GOOD}_n(x')$ for $x > x'$, it suffices to show that consensus is impossible under message adversary $\mathsf{MA} = \Diamond\mathsf{STABLE}_{<\infty,D}(2D-1)$.

Pick an arbitrary $D \in \mathbb{N}$ and suppose an algorithm $\mathcal{A}$ solves consensus under $\mathsf{MA}$. Let $n$, resp. $n'$, denote the number of nodes in the communication graphs of $\sigma$, resp. $\sigma'$ from Figure 6.1. We provide two admissible executions $\varepsilon, \varepsilon'$ based on $\sigma$, resp. $\sigma'$ and prove that, with $\varepsilon_r, \varepsilon'_r$ denoting their first $r$ rounds, for $r \leqslant \tau$ we have $\varepsilon_r \sim_{p_{D+1}} \varepsilon'_r$. We show that $p_{D+1}$ decides 0 in $\varepsilon_\tau$ and hence in $\varepsilon'_\tau$, whereas process $p_n$ decides 1 in $\varepsilon'$.

Let $C^0$ be the initial configuration with input values $x_p = 0$ if $p \in \{p_1, \ldots, p_{D+2}\}$ and $x_p = 1$ otherwise, and let $\overline{C}^0$ be the initial configuration where for all input values we have $x_p = 0$.

Consider execution $\varepsilon = \langle C^0, \sigma \rangle$ with $\sigma$ from Figure 6.1, where a dotted edge exists only in every second graph of the sequence, and all processes not depicted have an in-edge from every depicted process. We have $\sigma \in \mathsf{MA}$, since it guarantees eventual stability for $2D-1$ rounds, adheres to the dynamic network depth $D$ and in every round the communication graph is rooted. By the assumed correctness of $\mathcal{A}$, there is a finite time $\hat{\tau}$ by which every process has decided in $\varepsilon$; let $\tau = \max\{\hat{\tau}, 2D\}$. For $p \in \{p_{D+1}, p_{D+2}\}$ (and by agreement thus for all processes), the decision must be 0 because we have $q \in \mathrm{CP}_p^\tau(0) \Rightarrow x_q = 0$, hence $\varepsilon \sim_p \langle \overline{C}^0, \sigma \rangle$, and 0 is decided in $\langle \overline{C}^0, \sigma \rangle$ by validity.

Now, consider the execution $\varepsilon' = \langle C^0, \sigma' \rangle$ with $\sigma'$ from Figure 6.1 and $n' > \tau + D + 3$. Again, $\sigma' \in \mathsf{MA}$, since $(G_r)_{r=\tau+1}^{\infty}$ is $p_{n'}$-rooted. With $\varepsilon_r$, $\varepsilon'_r$ denoting the first $r$ rounds of $\varepsilon$, resp. $\varepsilon'$, for $r \leqslant \tau$ and $p \in \{p_{D+1}, p_{D+2}\}$, we have $\varepsilon \sim_p \varepsilon'$: This is immediately obvious for $1 \leqslant r \leqslant D-1$. For $D \leqslant r \leqslant \tau$, we have $\varepsilon_D \not\sim_q \varepsilon'_D \Leftrightarrow q = p_1$ and, as a simple induction shows, $\varepsilon_r \sim_q \varepsilon'_r \Leftrightarrow q \neq p_1 \wedge p_1 \notin \mathrm{CP}_q^r(D)$. It is not hard to see that $p_1 \notin \mathrm{CP}_p^r(D)$, hence $\varepsilon_r \sim_p \varepsilon'_r$ is maintained.

Consequently, by time $\tau$, $p_{D+1}$ has decided 0 also in $\varepsilon'$. Yet, by construction of $\varepsilon'$, for an arbitrary time $r$, we have $q \in \mathrm{CP}_{p_{n'}}^r(0) \Rightarrow x_q = 1$ since $n' > \tau + D + 3$. By termination, validity, and an analogous argument as above, $p_{n'}$ must hence decide 1 in $\varepsilon'$ eventually, which violates agreement and provides the required contradiction. □

Theorem 6.3.1 shows that consensus is impossible under $\Diamond\mathsf{STABLE}_{<\infty,D}(D+1)$, since no process has a bound on the system size. In the remaining chapter, we thus study the adversary $\Diamond\mathsf{STABLE}_{\leqslant N,D}(D+1)$, for which we show in the next section that consensus can indeed be solved.

We now present two lower bounds for the duration $x$ of the stable period: We prove that even if there is an upper bound $N$ on the number of processes in the current sequence, consensus is impossible under $\Diamond\mathsf{STABLE}_{\leqslant N,D}(x)$ if $x < D$, resp. if $x \leqslant D$ (Theorems 6.3.5 and 6.3.6). Clearly, the latter result improves the former and thus reveals that the impossibility for $x < D$ is not tight. We note, however, that the proof of the impossibility for $x < D$ is more general in that it shows the existence of a perpetually bivalent execution when starting from an *arbitrary* stabilization round $r_0$; Theorem 6.3.6 shows this only for $r_0 = 1$, i.e., when the stable period occurs immediately (which is of course enough for stating an impossibility result).

**Lemma 6.3.2.** *Consider two runs of a consensus algorithm $\mathcal{A}$ under message adversary $\Diamond\mathsf{STABLE}_{\leqslant N,D}(D-1)$, which start from two univalent configurations $C'$ and $C''$ that differ only in the state of one process $p$ at the beginning of round $r$. Then, $C'$ and $C''$ cannot differ in valency.*

*Proof.* Throughout this proof, we assume that the stability phase is fixed to an interval $J$, i.e., as required by $\Diamond\mathsf{GOOD}_n(D-1)$, the $S$-rooted subsequence $\sigma'$ with $|\sigma'| = D-1$ occurs during $J$.

The proof proceeds by contradiction, i.e., suppose that $C'$ and $C''$ have different valency. We will then apply the same sequence of round graphs to extend the execution prefixes that led to $C'$ and $C''$ to get two different runs $e'$ and $e''$. It suffices to show that there is at least one process $q$ that cannot distinguish $e'$ from $e''$: This implies that $q$ will eventually decide on the same value in both executions, which contradicts the assumed different valency of $C'$ and $C''$.

Our choice of the round graphs depends on the following exhaustive cases:

(i) For $p \notin S$, we let the adversary choose any root component consisting of the processes in $S$, for all $G_s$ with $s \geqslant r$. Obviously, every process (i.e., we can choose any) $q \in S$ has the same state throughout $e'$ and $e''$.

(ii) For $p \in S$ and $r \in J$, we choose any root component consisting of the processes in $S$ for all $G_s$ with $s \in J$. For all rounds after $J$, we chose the root component $\{q\}$, where $q$ is the process that is not influenced by a process time pair of $S \times J$ (and hence by $(p, s)$ with $s \in J$). Clearly this process exists by Definition 6.2.1. Hence, $q$ has the same state in $e'$ and $e''$, both during $J$ and afterwards, where it is the single root component.

(iii) For $p \in S$ and $r \notin J$, we choose graphs $G_s$ where the root component is $\{q\}$ and $p$ has only in-edges before $J$; $q$ (satisfying $q \notin S$ and hence $q \neq p$) is again the "distant" process allowed by Definition 6.2.1. From $s \in J$ onwards, we choose the same graphs $G_s$ as in case (ii). Again, $q$ has the same state throughout $e'$ and $e''$, since $(p, r)$ cannot influence any process before $J$ and $(p, s)$ never influences $q$ (in particular, within $J$).

In any case, for process $q$, the sequence of states in the extensions starting from $C'$ and $C''$ is hence the same. Therefore, the two runs are indistinguishable for $q$, which cannot hence decide differently. This provides the required contradiction to the different valencies of $C'$ and $C''$. □

The following technical Lemma 6.3.3 establishes the *connectedness* of the successor configurations, as introduced in [SWK09]. It is based on constructing a sequence of applicable communication graphs that differ only in one edge. Compared to the application in [SWK09][Lemma 1], our construction is complicated by the fact that we must maintain the dynamic network depth of the resulting communication pattern. We use $d_G(v, w)$ to denote the distance (number of edges on a shortest path) from $v$ to $w$ in graph $G$. Subsequently, Lemma 6.3.4 shows that in the case where $n > 2$, we can even find connected successor graphs while avoiding a specific root component $S$.

**Lemma 6.3.3** (Connectedness)**.** *For any two graphs $G'$ and $G''$, we can find a finite sequence of rooted graphs $G', G_1, \dots G_i \dots G''$, where any two consecutive graphs differ only by at most one edge. We say that the configurations $C'$ resp. $C''$ reached by applying $G'$ resp. $G''$ to the same configuration $C$ are* connected *in this case. Moreover, with $\mathrm{Root}(G') = S'$ and $\mathrm{Root}(G'') = S''$, the following can be asserted:*

*(i) If $S' = S'' = S$ for some $S$ then, for all $i$, $\mathrm{Root}(G_i) = S$ and either $G' \subseteq G_i$ or $G'' \subseteq G_i$.*

*(ii) If $S' \neq S''$ and $\mathrm{Root}(G) = S_i \in \{S', S''\}$, for $G \in \{G', G''\}$, then either for all $v \in S_i$, for all $w \in G_i$, $d_{G_i}(v, w) \leqslant d_G(v, w)$ or all but one node of $S_i$ have distance 1 to all other nodes, i.e., $\left|\{v \in S_i \mid \forall w \in G_i : d_{G_i}(v, w) = 1\}\right| \geqslant |S_i| - 1$.*

*Proof.* We describe how to construct the sequence by stating, for each step of our construction, which edge $e$ is modified in $G_i$ in order to arrive at $G_{i+1}$. Let $G' = \langle V, E' \rangle$ and let $G'' = \langle V, E'' \rangle$.

To show (i), we provide a construction that assumes $S' = S'' = S$. In the first phase of the construction, add some edge $e$ from $E'' \setminus E'$. When no such edge remains we have constructed the graph $G_j = G' \cup G''$. We then commence the second phase by removing some $e$ from $E' \setminus E''$ until no such edge remains. For each $G_i$ in the sequence constructed in this way, we have that either $G' \subseteq G_i$ or $G'' \subseteq G_i$, which implies that each graph is rooted since $G'$ and $G''$ both are rooted themselves. To see that all $G_i$ have the same root component, suppose some $G_i$ has a root component different from $S$. Hence an edge $(v, w)$ was added for $v \notin S$, $w \in S$ or all edges $(v, w)$ with $w \in S \setminus \{v\}$ were removed for some $v \in S$. Both contradict the assumption that $S' = S'' = S$, however.

To show (ii), we assume $S' \neq S''$ and, w.l.o.g., that there is some $u \in S' \setminus S''$ (if there is no such node $u$, we can still use the same construction, albeit with reversed direction). First, starting from $G'$, we add some edge $e$, chosen arbitrarily one-by-one, until we arrive at the complete graph. Since $u$ is always in the root component of each $G_i$ generated this way (there is always a spanning tree rooted at $u$ contained in $G_i$), each $G_i$ is rooted and no $G_i$ has root component $S''$. Furthermore, $G' \subseteq G_i$ and hence (ii) holds. It remains to be shown that from the complete graph we can successively delete edges to arrive at $G''$ while respecting (ii). For simplicity, we show that, equivalently, we can add edges to $G''$ and arrive at the complete graph without ever violating (ii). We start by adding to $G''$ the edge $e = (v, w)$ for $v \in S''$, $w \in V$ (including $w = u$) until no such edge remains. Clearly adding those edges cannot increase any distances and does not change the root, hence (ii) is preserved.

We continue by removing $e = (u, v)$ for all $v \in V \setminus \{u\}$. We observe that in the resulting graphs $G_i$ the distance from the root component to any other node is 1, hence (ii) holds. Moreover, no new root component could have been generated this way, since $u$ already has some incoming edges by our construction. We note that in the following final steps only edges are added, hence no additional root component can be generated here as well. Note carefully, though, that in the case of $S' \subseteq S''$ it may happen that $G_i = S'$ is reached before the complete graph is reached in the construction. In this case, we cannot assume $G' \subseteq G_i$ and hence $d_{G_i}(v, w) \leq d_{G'}(v, w)$, but can guarantee the distance 1 condition in the statement of our lemma. Fix some $v \in V \setminus \{u\}$ and add $e = (v, w)$ for any $w \in V$ (if it is not already present, which is the case for $v \in S''$). In the resulting $G_i$, for all nodes of $S_i \setminus \{v\}$, the distance to any other node is equal to 1, ascertaining (ii). Eventually, this yields some $G_i$ where there is an edge $(v, w)$ from every $v \in V \setminus \{u\}$ to each $w \in V$. Finally, we add $e = (u, v)$ for every $v \in V$ to arrive at the complete graph. Again, the distance from any node in $S_i \setminus \{u\}$ to any other node is equal to 1 and hence (ii) holds. □

**Lemma 6.3.4.** *Pick two arbitrary rooted graphs $G'$, $G''$ with* $\mathrm{Root}(G) = S'$ *and* $\mathrm{Root}(G'') = S''$. *If $n > 2$ and given some non-empty $S$ with $S \neq S'$ and $S \neq S''$,*

*there is a sequence of rooted graphs $G', \ldots, G_i, \ldots, G''$ such that any two consecutive graphs of the sequence differ in at most one edge and each $G_i$ of the sequence has a root component different from $S$.*

*Proof.* We show that, for any graph $\overline{G}'$ with root component $\overline{S}'$, there is such a sequence $\overline{G}', \ldots, \overline{G}_i, \ldots, \overline{G}''$ for any graph $\overline{G}''$ with root component $\overline{S}''$ if $\overline{S}''$ differs from $\overline{S}'$ in at most one node, i.e., $|\overline{S}' \setminus \overline{S}'' \cup \overline{S}'' \setminus \overline{S}'| \leqslant 1$. Repeated application of this fact implies the lemma, because, for $n > 2$, it is easy to find a sequence $S', \ldots, S_i, \ldots S''$ of subsets of $\Pi$ s.t. each two consecutive sets of the sequence differ from each other in at most one node and each set of the sequence is $\neq S$.

We sketch how to construct the desired graphs $\overline{G}_i$ of the sequence in three phases.
Phase 1: Remove all edges (one by one) between nodes of $\overline{S}'$ until only a cycle (or, in general, circuit) remains, and then remove all edges between nodes outside of $\overline{S}'$ until only chains going out from $\overline{S}'$ remain. Let $\overline{G}_j$ be the graph resulting from this operation and $\overline{S}_j = \overline{S}'$ be its root component.
Phase 2: If we need to add a node $p$ to $\overline{S}_j$, for some $q \in \overline{S}_j$, first add $(q, p)$. For any $q' \neq q$, $(q', p) \in \overline{G}_j$, where $p \neq q'$, remove $(q', p)$. Finally, add $(p, q)$. If we need to remove a node $p$ from $\overline{S}_j$, for some $(q, p)$, $(p, q') \in \overline{G}_j$, with $q, q' \in \overline{S}_j$, subsequently add $(q, q')$ and $(q', q)$, then remove $(p, q)$ and $(p, q')$.
Phase 3: Since we now already have some graph $\overline{G}_k$ with root component $\overline{S}''$, it is easy to add/remove edges one by one to arrive at the topology of $\overline{G}''$. First, we add edges until the nodes of $\overline{S}''$ are completely connected among each other, the nodes not in $\overline{S}''$ are completely connected among each other, and there is an edge from every node of $\overline{S}''$ to each node not in $\overline{S}''$. Second, we remove the edges not present in $\overline{G}''$. □

The proof of the following impossibility result follows roughly along the lines of the proof of [SWK09, Lemma 3]. It shows, by means of induction on the round number, that a consensus algorithm $\mathcal{A}$ cannot reach a univalent configuration after any finite number of rounds.

**Theorem 6.3.5** (Impossibility of consensus under $\Diamond\mathsf{STABLE}_{\leqslant N,D}(D-1)$)**.** *There is no algorithm that solves consensus under the message adversary $\Diamond\mathsf{STABLE}_{\leqslant N,D}(D-1)$ for $D > 1$ and $n > 2$.*

*Proof.* We follow roughly along the lines of the proof of [SWK09, Lemma 3] and show, per induction on the round number, that no algorithm $\mathcal{A}$ can reach a univalent configuration by round $r$, for any $r > 0$. Since no process can have decided in a bivalent configuration, this violates the termination property of consensus. Throughout this proof, we assume that the stability phase is fixed to an interval $J$, i.e., as required by $\Diamond\mathsf{GOOD}_n(D-1)$, the $R$-rooted subsequence $\sigma'$ with $|\sigma'| = D - 1$ occurs during $J$.

For the base case, we consider binary consensus only and argue similarly to [FLP85] but make use of our stronger validity property: Let $C_x^0$ be the initial configuration, where

the $x$ processes with the smallest ids start with 1 and all others with 0. Clearly, in $C_0^0$ all processes start with 0 and in $C_n^0$ all start with 1, so the two configurations are 0- and 1-valent, respectively. To see that for some $x$ $C_x^0$ must be bivalent, consider that this is not the case, then there must be a $C_x^0$ that is 0-valent while $C_{x+1}^0$ is 1-valent. But, these configurations differ only in $p$, and so by Lemma 6.3.2 they cannot be univalent with different valency.

For the induction step we assume that there is a bivalent configuration $C$ at the beginning of round $r-1$, and show that there is at least one such configuration at the beginning of round $r$. We proceed by contradiction and assume all configurations at the beginning of round $r$ are univalent. Since $C$ is bivalent and all configurations at the beginning of $r$ are univalent, there must be two configurations $C'$ and $C''$ at the beginning of round $r$ which have different valency. Clearly, $C'$ and $C''$ are reached from $C$ by two different round $r-1$ graphs $G' = \langle \Pi, E' \rangle$ and $G'' = \langle \Pi, E'' \rangle$. As we explain in more detail below, we can apply Lemmas 6.3.3 and 6.3.4 to show that there is a sequence of applicable graphs such that $C'$ and $C''$ are connected. Each pair of subsequent graphs in this sequence differs only in one link $(v, w)$, such that the resulting configurations differ only in the state of $w$. Moreover, if the root component in $G'$ and $G''$ is the same, all graphs in the sequence also have the same root component. Since the valency of $C'$ and $C''$ was assumed to be different, there must be two configurations $\overline{C}'$ and $\overline{C}''$ in the corresponding sequence of configurations that have different valency and differ only in the state of one process, say $p$. Applying Lemma 6.3.2 to $\overline{C}'$ and $\overline{C}''$ again produces a contradiction, and so not all successors of $C$ can be univalent.

It remains to be shown that Lemmas 6.3.3 and 6.3.4 indeed yield a sequence of applicable graphs, i.e, that extending the sequence of the $r-1$ graphs so far accordingly yields a prefix of some sequence of $\Diamond\mathsf{STABLE}_{\leqslant N,D}(D-1)$. By the assumptions of the theorem we may assume that $D > 1$ and $n > 2$, which allows us to apply all the claims of Lemma 6.3.4. Since all graphs in the sequence described by Lemmas 6.3.3 and 6.3.4 have exactly one root component, the resulting communication pattern prefix is a prefix of $\mathsf{ROOTED}_n$.

Let $\mathrm{Root}(G_{r-1}) = S_{r-1}$, $\mathrm{Root}(G') = S'$, and $\mathrm{Root}(G'') = S''$. If $S_{r-1} \neq S'$ and $S_{r-1} \neq S''$, Lemma 6.3.4 allows us to construct a sequence s.t. the root component of no $G_i$ of the sequence is $S_{r-1}$. Since vertex-stability has ended, the dynamic network depth $D$ is trivially preserved. If $S' = S''$, since both $G'$ and $G''$ preserved the dynamic network depth $D$, so does every $G_i$ in the sequence described in item (i) of Lemma 6.3.3, because every $G_i$ contains either $G'$ or $G''$. If $S' \neq S''$ and, say $S' = S_{r-1}$, then each $G_i$ with root component $S_i \neq S' = S_{r-1}$ trivially preserves the dynamic network depth $D$ as before. Each $G_i$ with root component $S_i = S'$ either preserves the distances from $S'$ to all other nodes or the distance from all but at most one node $q$ of $S'$ to all other nodes is 1, i.e., $\forall \ell \in S_i \setminus \{q\}, \forall k \in \Pi\colon d_{G_i}(\ell, k) = 1$. In the former case, the dynamic network depth $D$ is preserved because $G'$ maintains the dynamic network depth $D$. In the latter case, $q$ reached at least one process $\ell \in S_i$ in round $r-1$, since it was part of $S_{r-1} = S'$ by assumption. As $\forall \ell \in S_i \setminus \{q\}, \forall k \in \Pi\colon d_{G_i}(\ell, k) = 1$, in addition to $\ell \in \mathrm{CP}_k^r(r-1)$, we have that $q \in \mathrm{CP}_k^r(r-1)$. Thus, the dynamic network depth $D$ is preserved for $D > 1$,

as required.

If $r \in J$, i.e., round $r$ is part of the stability phase, it follows that $G'$ and $G''$ have the same root component and so do all graphs in the sequence provided by item (i) of Lemma 6.3.3. It follows that the resulting communication pattern is in $\Diamond\mathsf{GOOD}_n(D-1)$.

We have hence established that $\Diamond\mathsf{STABLE}_{\leqslant N,D}(D-1)$ is too strong for consensus. □

We now proceed with the proof that consensus is impossible under $\Diamond\mathsf{STABLE}_{\leqslant N,D}(x)$ if $x \leqslant D$. For $N = 2$, it can be derived from [SW89], where it was shown that consensus is impossible if at most $n-1$ messages are lost in each round. In our terminology, this means that consensus is impossible under $\Diamond\mathsf{STABLE}_{\leqslant 2,1}(1)$. For general $N, D$, Theorem 6.3.6 below shows that a stability period of $D$ or less rounds is insufficient for solving consensus for arbitrary values of $N$ as well. This result is not subsumed by [SW89], since the adversary is not restricted by the number of link failures but by the structure of the communication patterns.[2]

Informally, the reason for the impossibility is that there are executions where, even with a stability phase of $D$ rounds, there is a process that cannot precisely determine the root component of the stability window. This can be seen when considering the graphs $G_a, G_b, G_c, G_d$ from Figure 6.2. Fix the input values such that $x_{p_1} \neq x_{p_2}$ and, for each graph $G_a, G_b, G_c, G_d$, consider the four configurations that result when applying the graph repeatedly for $D$ rounds. As these configurations are connected by an indistinguishability relation, and all processes can become the single root component "forever after" (thereby remaining unable to distinguish the four executions), not all these configurations can be univalent; if they were, the configuration resulting from applying $G_a$ for $D$ rounds would have the same valence as the one resulting from applying $G_d$ for $D$ rounds. An inductive argument, similar to the one employed by [SW89], shows that this bivalence can be preserved forever and hence no correct consensus algorithm exists.

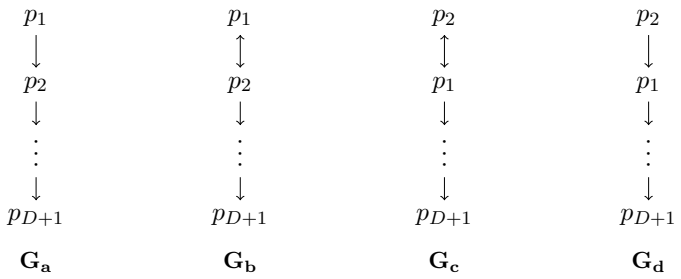


Figure 6.2: Communication graphs for Theorem 6.3.6. We assume there is an edge from every process depicted in the graph to every process not depicted in the graph.

[2] As we will see in the next section, consensus is possible under $\Diamond\mathsf{STABLE}_{n,D}(D+1)$ even though in some cases (e.g. if $D = n-1$ and all communication graphs are chains) up to $(n-1)^2$ messages may be lost in each round!

**Theorem 6.3.6.** *There is no consensus algorithm for* $\Diamond\mathsf{STABLE}_{\leqslant N,D}(x)$ *with* $1 \leqslant x \leqslant D$*, even if the adversary guarantees that the first* $D$ *rounds are* $R$*-rooted.*

*Proof.* In the case where $D = 1$, we need to show the impossibility of $\Diamond\mathsf{STABLE}_{\leqslant N,D}(1)$. We immediately note that $\sigma \in \Diamond\mathsf{STABLE}_{\leqslant N,D}(1)$ if and only if each $G \in \sigma$ is rooted and has a radius of 1 (c.f. Definition 4.0.1). Clearly, the graph sequence where (i) in every graph $G$, two fixed processes $p, q$ have non-self-loop in-edges at most from each other and $(p, q) \in G \vee (q, p) \in G$, and (ii) all other processes have an in-edge from both $p$ and $q$ is in $\Diamond\mathsf{STABLE}_{\leqslant N,D}(1)$. This, however, can be reduced to solving consensus among the two processes $p$, $q$ with up to one link failure between them, which was shown to be impossible in [SW89].

For the remainder of the proof, let us thus assume $D \geqslant 2$. Since $\Diamond\mathsf{STABLE}_{\leqslant N,D}(x) \supset \Diamond\mathsf{STABLE}_{\leqslant N,D}(D)$ for $x \leqslant D$, it suffices to show the impossibility of consensus under $\Diamond\mathsf{STABLE}_{\leqslant N,D}(D)$: If the execution $\varepsilon$ where consensus cannot be solved is admissible under message adversary $\Diamond\mathsf{STABLE}_{\leqslant N,D}(D)$, it is still admissible under $\Diamond\mathsf{STABLE}_{\leqslant N,D}(x)$. The proof proceeds roughly along the lines of [SWK09, Lemma 3]. It first shows that for all consensus algorithms there is a bivalent configuration at time $D$ and proceeds to show by induction that every bivalent configuration has a bivalent successor configuration. Hence, all consensus algorithms permit a perpetually bivalent execution under $\Diamond\mathsf{STABLE}_{\leqslant N,D}(D)$, where consensus cannot be solved.

We show that a bivalent execution is contained in $\Diamond\mathsf{STABLE}'_{\leqslant N,D}(D) \subseteq \Diamond\mathsf{STABLE}_{\leqslant N,D}(D)$, which consists of those executions of $\Diamond\mathsf{STABLE}_{\leqslant N,D}(D)$ where already the first $D$ rounds are $R$-rooted.

For the induction base, we show that not all configurations of $\mathcal{A}$ at time $D$ can be univalent: Assume that an algorithm $\mathcal{A}$ solves consensus under $\Diamond\mathsf{STABLE}'_{\leqslant N,D}(D)$ and suppose that all configurations of $\mathcal{A}$ at time $D$ were univalent.

Let $C^0$ be an initial configuration of $\mathcal{A}$ with $x_{p_1} = 0$ and $x_{p_2} = 1$ and recall the graphs $G_a, G_b, G_c$ and $G_d$ from Figure 6.2. For $i \in \{a, b, c, d\}$ let $C_i^D = \langle C^0, (G_i)_{r=1}^D \rangle$ denote the configuration which results from applying $G_i$ $D$ times to $C^0$. Let $\mathcal{S}(p)$ denote the star-like graph where there is an edge from the center vertex $p$ to every other vertex and from every vertex to itself but there are no other edges in the graph. Clearly, $C_a^D$ is 0-valent since $\langle C_a^D, (\mathcal{S}(p_1))_{D+1}^\infty \rangle \in \Diamond\mathsf{STABLE}'_{\leqslant N,D}(D)$ and for $p_1$ this is indistinguishable from the situation where all processes $p$ have $x_p = 0$. A similar argument shows that $C_d^D$ is 1-valent.

Consider two cases:

(1) $C_b^D$ is 1-valent. But then, $C_a^D$ cannot be 0-valent since $\langle C_a^D, (\mathcal{S}(p_{D+1}))_{D+1}^\infty \rangle \sim_{p_{D+1}} \langle C_b^D, (\mathcal{S}(p_{D+1}))_{D+1}^\infty \rangle$.

(2) $C_b^D$ is 0-valent. Then, since $\langle C_b^D, (\mathcal{S}(p_1))_{D+1}^\infty \rangle \sim_{p_1} \langle C_c^D, (\mathcal{S}(p_1))_{D+1}^\infty \rangle$, $C_c^D$ is also 0-valent. But then $C_d^D$ cannot be 1-valent because the following indistinguishability holds: $\langle C_c^D, (\mathcal{S}(p_{D+1}))_{D+1}^\infty \rangle \sim_{p_{D+1}} \langle C_d^D, (\mathcal{S}(p_{D+1}))_{D+1}^\infty \rangle$.

Hence, not all configurations at time $D$ are univalent.

For the induction step, let us assume that there exists a bivalent configuration $C^r$ for a time $r \geqslant D$. For a contradiction, assume that all configurations at time $(r+1)$ reachable from $C^r$ are univalent. Thus, there exists a 0-valent configuration $C_0^{r+1} = \langle C^r, G_0 \rangle$ that results from applying a communication graph $G_0$ to $C^r$. Moreover, there is a 1-valent configuration $C_1^{r+1} = \langle C^r, G_1 \rangle$ that results from applying a communication graph $G_1$ to $C^r$.

First, let us show that for $G \in \{G_0, G_1\}$, it holds that, if $\mathrm{Root}(G) = \mathrm{Root}(G_r)$, there is an applicable graph $G'$ s.t. $\langle C^r, G' \rangle$ has the same valency as $\langle C^r, G \rangle$ and $\mathrm{Root}(G) \neq \mathrm{Root}(G')$. The reason for this is that we can construct $G'$ from $G$ by simply adding an edge $(p \to q)$ for a $q \neq p$, $p \notin \mathrm{Root}(G)$, $q \in \mathrm{Root}(G)$ if $|\mathrm{Root}(G)| = 1$, respectively, by removing $(p \to q)$ for a $p \in \mathrm{Root}(G)$ and all $p \neq q \in \mathrm{Root}(G)$ if $|\mathrm{Root}(G)| > 1$. This yields a graph $G'$ with the desired property, as $\langle C^r, G, (\mathcal{S}(p))_{r+1}^{\infty} \rangle \sim_p \langle C^r, G', (\mathcal{S}(p))_{r+1}^{\infty} \rangle$. The applicability of $G'$ follows because $G'$ is rooted and $\mathrm{Root}(G') \neq \mathrm{Root}(G_r)$ ensures that the resulting subsequence is a prefix of a sequence of $\mathsf{DEPTH}_n(D)$ for all $D > 1$, because, for these choices of $D$, a changing root component trivially satisfies Definition 6.2.1.

Hence there are graphs $G_0', G_1'$ such that $\mathrm{Root}(G_0') \neq \mathrm{Root}(G_r)$, $\mathrm{Root}(G_1') \neq \mathrm{Root}(G_r)$, and $\langle C^r, G_0' \rangle$ is 0-valent while $\langle C^r, G_1' \rangle$ is 1-valent. As we assumed $D \geqslant 2$ it follows that $n > 2$. We can hence apply Lemma 6.3.4 to go from $G_0'$ to $G_1'$ by adding/removing a single edge at a time, without ever arriving at a graph that has more than one root component or has the same root component as $G_r$. Somewhere during adding/removing a single edge, we transition from a graph $G_i$ to a graph $G_{i+1}$, by modifying an edge $(p \to q)$, where the valency of $C = \langle C^r, G_i \rangle$ differs from the valency of $C' = \langle C^r, G_{i+1} \rangle$. Nevertheless, $G_i$ and $G_{i+1}$, are applicable to $C^r$ because they are rooted and have a different root component as $G_r$, hence guarantee the membership of the sequence in $\mathsf{DEPTH}_n(D)$ for all $D > 1$. However, $C$ and $C'$ cannot have a different valency because $\langle C, (\mathcal{S}(p))_{r+1}^{\infty} \rangle \sim_p \langle C', (\mathcal{S}(p))_{r+1}^{\infty} \rangle$. This is a contradiction and hence not all configurations at time $(r+1)$ can be univalent.

□

## 6.4 Solving Consensus with $D+1$ Rounds of Stability

We now present our consensus algorithm for the message adversary $\Diamond\mathsf{STABLE}_{\leqslant N,D}(D+1)$, where a bound $N \geqslant n$ is known *a priori*. The pseudo-code for the main algorithm is presented in Algorithm 6.2. It relies on a collection of functions given in Algorithm 6.1.

Since the detailed correctness proof is somewhat tedious, we first give an informal description of the algorithm where we provide references to the lemmas that correspond to our informal description.

**Overview.** In essence, each process $p$ that executes Algorithm 6.2 tries to solve consensus by approximating the current graph sequence and keeping track of the relevant partial

states of the other processes. Most prominently, this includes their proposal value $\mathsf{x}$, basically their current decision value estimate. If a process observes that in the current graph sequence, the stability phase could have occurred, i.e., it finds a root component $R$ that might have been stable for $D+1$ rounds, it locks on to the maximum over the proposal values of the members of $R$. Subsequently, $p$ waits if there is evidence refuting its observation. As soon as $p$ finds such contradictory evidence, it clears its locked-on state. If, on the other hand, $p$ does not find such evidence for a sufficiently long time, it decides on its proposal value. In order for the latter to be a safe decision, the algorithm has a mechanism that guarantees the following: As soon as a process $q$ detects that there might have been a process $p$ that is convinced it holds the correct proposal value, $q$ adopts the proposal value of $p$. Crucially, $q$ does not enter a locked-on state in this case. The main difficulties of this approach, apart from implementing the usual graph approximation and book-keeping mechanisms, are to ensure that (1) it cannot happen that distinct processes are simultaneously convinced of two different proposal values for too long, that (2) the significant information propagation delays, inherent in this setting, still guarantee a timely adaptation of the proposal of a convinced process in the entire system, while maintaining that (3) the stability period will eventually lead to the same decision at every process.

**Detailed description.** The essential data structures that each process $p$ maintains are the set of the (reduced) *past process states* $\mathsf{S}$, the *communication graph approximation* $\mathsf{A}$ of the communication graphs that occurred so far, and the *set of participating processes* $\mathsf{P}$ of whose existence $p$ has learned by now (Lemma 6.4.2; recall that, given a $\sigma \in \Diamond\mathsf{STABLE}_{\leqslant N,D}(D+1)$, $p$ does not have exact knowledge of $|\Pi_\sigma|$ and hence no *a priori* knowledge of $\Pi_\sigma$). Process $p$ also holds two crucial local variables, the *lock round* $\ell$ and the *proposal value* $\mathsf{x}$, whose significance we outline below. In order to access $\mathsf{S}$ in a clean way, Algorithm 6.1 provides the functions $\mathtt{L}(q,s)$ and $\mathtt{X}(q,s)$ that, when executed by $p$, return the value of $\ell$ resp. $\mathsf{x}$ of remote process $q$ at time $s$, or $-1$ if this value is unknown to $p$ (Lemma 6.4.3). At the start of each round $r$, process $p$ attempts to broadcast $\mathsf{P}$, $\mathsf{S}$, and $\mathsf{A}$. By our system model, every process $q$ with $(p,q) \in G_r$, including $q=p$, will receive this message. Maintenance of $\mathsf{S}$ and $\mathsf{A}$ is delegated primarily to the `update` function, which merges the content of all messages received from a process $q$ into the local copies of $\mathsf{A}$ and $\mathsf{S}$ and adds an edge $(q,p)$, labeled with the appropriate round, to $\mathsf{A}$. In addition to this, $p$ appends its own value of $\ell$ and $\mathsf{x}$ to $\mathsf{S}$ at the end of round $r$.

Before we continue with the logic of the main algorithm, we note a key insight about our system model: To some extent, by recording received messages, interpreting them as in-edges in a communication graph, and trying to flood the system with these records, processes can reliably detect root components. Algorithm 6.1 implements this in the function $\mathtt{searchRoot}(s)$, which, when called by $p$ in round $r$, returns a set $R$ that satisfies two key properties: If $R \neq \emptyset$ then $R = \mathrm{Root}(G_s)$ (Lemma 6.4.5) and if $R = \mathrm{Root}(G_s)$ and there is a chain of messages such that $R \subseteq \mathrm{CP}_p^r(s)$ then $\mathtt{searchRoot}(s) = R$ (Lemma 6.4.4).

The very basic operation of Algorithm 6.2 is to find root components and "lock on" to

them, by setting the lock round $\ell$ to the current round and $\mathsf{x}$ to a deterministic function (e.g. max()) of the proposal values of the root members. In some sense, the process hopes to hit $\rho$ of the $\rho$-rooted sequence of length $D+1$ that is promised by the specification of $\Diamond\mathsf{STABLE}_{\leqslant N,D}(D+1)$. After this, a process simply waits for contradictory evidence that suggests that the currently locked-on root component could not have been $\rho$. In more detail, in each round $r$, Algorithm 6.2 proceeds in two phases:

In the first phase, $p$ checks whether it needs to adapt its values of $\mathsf{x}$ or $\ell$. It does this in three cases:

1. When $p$ detects that a root component $R$ occurred $D$ rounds ago and $p$ either had $\ell = 0$ or, $D+1$ rounds ago, had detected a different root component $R' \neq R$. In either case, $p$ sets its lock round $\ell \leftarrow r$ and its proposal $\mathsf{x}$ gets the maximum proposal $\mathsf{x}_q$ value over all processes $q \in R$ from $D$ rounds ago in Line 9 (Lemma 6.4.10). This works because if the root component $R'$ that was found was indeed $\rho$, every process must have locked on to it and thus, when a process detects a changed root component $R$, it is guaranteed that all members of $R$ are already locked-on to $\rho$. In this way the proposal value of $\rho$ is preserved forever (Lemma 6.4.15).

2. If the detection from the previous case failed, $p$ sets $\ell \leftarrow 0$ if there is evidence that contradicts the current proposal value $\mathsf{x}$ and lock round $\ell$. That is, if a process $q$ had $\ell_q = 0$ or a proposal different from $\mathsf{x}$ within the last $N$ rounds but after the current lock round. Algorithm 6.1 implements this in the function $\texttt{latestRefutation}(r-N, r-1)$ (Lemma 6.4.8), as called in Line 13. The main aspect of this procedure is that $p$ cannot possibly remove a lock on $\rho$, as $\rho$ would lead all processes to lock on to it and remain locked on to it forever, hence there is no contradictory evidence (Lemma 6.4.15).

3. Possibly in addition to case 2 above, process $p$ adapts its own proposal to $v$ if $p$ sees that every process that was locked on to something during the last $N$ rounds was locked on to $v$. This is to ensure that processes adapt their proposal if there is a set of locked-on processes that never learned of contradictory evidence and might be tempted to assume that their lock value stems from $\rho$ itself. In this case, the function $\texttt{uniqueCandidate}(r-N, r-1)$ returns the value $v$ when called in Line 16 (Lemma 6.4.6 and Lemma 6.4.7).

In the second phase, process $p$ waits until it is safe to decide on $\mathsf{x}$. This is the case when, according $p$'s point of view, in the last $N(D+2N)$ rounds all processes are locked on and have the same proposal value. Process $p$ performs this check via the call to $\texttt{allGood}(r-N(D+2N), r-1)$ in Line 17 (Lemma 6.4.9). The crucial point here is that, by a pigeon-hole argument, the observation of $N(D+2N)$ rounds where all processes are locked on and have the same proposal value implies that there was in fact a "$v$-locked sequence" of $D+2N$ rounds (Lemma 6.4.13). A $v$-locked sequence (Definition 6.4.11) consists only of communication graphs where every process in the root component is locked on and has the same proposal value. Such a sequence guarantees that all future proposal values are $v$ (Lemma 6.4.12), thereby ensuring the safety of the algorithm.

**Algorithm 6.1:** Helper functions for process $p$.

```
1  Function update(q, P_q, S_q, A_q):
2      P ← P ∪ {q} ∪ P_q
3      S ← S ∪ S_q
4      A ← A ∪ {(r, q, p)} ∪ A_q
5  Function searchRoot(s):
6      V ← {v ∈ P | ∃(s, *, v) ∈ A or ∃(s, v, *) ∈ A}
7      E ← {(u, v) ∈ P^2 | ∃(s, u, v) ∈ A}
8      Let SCC(V, E) denote the set of vertex sets of the strongly connected components
       (SCCs) of ⟨V, E⟩. A single node q may constitute a SCC only if (q, q) ∈ E.
9      foreach C ∈ SCC(V, E) do
10         if ∄v ∈ V \ C: (v, u) ∈ E for a u ∈ C then
11             return C
12     return ∅
13 Function L(q, s):
14     if ∃(q, s, *, ℓ) ∈ S then return ℓ
15     else return −1
16 Function X(q, s):
17     if ∃(q, s, x, *) ∈ S then return x
18     else return −1
19 Function latestRefutation(a, b):
20     T ← {i ∈ [a, b] | ∃q ∈ P: L(q, i) = 0 or X(q, i) ∉ {−1, x}}
21     if T ≠ ∅ then return max(T)
22     else return −1
23 Function uniqueCandidate(a, b):
24     if ∃k ∈ ℕ: ∀u ∈ P, ∀i ∈ [a, b]: L(u, i) > 0 ⇒ X(u, i) = k and ∃q ∈ P, ∃j ∈ [a, b]:
       L(q, j) > 0 then
25         return k
26     else
27         return −1
28 Function allGood(a, b):
29     return (∀q ∈ P, ∀i ∈ [a, b]: L(q, i) ≠ 0 and X(q, i) ∈ {−1, x})
```

## Correctness proof

We now prove the correctness of Algorithm 6.2 under an arbitrary $\sigma \in \Diamond\mathsf{STABLE}_{\leqslant N,D}(D+1)$. For variable v, in our analysis, we use $\mathsf{v}_p^r$ to denote the value of v at process $p$ at time $r$ (or, equivalently, the value that v had after it was written to by $p$ for the last time in round $r$). Similarly, we use $\texttt{func}_p^r(\mathsf{v})$ to describe the return value of function `func` when called by $p$ with parameter v during its round-$r$ computation. Note that these return values are the same as the return values after the end of round $r$: Algorithm 6.2 calls functions that provide a return value only on Lines 8–18 and these functions rely exclusively on P, S, and A, which are updated in Line 7 only and not modified in Lines 8–18.

**Algorithm 6.2:** Consensus algorithm, code for process $p$. Uses function definitions from Algorithm 6.1.

**Initialization**:
1 $r \leftarrow 0$, $\mathsf{x} \leftarrow x_p$, $\ell \leftarrow 0$, $\mathsf{A} \leftarrow \emptyset$, $\mathsf{P} \leftarrow \emptyset$
2 $\mathsf{S} \leftarrow \{(p, r, \mathsf{x}, \ell)\}$
3 $r \leftarrow 1$

**Round $r$ communication**:
4 Attempt to send $(\mathsf{P}, \mathsf{S}, \mathsf{A})$ to all
5 Receive $m_q$ from all $q$ with $(q, p) \in G_r$ (incl. $m_p$, for $q = p$)

**Round $r$ computation**:
6 **foreach** $m_q$ *s.t.* $p$ *received* $m_q = (\mathsf{P}_q, \mathsf{S}_q, \mathsf{A}_q)$ *in round* $r$ **do**
7 | `update`$(q, \mathsf{P}_q, \mathsf{S}_q, \mathsf{A}_q)$
8 $\mathsf{R} \leftarrow$ `searchRoot`$(r - D)$
9 **if** $\mathsf{R} \neq \emptyset$ *and* ($\ell = 0$ *or* $\mathsf{R} \neq$ `searchRoot`$(r - D - 1)$) **then**
10 | $\mathsf{x} \leftarrow \max\{\mathtt{X}(q, r - D) \mid q \in \mathsf{R}\}$
11 | $\ell \leftarrow r$
12 **else if** $r > N$ **then**
13 | **if** `latestRefutation`$(r - N, r - 1) \geqslant \ell$ **then**
14 | | $\ell \leftarrow 0$
15 | **if** `uniqueCandidate`$(r - N, r - 1) \neq -1$ **then**
16 | | $\mathsf{x} \leftarrow$ `uniqueCandidate`$(r - N, r - 1)$
17 **if** $r > N(D + 2N)$, $y_p = \perp$, $\ell > 0$, *and* `allGood`$(r - N(D + 2N), r - 1) =$ TRUE **then**
18 | $y_p \leftarrow \mathsf{x}$
19 $\mathsf{S} \leftarrow \mathsf{S} \cup (p, r, \mathsf{x}, \ell)$
20 $r \leftarrow r + 1$

### Algorithm 6.1: Supporting functions

Before commencing with the proof that our algorithm satisfies the consensus specification, we show some essential properties of the supporting functions in Algorithm 6.1. As a preliminary, Lemma 6.4.1 shows that $\mathsf{A}_p^r$ contains an under-approximation of the edges of $G_s$ and that, if $p$ learned that $(u, v) \in G_s$, then $p$ learned all the in-edges of $v$ in $G_s$ as well. The following Lemma 6.4.2 and Lemma 6.4.3 establish the connection between the causal past, the value of $\mathsf{P}$ and the return values of the functions `L`() and `X`().

**Lemma 6.4.1.** *Pick an arbitrary time $s$ and let $r > s$. Then, $(s, u, v) \in \mathsf{A}_p^r$ if and only if $(u, v) \in G_s$ and $v \in \mathrm{CP}_p^r(s)$.*

*Proof.* First, we show that $(s, u, v) \in \mathsf{A}_p^r$ implies $(u, v) \in G_s$. By Line 4 of Algorithm 6.1, at time $s$, $(s, u, v) \in \mathsf{A}_v^s$ if and only if $w = v$. Hence $v$ called Line 7 of Algorithm 6.2 in round $s$ as it received a message from $u$ in that round. But then $(u, v) \in G_s$, as per our system model.

Next, we show by induction on $r \geqslant s+1$ that $(s,u,v) \in \mathsf{A}_p^r$ implies $v \in \mathrm{CP}_p^r(s)$ and $(u,v) \in G_s$.

For $r = s+1$, if $v = p$, the claim is immediate. If $v \neq p$, we have again that $(r-1,u,v) \in \mathsf{A}_w^{r-1}$ if and only if $v = w$. Because we assume $(r-1,u,v) \in \mathsf{A}_p^r$, there must be $(v,p) \in G_r$, which implies $v \in \mathrm{CP}_p^r(r-1)$ by Definition 3.3.1 as well as $(u,v) \in G_s$.

For every time $r > s+1$, if $(s,u,v) \in \mathsf{A}_p^{r-1}$, as we assume self-loops in every communication graph, the claim follows from the induction hypothesis and Definition 3.3.1. So, let $(s,u,v) \notin \mathsf{A}_p^{r-1}$. Since we assume $(s,u,v) \in \mathsf{A}_p^r$, this means that $(s,u,v) \in \mathsf{A}_q^{r-1}$ for a $q \in \Pi_\sigma$ with $(q,p) \in G_r$. Applying the hypothesis to $q$ yields $v \in \mathrm{CP}_q^{r-1}(s)$ and $(u,v) \in G_s$. Taken together, this means that $v \in \mathrm{CP}_p^r(s)$ by Definition 3.3.1 and $(u,v) \in G_s$, as asserted.

Finally, we show by induction on $r \geqslant s+1$ that $(u,v) \in G_s$ and $v \in \mathrm{CP}_p^r(s)$ implies $(s,u,v) \in \mathsf{A}_p^r$.

For $r = s+1$, because $(u,v) \in G_{r-1}$, Line 4 of Algorithm 6.1 adds $(r-1,u,v)$ to $\mathsf{A}_v^{r-1}$. Since $v \in \mathrm{CP}_p^r(r-1)$, $p$ receives $\mathsf{A}_v^{r-1}$ in a round-$r$ message from $v$ and adds it to $\mathsf{A}_p^r$.

For $r > s+1$, the assumption that $v \in \mathrm{CP}_p^r(s)$ allows us to use Definition 3.3.1 to find a $q \in \Pi_\sigma$ such that $v \in \mathrm{CP}_q^{r-1}(s)$ and $(q,p) \in G_r$. As we assume $(u,v) \in G_s$, the hypothesis asserts that $(s,u,v) \in \mathsf{A}_q^{r-1}$. Hence $p$ adds $(s,u,v)$ to $\mathsf{A}_p^r$ in Line 4 of Algorithm 6.1 when it receives $\mathsf{A}_q^{r-1}$ with the round-$r$ message of $q$. □

**Lemma 6.4.2.** *If $q \in \mathrm{CP}_p^r(1)$ then $q \in \mathsf{P}_p^r$.*

*Proof.* By induction on $r \geqslant 1$. The base case, $r = 1$, holds trivially because, by Definition 3.3.1, $\mathrm{CP}_p^1(1) = \emptyset$. For $r > 1$, suppose $q \in \mathrm{CP}_p^r(1)$ and $q \notin \mathsf{P}_p^r$. As $\mathsf{P}_p^r$ never shrinks ($\mathsf{P}$ is only extended in Algorithm 6.1, Line 2), $q \notin \mathsf{P}_p^{r-1}$. By the hypothesis thus $q \notin \mathrm{CP}_p^{r-1}(1)$. But since $q \in \mathrm{CP}_p^r(1)$, by Definition 3.3.1, either $q \in \mathrm{In}_p(G_r)$ or else there is a process $u$ s.t. $q \in \mathrm{CP}_u^{r-1}(1)$ and $u \in \mathrm{In}_p(G_r)$. In the latter case, invoking the hypothesis for $u$ reveals that $q \in \mathsf{P}_u^{r-1}$. In both cases, $q$ is added by Line 2 of Algorithm 6.1 during the round-$r$ computation of process $p$. Hence $q \in \mathsf{P}_p^r$, a contradiction. □

**Lemma 6.4.3.** *$\mathtt{L}_p^r(q,s)$ returns $\ell_q^s$ if $q \in \mathrm{CP}_p^r(s)$ and $-1$ otherwise. Similarly, $\mathtt{X}_p^r(q,s)$ returns $\mathsf{x}_q^s$ if $q \in \mathrm{CP}_p^r(s)$ and $-1$ otherwise.*

*Proof.* We show the claim for $\mathtt{L}_p^r(q,s)$. The proof for $\mathtt{X}_p^r(q,s)$ is analogous. We proceed by induction on $r \geqslant s$.

For $r = s$, we observe that $p$ cannot receive a message containing $(q,s,x_q^s,\ell_q^s)$ from a process $q$ before round $s+1$. Hence $(q,s,x_q^s,\ell_q^s) \notin \mathsf{S}_p^s$ and $\mathtt{L}_p^s(q,s)$ returns $-1$, which shows the claim as $q \notin \mathrm{CP}_p^s(s)$ because $\mathrm{CP}_p^s(s) = \emptyset$ according to Definition 3.3.1.

For $r > s$, we distinguish two cases (i) and (ii):

(i) $q \notin \mathrm{CP}_p^r(s)$. By the induction hypothesis and Definition 3.3.1, every $u \in \mathrm{In}_p(G_r)$ (including $p$ itself) satisfy $u \neq q$ and $\mathtt{L}_u^{r-1}(q,s) = -1$. Due to Line 14 of Algorithm 6.1, $\nexists(q,s,*,*) \in \mathsf{S}_u^{r-1}$. Thus, when calling $\mathtt{update}(u, \mathsf{P}_u, \mathsf{S}_u, \mathsf{A}_u)$ in round $r$, no tuple $(q,s,*,*)$ is added to $\mathsf{A}_p$ and thus $\mathtt{L}_p^r(q,s) = -1$.

(ii) $q \in \mathrm{CP}_p^r(s)$. Suppose $\mathtt{L}_p^r(q,s) \neq \ell_q^s$. Since a tuple $(q,s,*,\ell)$ in $\mathsf{S}_p^r$ could only originate at process $q$ in round $s$ at Line 19 of Algorithm 6.2, we have $\ell = \ell_q^s$. $\mathtt{L}_p^r(q,s) \neq \ell_q^s$ thus implies that $\nexists(q,s,*,*) \in \mathsf{S}_p^r$. As $\mathsf{S}_p^r$ never shrinks, $\nexists(q,s,*,*) \in \mathsf{S}_p^{r-1}$ and hence $\mathtt{L}_p^{r-1}(q,s) = -1$. By the induction hypothesis, $q \notin \mathrm{CP}_p^{r-1}(s)$. Due to the assumption that $q \in \mathrm{CP}_p^r(s)$ and Definition 3.3.1, either $q \in \mathrm{In}_p(G_r)$ or there is a process $u \in \mathrm{In}_p(G_r)$ with $q \in \mathrm{CP}_u^{r-1}(s)$. In the former case, we set $u = q$, in the latter case, applying the induction hypothesis to $u$ reveals $(q,s,*,\ell_q^s) \in \mathsf{S}_u^{r-1}$. In both cases, $p$ calls $\mathtt{update}(u,*,\mathsf{S}_u^{r-1},*)$ in Line 7 in round $r$ and thus adds $\mathsf{S}_u^{r-1}$ to $\mathsf{S}_p^r$ in Line 3 of Algorithm 6.1. Hence $(u,s,*,\ell_q^s) \in \mathsf{S}_p^r$ and, by Line 14 of Algorithm 6.1, $\mathtt{L}_p^r(q,s) = \ell_q^s$, a contradiction. □

The following Lemmas 6.4.4 and 6.4.5 prove the formal properties `searchRoot()` provides for the main consensus algorithm. In our analysis, we use $\mathsf{E}_p^r$, $\mathsf{V}_p^r$ to denote the value of $\mathsf{E}$, $\mathsf{V}$ at process $p$ after $p$ finished Line 7 of Algorithm 6.1 during a call of $\mathtt{searchRoot}_p^r(s)$, which originate solely in tuples $(s,*,*) \in \mathsf{A}_p^r$.

**Lemma 6.4.4.** *Pick $\sigma \in \Diamond\mathsf{STABLE}_{\leqslant N,D}(D+1)$, fix a round $s$ and let $r > s$. If any of the following holds:*

*(1) $R = \mathrm{Root}(G_s)$ and $R \subseteq \mathrm{CP}_p^r(s)$, or*

*(2) $(G_i)_{i=s}^{s+D}$ is $R$-rooted and $r \geqslant s+D$,*

*then $\mathtt{searchRoot}_p^r(s) = R$ for every process $p$.*

*Proof.* Since $\sigma \in \Diamond\mathsf{STABLE}_{\leqslant N,D}(D+1)$ we have $\sigma \in \mathsf{DEPTH}_n(D)$ with $n \leqslant N$ and, by Definition 6.2.1, for all $p \in \Pi_\sigma$ and all $q \in R$ we have $q \in \mathrm{CP}_p^r(s)$ if (1) or (2) holds. Since $R = \mathrm{Root}(G_s)$ in both cases, by Lemma 6.4.1 and Lines 6 and 7 of Algorithm 6.1, $R$ is a root component of $\langle \mathsf{V}_p^r, \mathsf{E}_p^r \rangle$ after $p$ finished Line 7 during a call of $\mathtt{searchRoot}_p^r(s)$.

If $R$ is the unique root component of $\langle \mathsf{V}_p^r, \mathsf{E}_p^r \rangle$, then the loop of Line 6 in Algorithm 6.1 returns $R$ as $R$ is the only strongly connected component of $\langle \mathsf{V}_p^r, \mathsf{E}_p^r \rangle$ that has no incoming edge from a node not in $R$.

If $R$ is not the unique root component of $\langle \mathsf{V}_p^r, \mathsf{E}_p^r \rangle$, then $G_s$ cannot be rooted by Lemma 6.4.1. But then $\nexists n \leqslant N\colon\ \sigma \in \mathsf{ROOTED}_n$, contradicting $\sigma \in \Diamond\mathsf{STABLE}_{\leqslant N,D}(D+1)$. □

**Lemma 6.4.5.** $\forall r > s\ \forall p \in \Pi_\sigma\colon$ *if $R = \mathtt{searchRoot}_p^r(s) \neq \emptyset$, then*

*(1) $R = \mathrm{Root}(G_s)$,*

*(2)* $\forall q \in R\colon\ q \in \mathrm{CP}_p^r(s)$, *and*

*(3) if* $\exists t \leqslant s\colon\ \mathrm{Root}(G_t) = R$ *then* $\texttt{searchRoot}_p^r(t) = R$.

*Proof.* To show the lemma, let us assume that $R = \texttt{searchRoot}_p^r(p,s) \neq \emptyset$, i.e., according to the loop of Line 6 in Algorithm 6.1, for all $u \in R$, $v \in \mathsf{V}_p^r \setminus R$, there is no edge $(v,u) \in \mathsf{E}_p^r$

(1) Since $R$ was returned by the loop in Line 6 of Algorithm 6.1, $R$ is a strongly connected component of $\langle \mathsf{V}_p^r, \mathsf{E}_p^r \rangle$, which implies that for every $v$ of $R$ there is an edge $(*,v) \in \mathsf{E}_p^r$. By Lemma 6.4.1, if $p$ learned one in-edge of a node $v$ in $G_s$, it learned all in-edges of $v$ in $G_s$. Thus, $R$ is also a strongly connected component in $G_s$, and there is no edge in $G_s$ to a node of $R$ from a node not in $R$. Taken together, this matches precisely Definition 3.2.1, hence $R = \mathrm{Root}(G_s)$.

(2) By the construction of $\langle \mathsf{V}_p^r, \mathsf{E}_p^r \rangle$ in Lines 6 and 7 of Algorithm 6.1, $(u,v) \in \mathsf{E}_p^r$ if and only if $(s,u,v) \in \mathsf{A}_p^r$. As we have shown above, $R$ is the root component of $\langle \mathsf{V}_p^r, \mathsf{E}_p^r \rangle$. Therefore, for all $v \in R$ there is an edge $(s,u,v) \in \mathsf{A}_p^r$. By Lemma 6.4.1, $v \in \mathrm{CP}_p^r(s)$.

(3) Because, as we have shown above, for all $v \in R$ we have $v \in \mathrm{CP}_p^r(s)$, by Definition 3.3.1, $v \in \mathrm{CP}_p^r(t)$. Therefore $\texttt{searchRoot}_p^r(t) = R$ by Lemma 6.4.4. □

We continue with lemmas about the remaining helper functions. We show how they use the "getter-functions" $\texttt{L}()$ and $\texttt{X}()$ to check some crucial properties of the local state and approximation sets $\mathsf{S}$, $\mathsf{A}$.

**Lemma 6.4.6.** *Let* $\texttt{uniqueCandidate}_p^r(r-N, r-1) \neq -1$. *Assume either of the following holds:*

*(1)* $\forall s \in [r-N, r-1], \forall q \in \mathrm{CP}_p^r(s)\colon\ \ell_q^s = 0 \vee \mathsf{x}_q^s = v$, *or*

*(2)* $\exists s \in [r-N, r-1], \exists q \in \mathrm{CP}_p^r(s)\colon\ \ell_q^s > 0 \wedge \mathsf{x}_q^s = v$,

*then* $\texttt{uniqueCandidate}_p^r(r-N, r-1) = v$.

*Proof.* Since $\texttt{uniqueCandidate}_p^r(r-N, r-1) \neq -1$, as stated in Line 24 of Algorithm 6.1, $\exists k \in \mathbb{N}\colon\ \forall u \in \mathsf{P}, \forall i \in [r-N, r-1]\colon\ \texttt{L}(u,i) > 0 \Rightarrow \texttt{X}(u,i) = k$ and $\exists q \in \mathsf{P}, \exists j \in [r-N, r-1]\colon\ \texttt{L}(q,j) > 0$. That is, by time $r$, $p$ was influenced by a process $q$ that was locked, i.e., had $\ell_q > 0$, in the last $N$ rounds and in fact all locked processes that managed to influence $p$ in the last $N$ rounds were locked on to the same value $k$.

(1) By Lemma 6.4.2 and 6.4.3, $\forall q \in \mathsf{P}_p^r, \forall s \in [r-N, r-1]\colon\ \texttt{X}(q,s) = v$ or $\texttt{L}(q,s) \in \{-1, 0\}$. Thus, all processes that are locked, are locked on $v$ and hence $k = v$. It follows that $\texttt{uniqueCandidate}_p^r(r-N, r-1) = v$.

(2) By Lemma 6.4.2 and 6.4.3, $\exists s \in [r-N, r-1], \exists q \in \mathsf{P}_p^r$ with $\texttt{L}_p^r(q,s) > 0$ and $\texttt{X}_p^r(q,s) = v$. Hence $k = v$ and $\texttt{uniqueCandidate}_p^r(r-N, r-1) = v$. □

**Lemma 6.4.7.** *If* $\exists s \in [r-N, r-1], \exists q \in \mathrm{CP}_p^r(s)$ *with* $\ell_q^s > 0$, *and* $\forall s \in [r-N, r-1], \forall q \in \mathrm{CP}_p^r(s)$: $\mathsf{x}_q^s = v$, *then* $\texttt{uniqueCandidate}_p^r(r-N, r-1) = v$.

*Proof.* By Lemma 6.4.2 and 6.4.3, $\exists q \in \mathsf{P}_p^r$ with $\mathtt{L}_p^r(q,s) > 0$ and $s \in [r-N, r-1]$, and $\forall q \in \mathsf{P}_p^r$ we have $\mathtt{L}_p^r(q,s) > 0 \Rightarrow \mathtt{X}_p^r(q,s) = v$ for every $s \in [r-N, r-1]$. As stated in Line 24 of Algorithm 6.1, we therefore have $\texttt{uniqueCandidate}_p^r(r-N, r-1) = v$. □

**Lemma 6.4.8.** *Let* $v = \mathsf{x}_p^r$. *The following all hold for all processes* $p$:

*(1) If* $q \in \mathrm{CP}_p^r(s)$ *for a* $s \in [r-N, r-1]$ *and* $\mathsf{x}_q^s \neq v$ *then* $\texttt{latestRefutation}_p^r(r-N, r-1) \geqslant s$.

*(2) If* $\mathsf{x}_q^t = v$ *and* $\ell_q^t > 0$ *for all* $q \in \Pi_\sigma$ *and all* $t \in [s, r]$ *then* $\texttt{latestRefutation}_p^r(r-N, r-1) < s$.

*(3)* $\texttt{latestRefutation}_p^r(r-N, r-1) < r$.

*Proof.* Let $\mathsf{T}_p^r$ denote $p$'s value of $\mathsf{T}$ during a call of $\texttt{latestRefutation}_p^r(r-N, r-1)$ after computing Line 20.

(1) By Lemma 6.4.3, $\mathtt{X}_p^r(q,s) \notin \{-1, v\}$. Since $s \in [r-N, r-1]$, Line 20 of Algorithm 6.1 ensures that $s \in \mathsf{T}_p^r$. Hence $\max(\mathsf{T}_p^r) \geqslant s$ is returned in Line 21.

(2) Suppose that $\texttt{latestRefutation}_p^r(r-N, r-1) = t \geqslant s$. By Line 20 of Algorithm 6.1, $t \in \mathsf{T}_p^r$ with $\mathtt{L}(q,t) = 0$ or $\mathtt{X}(q,t) \notin \{-1, v\}$ for a $q \in \mathsf{P}_p^r$. According to Lemma 6.4.3, hence $\mathsf{x}_q^t \neq v$ or $\ell_q^t = 0$, which contradicts our assumption.

(3) Follows because $\mathsf{T}_p^r \subseteq [r-N, r-1]$ by Line 20 of Algorithm 6.1. □

**Lemma 6.4.9.** *With* $v = \mathsf{x}_p^r$, $\texttt{allGood}_p^r(r-N(D+2N), r-1)$ *is false if and only if* $\exists s \in [r-N(D+2N), r-1]$ *s.t.* $q \in \mathrm{CP}_p^r(s)$ *with* $\ell_q^s = 0$ *or* $\mathsf{x}_q^s \neq v$.

*Proof.* "If direction": By Lemma 6.4.3, $\mathtt{L}_p^r(q,s) = 0$ or $\mathtt{X}_p^r(q,s) \notin \{-1, v\}$ and, by Lemma 6.4.2, $q \in \mathsf{P}_p^r$. Hence, the expression in Line 29 of Algorithm 6.1 is false.

"Only if direction": If $\nexists s \in [r-N(D+2N), r-1]$ s.t. $q \in \mathrm{CP}_p^r(s)$ with $\ell_q^s = 0$ or $\mathsf{x}_q^s \neq v$, by Lemma 6.4.3 for all $s \in [r-N(D+2N), r-1]$, we have $\mathtt{L}_p^r(q,s) \neq 0$ and $\mathtt{X}_p^r(q,s) \in \{-1, v\}$. Hence, Line 29 of Algorithm 6.1 evaluates to true. □

### Algorithm 6.2: Main consensus algorithm

We are now ready to formally prove that Algorithm 6.2 solves consensus. Throughout the remaining proofs, unless stated otherwise, all line numbers are w.r.t. Algorithm 6.2. We start with a formal argument for why an assignment $\mathsf{x}_p \leftarrow v$ of process $p$ in round $r$ via Line 10 has the desired outcome of setting $\mathsf{x}_p$ to the maximum proposal $x_q^{r-D}$ over all processes $q$ that were member of the root component of $G_{r-D}$.

**Lemma 6.4.10.** *If $p$ enters Line 10 in round $r$ then $\mathsf{x}_p^r = \max\{\mathsf{x}_q^{r-D} \mid q \in \mathrm{Root}(G_{r-D})\}$.*

*Proof.* Since the guard of Line 9 has been passed, we have $\mathsf{R}_p^r = \mathtt{searchRoot}_p^r(r-D) \neq \emptyset$. Part (2) of Lemma 6.4.5 asserts that $\mathsf{R}_p^r \subseteq \mathrm{CP}_p^r(r-D)$. By Lemma 6.4.3, for all $q \in \mathsf{R}_p^r$, we hence have $\mathsf{X}_p^r(q, r-D) = \mathsf{x}_q^{r-D}$. Thus, after executing the assignment of Line 10, we have $\mathsf{x}_p^r = \max\{\mathsf{X}_p^r(q, r-D) \mid q \in \mathsf{R}_p^r\}$. This, combined with the observation that $\mathsf{R}_p^r = \mathrm{Root}(G_{r-D})$ by item (1) of Lemma 6.4.5, asserts that $\mathsf{x}_p^r = \max\{\mathsf{x}_q^{r-D} \mid q \in \mathrm{Root}(G_{r-D})\}$ as $x_p$ is not modified after this point. □

For our analysis, we introduce the useful term *$v$-locked root component* to denote a round-$r$ root component where all members $q$ are *$v$-locked* for the same $v$, i.e., have the proposal $\mathsf{x}_q^r = v$ and are locked, i.e., $\ell_q^r > 0$. In a *$v$-locked sequence*, every graph has a (possibly different) $v$-locked root component for the same value $v$.

**Definition 6.4.11.** *$\mathrm{Root}(G_r)$ is a $v$-locked root component if $\forall p \in \mathrm{Root}(G_r)\colon\ \ell_p^r > 0$ and $\mathsf{x}_p^r = v$. A sequence $(G_r)_{r\in I}$ is $v$-locked if for all $r \in I$, $\mathrm{Root}(G_r)$ is a $v$-locked root component.*

The following technical lemma is key for the proof of the agreement property of consensus. The crucial part (2) assures that if a sequence of at least $D + 2N$ communication graphs occurs in which all root components happen to be $v$-locked by our algorithm, then every process $p$'s proposal value is $x_p^r = v$ at every time $r$ following this sequence.

**Lemma 6.4.12.** *Let $\rho = (G_r)_{r=a}^b$ be a $v$-locked sequence with $|\rho| = 2N + D$ and $\rho \subset \sigma \in \Diamond\mathsf{STABLE}_{\leqslant N,D}(D+1)$. Then, all of the following hold for all processes $p \in \Pi_\rho$:*

*(1)* $\forall r \in [a+N+D, b]\colon\ \mathsf{x}_p^r = v \vee \ell_p^r = 0$.

*(2)* $\forall r \geqslant b\colon \mathsf{x}_p^r = v$.

*Proof.* We observe from the code of Algorithm 6.2 that $\mathsf{x}_p^r$ is only written to in Line 10 and Line 16.

(1) Pick an arbitrary $p \in \Pi_\rho$, $r \in [a+N+D, b]$. If $p \in \mathrm{Root}(G_r)$ the claim is immediate because $\mathrm{Root}(G_r)$ is $v$-locked by assumption. Otherwise, if $p$ enters Line 10 or Line 16 at time $r$, we have $\mathsf{x}_p^r = v$. In case of the former, this follows from Lemma 6.4.10 and because $\mathrm{Root}(G_{r-D})$ is $v$-locked by assumption. In case of the latter, because $\rho$ is $v$-locked, by Theorem 3.3.2, $\exists s \in [r-N, r-1], \exists q \in \mathrm{CP}_p^s(r)\colon \ell_q^s > 0 \wedge x_q^s = v$, and hence the claim holds due to (2) of Lemma 6.4.6.

Hence, assume finally that none of line 10 and 16 is entered and that $\mathsf{x}_p^{r-1} \neq v$ and $\ell = \ell_p^{r-1} > 0$, as the claim follows immediately otherwise. We find that then $\ell < a+D$, as an inductive argument shows that $\ell = 0 \vee \mathsf{x}_p^{r-1} = v$ if not: If $\ell \geqslant a+D$, by Lemma 6.4.10 and because $\rho$ is $v$-locked, $\mathsf{x}_p^\ell = v$. Consider time $s \in [\ell+1, r-1]$ and assume the

hypothesis $\ell_p^{s-1} = 0 \vee \mathsf{x}_p^{s-1} = v$. If $p$ enters Line 10, again by Lemma 6.4.10 and because $\rho$ is $v$-locked, $\mathsf{x}_p^s = v$. If $p$ enters Line 16 and $\mathsf{x}_p^s = v' \neq v$, we have $\ell_p^{s-1} = \ell = 0$ because, if $\ell_p^{s-1} > 0$, then by hypothesis $\mathsf{x}_p^{s-1} = v$ and hence $v' = v$ by item (2) of Lemma 6.4.6 and because $p \in \mathrm{CP}_p^{s-1}(s)$.

As Line 10 was not entered and clearly $r > N$, $p$ passes the guard of Line 12. Since $\rho$ is $v$-locked, by Theorem 3.3.2, if $p \notin \mathrm{Root}(G_r)$, there is a $q \in \mathrm{CP}_p^r(s)$ and a $s \in [r-N, r-1] \subseteq [a+D, b]$ with $\ell_q^s > 0$ and $\mathsf{x}_q^s = v$. But according to Lemma 6.4.8, we then have $\mathtt{latestRefutation}_p^r(r-N, r-1) \geqslant s > \ell_p^{r-1}$ and thus Line 14 is executed, setting $\ell_p^r \leftarrow 0$.

(2) We use induction on $r \geqslant b$. For $r = b$, assume $p \notin \mathrm{Root}(G_b)$ as otherwise the claim follows because $\rho$ is $v$-locked. If $p$ enters Line 10 in round $b$, $x_p^b = v$ because $\rho$ is $v$-locked. Otherwise, by Theorem 3.3.2, as $p \notin \mathrm{Root}(G_b)$, there is a $s \in [r-N, r-1]$ and a $q \in \mathrm{CP}_p^r(s)$ with $\mathsf{x}_q^s = v$ and $\ell_q^s > 0$. By (1), $\forall s \in [r-N, r-1], \nexists q \in \mathrm{CP}_p^r(s)\colon\ \mathsf{x}_q^s \neq v$ and $\ell_q^s > 0$. Hence, Line 16 is executed and $\mathsf{x}_p^b = v$ by Lemma 6.4.7.

For $r \geqslant b$, by the induction hypothesis, it suffices to show that all modifications of $\mathsf{x}_p$ at process $p$ during round $r$ do not invalidate the claim. If $p$ enters Line 10 in round $r$, $\mathsf{x}_p \leftarrow v$ is assigned according to Lemma 6.4.10 and because either $R = \mathrm{Root}(G_{r-D})$ is $v$-locked by assumption (for $r - D \geqslant b$), or else $q \in R \Rightarrow \mathsf{x}_q^{r-D} = v$ by the induction hypothesis. If $p$ passes the guard of Line 16, because of (1) and because of the induction hypothesis, we have $\mathsf{x}_p^r = v$ due to item (1) of Lemma 6.4.6. □

With these preparations, we can now prove agreement, validity and termination of our consensus algorithm.

**Lemma 6.4.13.** *Algorithm 6.2 ensures agreement under each communication pattern* $\sigma \in \Diamond\mathsf{STABLE}_{\leqslant N,D}(D+1)$.

*Proof.* We show that if a process $p$ decides $v$ in round $r$, all future decisions are $v$ as well. A decision $y_p^r \leftarrow v$ by $p$ in round $r$ can only occur if $p$ executed Line 18, thus $p$ was undecided, $r > N(D+2N)$, and $\mathtt{allGood}_p^r(r - N(D+2N), r-1)$ is true. We show that this implies that there is a $v$-locked sequence $\sigma \subseteq (G_r)_{r=r-N(D+2N)}^{r-1} = \sigma'$ with $|\sigma| > 2N + D$. This shows the claim as a decision by $q$ in a round $s \geqslant r$ is based on $\mathsf{x}_q^s$ and $\mathsf{x}_q^s = v$ by item (2) of Lemma 6.4.12.

Suppose $\sigma'$ does not contain such a sequence $\sigma$. By the pigeonhole principle, there must be a set of rounds $S = \{r_1, \dots, r_N\} \subseteq [r - N(D+2N), r-1]$ such that for each $r_i \in S$, a $q_i \in \mathrm{Root}(G_{r_i})$ has $\ell_{q_i}^{r_i} = 0 \vee \mathsf{x}_{q_i}^{r_i} \neq v$. Setting $f(i) = q_i$ and $G = \{G_{r_1}, \dots, G_{r_n}\}$ allows us to apply Theorem 3.3.2. There are two cases:

If $p = q_n$, $\mathsf{x}_p^{r_n} \neq v \vee \ell_p^{r_n} = 0$. But then either the check $\ell_p^r > 0$ fails (if $r = r_n$) or $\mathtt{allGood}_p^r(r - N(D+2N), r-1)$ is false by Lemma 6.4.9 and because $p \in \mathrm{CP}_p^r(r_n)$ (if $r > r_n$).

If $p \neq q_n$, by Theorem 3.3.2, $\exists i \in [1,n]\colon q_i \in \mathrm{CP}_p^r(r_i)$ with $\ell_{q_i}^{r_i} = 0 \vee \mathsf{x}_{q_i}^{r_i} \neq v$. Thus, again by Lemma 6.4.9, $\texttt{allGood}_p^r(r - N(D+2N), r-1)$ is false. □

**Lemma 6.4.14.** *Algorithm 6.2 ensures validity under all communication patterns $\sigma$ of $\Diamond\mathsf{STABLE}_{\leqslant N,D}(D+1)$.*

*Proof.* Validity follows from an induction on the time $r$, as all processes $p$ decide on the value of $\mathsf{x}_p^r$.

Initially, $\forall p \in \Pi_\sigma\colon \mathsf{x}_p^0 = x_p$ where $x_p$ is the input value of process $p$ by Line 1.

For all times $r > 0$ let the hypothesis be $\forall p \in \Pi_\sigma, \forall s \in [0, r-1]\colon\ \mathsf{x}_p^s = x_q$ for an input value $x_q$. Assume $p$ assigns $\mathsf{x}_p \leftarrow v$ in round $r$. As this assignment is via Line 10 or Line 16, $v = \mathsf{x}_p^r(q,s) \neq -1$ for a round $s < r$ and a process $q$. By Lemma 6.4.3, $v = \mathsf{x}_q^s$, which is the input $x_{q'}$ of a process $q'$ by the induction hypothesis. □

**Lemma 6.4.15.** *Algorithm 6.2 terminates under all communication patterns $\sigma$ of $\Diamond\mathsf{STABLE}_{\leqslant N,D}(D+1)$.*

*Proof.* Since $\sigma \in \Diamond\mathsf{STABLE}_{\leqslant N,D}(D+1)$, there is an earliest subsequence $\sigma' = (G_r)_{r=a}^b \subset \sigma$ with $|\sigma'| = D+1$ that has a common root $R$ and hence, for $a > 1$, $R' = \mathrm{Root}(G_{a-1}) \neq R$; for $a = 1$, we just set $R' = \emptyset$. Let $v = \max\{\mathsf{x}_q^a \mid q \in R\}$. We show by induction that for all processes $p$ and for all times $r \geqslant b$, $\ell_p^r \geqslant b$, and $\mathsf{x}_p^r = v$. This shows the claim as, at the latest in round $s = b + N(D+2N)$, $\texttt{allGood}_p^s(b, s-1)$ is true by Lemma 6.4.9 at every undecided process $p$, leading to $p$ deciding via Line 18.

In round $b$, for all processes $p$, $\texttt{searchRoot}_p^b(b-D) = R \neq \emptyset$ by (2) of Lemma 6.4.4. Furthermore, $\texttt{searchRoot}_p^b(b-D-1) = \emptyset$ if $R' = \emptyset$ and $\texttt{searchRoot}_p^b(b-D-1) = R' \neq R$ by Lemma 6.4.5. Hence $p$ passes Line 9 and enters Line 10 and Line 11. Thus $\ell_p^b = b$, and, by Lemma 6.4.10, $\mathsf{x}_p^b = v$.

Pick a round $r > b$. Using the induction hypothesis, it suffices to show that a modification of $\ell_p$ resp. $\mathsf{x}_p$ in round $r$ does not invalidate our claim.

If either is modified by Line 10 or Line 11, since by the hypothesis $\ell_p^{r-1} > 0$, we must have $\texttt{searchRoot}_p^r(r-D-1) \neq \texttt{searchRoot}_p^r(r-D)$. As $\texttt{searchRoot}_p^r(b-D) = R \neq \emptyset$ and $\sigma' = (G_i)_{i=a}^b$ is $R$-rooted, by item (3) of Lemma 6.4.5, $r > b + D$. By the hypothesis, thus $\mathsf{x}_q^{r-D} = v$ for all $q \in \mathrm{Root}(G_{r-D})$ and hence $\mathsf{x}_p^r = v$ by Lemma 6.4.10.

If $\ell_p$ is set to 0 in Line 14, recall that the induction hypothesis guarantees $\mathsf{x}_q^s = v$ and $\ell_q^s \geqslant b$ for all processes $q$ and every time $s \in [b, r-1]$. Thus $\texttt{latestRefutation}(r-N, r-1) < b$ by Lemma 6.4.8 (2), and the check in Line 13 fails.

If $\mathsf{x}_p$ is modified by Line 16, we have $\mathsf{x}_p^r = v$ because of (2) of Lemma 6.4.6, as process $p$ itself satisfies $p \in \mathrm{CP}_p^{r-1}(r)$ by Definition 3.3.1, and $\ell_p^{r-1} \geqslant b$ and $\mathsf{x}_p^{r-1} = v$ by the hypothesis. □

Lemmas 6.4.13, 6.4.14, and 6.4.15 yield the correctness of Algorithm 6.2:

**Theorem 6.4.16.** *Algorithm 6.2 solves consensus under all* $\sigma \in \Diamond\mathsf{STABLE}_{\leqslant N,D}(D+1)$.

Chapter 7

# Beyond Rootedness and Consecutive Stability

So far, we have studied how the duration of a stability phase affects consensus solvability in message adversaries where every communication graph is rooted. In this section, we extend these results to adversaries that do not satisfy all properties of $\Diamond\mathsf{STABLE}_{\leqslant N,D}$ but only some of those.

**Chapter Organization** First, inspired by [CBFN16], we study the case where the adversary guarantees only a constant fraction of the communication graphs to be rooted. This turns out to be a rather benign case, as our previously established techniques are almost directly applicable here. Second, we look into the case where we do not require a constant fraction of communication graphs to be rooted. For this purpose, we introduce $\mathsf{STICKY}_n(x)$, a message adversary that guarantees that all root components $R$ that are vertex-stable for more than $x$ rounds have an $R$-rooted subsequence with a duration $\geq x$ during their lifetime. This means that there may be many root components that are overlapping, in the sense that they are vertex-stable simultaneously, but if one of them is stable long enough, it must become the only stable root component for a certain duration. Third, we investigate the impact of the duration of non-consecutive vertex-stability. We assume that this duration is fixed to $D$ rounds and study the solvability of consensus in relation to the parameter of $x$ of $\mathsf{STICKY}_n(x)$. Finally, we introduce a consensus algorithm that shows that consensus is solvable if the above impossibility conditions are not met.

## 7.1 Related Work

Whereas there is some work that studies what happens if the assumption that all communication graphs are rooted is dropped, getting rid of it is either almost trivial or, when done in a general way, introduces a lot of complex formalisms that may distract from the main point. As far as we are aware, there is thus not too much related work on this topic; apart from our own work [SWS16, BRS+18], on which most of this chapter

is based, [CBFN16] has a section on how the solvability of approximate agreement is affected if only a constant fraction of the communication graphs is rooted, which inspired Section 7.2. Regarding the relation between the number of simultaneous vertex-stable source components and the solvability of $k$-set agreement, [CFP+19] recently obtained a closely related result by introducing combinatorial topology to the dynamic network model. Among other interesting results, such as a lower bound on $\varepsilon$ for $\varepsilon$-approximate agreement and termination time lower bounds for $k$-set agreement, it was shown that if the domination number of the information-flow graph (the product graph of a communication pattern) is $> k$, then $k$-set agreement is impossible.

## 7.2 Dealing with a Constant Fraction of Rooted Graphs

We start our investigation with an adversary that does not satisfy the conditions of $\mathsf{ROOTED}_n$ in that it contains sequences where not all communication graphs are rooted graphs. Recall that this means that some communication graphs may contain multiple strongly connected components such that there is no directed path between them. In this sense, these strongly connected components define a partitioning of the system (even though the components may still be weakly connected to each other, as in Definition 3.2.3). Below we show how, depending on the severity of this partitioning, consensus may still solvable with the techniques described so far. In particular, it turns out that our methods are still applicable if there remains a constant fraction of rooted graphs in every communication pattern. In order to precisely express this type of partitioning, we introduce the message adversary $\mathsf{ROOTED}_n(k)$.

**Definition 7.2.1.** $\mathsf{ROOTED}_n(k)$ *is the set of those infinite communication patterns* $\sigma$ *where each subsequence* $\sigma' \subset \sigma$ *with* $|\sigma'| = k$ *contains at least one rooted communication graph.*

By definition, $\mathsf{ROOTED}_n(k)$ describes those graph sequences where at least every $k^{\text{th}}$ communication graph is rooted. For this kind of relaxation of $\mathsf{ROOTED}_n$, it is actually easy to adapt Algorithm 6.2. The reason for this is that $\Diamond\mathsf{GOOD}_n(D+1)$ still guarantees a rooted subsequence $\sigma$ of $D+1$ consecutive rounds and the more relaxed condition $\mathsf{ROOTED}_n(k)$ merely implies that we can rely only on every $k^{\text{th}}$ graph to be rooted outside $\sigma$. It follows that essentially waiting $k$ times longer in Algorithm 6.2 will result in the same amount of rooted graphs and thus the same level of information propagation throughout the execution. In the code, this simply amounts to increasing all durations by a factor of $k$, as presented in Algorithm 7.1.

Regarding the impossibility results from Section 6.3, it is not hard to see that they hold also for $\mathsf{ROOTED}_n(k)$. The reason for this is that the adversary may simply choose completely disconnected graphs (with self-loops) in all rounds except in the stability phase and in one round of every subsequence of length $k$, thereby satisfying $\mathsf{ROOTED}_n(k)$. This ensures that there is no information propagation in those disconnected rounds, which makes a straightforward alteration of Theorem 6.3.6 immediately applicable. As,

**Algorithm 7.1:** Consensus algorithm for $\mathsf{ROOTED}_n(k) \cap \Diamond\mathsf{GOOD}_n(D+1) \cap \mathsf{DEPTH}_n(D)$, code for process $p$. Uses function definitions from Algorithm 6.1.

**Initialization**:
$r \leftarrow 0$, $\mathsf{x} \leftarrow x_p$, $\ell \leftarrow 0$, $\mathsf{A} \leftarrow \emptyset$, $\mathsf{P} \leftarrow \emptyset$
$\mathsf{S} \leftarrow \{(p, r, \mathsf{x}, \ell)\}$
$r \leftarrow 1$

**Round $r$ communication**:
Attempt to send $(\mathsf{P}, \mathsf{S}, \mathsf{A})$ to all
Receive $m_q$ from all $q$ with $(q, p) \in G_r$

**Round $r$ computation**:
**foreach** $m_q$ *s.t.* $p$ *received* $m_q = (\mathsf{P}_q, \mathsf{S}_q, \mathsf{A}_q)$ *in round* $r$ **do**
  `update`$(q, \mathsf{P}_q, \mathsf{S}_q, \mathsf{A}_q)$
$\mathsf{R} \leftarrow$ `searchRoot`$(r - D)$
**if** $\mathsf{R} \neq \emptyset$ *and* ($\ell = 0$ *or* $\mathsf{R} \neq$ `searchRoot`$(r - D - 1)$) **then**
  $\mathsf{x} \leftarrow \max\{\mathtt{X}(q, r - D) \mid q \in \mathsf{R}\}$
  $\ell \leftarrow r$
**else if** $r > kN$ **then**
  **if** `latestRefutation`$(r - kN, r - 1) \geqslant \ell$ **then**
    $\ell \leftarrow 0$
  **if** `uniqueCandidate`$(r - kN, r - 1) \neq -1$ **then**
    $\mathsf{x} \leftarrow$ `uniqueCandidate`$(r - kN, r - 1)$
**if** $r > kN(D + 2N)$, $y_p = \perp$, $\ell > 0$, *and* `allGood`$(r - kN(D + 2N), r - 1) =$ TRUE **then**
  $y_p \leftarrow \mathsf{x}$
$\mathsf{S} \leftarrow \mathsf{S} \cup (p, r, \mathsf{x}, \ell)$
$r \leftarrow r + 1$

furthermore, the arguments for the correctness of Algorithm 6.2 can be adapted to Algorithm 7.1, we have the following lemma.

**Lemma 7.2.2.** *Fix some $k > 0$. Consensus is solvable under $\mathsf{ROOTED}_n(k) \cap \Diamond\mathsf{GOOD}_n(x) \cap \mathsf{DEPTH}_n(D)$ if and only if $x > D$.*

## 7.3 Lower Bounds for Partitioned Graphs

So far, we have seen that a constant fraction of rooted graphs is a very benign case of a partitioning in the sense that essentially all our previous results remain applicable. As we have motivated in the beginning, however, we would like to be able to model systems that can cope with arbitrary long periods of initial uncertainty. The question that we answer in the remainder of this section is how this can be realized with respect a partitioning that remains for an arbitrary long period. For this reason, we introduce the adversary $\mathsf{STICKY}_n(x)$ below. Informally, it restricts a partitioning only in that any vertex-stable root component that is stable for $\geq x$ rounds is the only root component at some point during its lifetime for at least $x$ consecutive rounds.

**Definition 7.3.1.** *For $x \geq 1$, the message adversary $\mathsf{STICKY}_n(x)$ is defined as follows: $\mathsf{STICKY}_n(1) = \mathsf{ROOTED}_n$ and for $x > 1$, $\mathsf{STICKY}_n(x)$ is the set of all infinite communication patterns $\sigma$ such that for every subsequence $\sigma' = (G_r)_{r=a}^{b} \subseteq \sigma$ with $|\sigma'| \geq x$, if there is a set $R \subseteq \Pi_\sigma$ such that $R$ is a root component of every $G_r$ with $r \in [a, b]$ but neither of $G_{a-1}$ nor $G_{b+1}$, then there is an $R$-rooted sequence $\sigma'' \subseteq \sigma'$ with $|\sigma''| \geq x$.*

By definition, $\mathsf{STICKY}_n(1)$ is equivalent to $\mathsf{ROOTED}_n$. For $x > 1$, Definition 7.3.1 essentially states that every sequence of length $\geq x$ in $\mathsf{STICKY}_n(x)$ that consists of communication graphs in which $R$ occurs as a root component in fact contains an $R$-rooted subsequence of length $x$. This permits vertex-stable root components that behave almost arbitrarily chaotic if their duration is $< x$ rounds and also allows multiple root components to some extent, even if their duration is $\geq x$. It does, however, provide a very important guarantee that will be exploited algorithmically later on: The first vertex-stable root component that remains stable for $\geq x$ rounds, must become the only vertex-stable root component for a phase of $\geq x$ rounds *before* any subsequent long-lived vertex-stable root components come into existence. A more precise version of this statement is given in the following Corollary 7.3.2.

**Corollary 7.3.2.** *Pick an arbitrary $\sigma \in \mathsf{STICKY}_n(x)$ and let $\sigma' \subseteq \sigma$ be the earliest subsequence of $\sigma$ with $|\sigma'| \geq x$ such that there is some $R \subseteq \Pi_\sigma$ and $R$ is a root component in every communication graph of $\sigma'$. There is some $\sigma'' \subseteq \sigma'$ such that $|\sigma''| = x$, $\sigma''$ is $R$-rooted and there exists no earlier $R'$-rooted subsequence with a duration $\geq x$ for any $R' \subseteq \Pi_\sigma$.*

*Proof.* Pick as $\sigma''$ the earliest $R$-rooted subsequence of $\sigma'$ with $|\sigma''| = x$, as guaranteed Definition 7.3.1. Suppose there exists an $R' \subseteq \Pi$ and an $R'$-rooted subsequence $\sigma'''$ with $|\sigma'''| = x$ that occurs before $\sigma''$ in $\sigma$. Since $\sigma''$ is $R$-rooted, there is no root component $R' \neq R$ in any of its communication graphs and thus there is an $R'$-rooted subsequence of $\sigma$ with duration $\geq x$ that happens before $\sigma'$. This contradicts the assumption that $\sigma'$ is the earliest such subsequence. □

As we will see below, solving consensus in systems that are partitioned in the sense of Definition 7.3.1 incurs the necessity for longer periods of stability than just $D+1$ rounds. In order to show this, we use $\Diamond\mathsf{GOOD}_n(y)$ from Definition 6.2.2 to introduce the combined message adversary $\Diamond\mathsf{STABLE}_{\leqslant N,D}(x, y)$, which lets us formulate a generalized version of $\Diamond\mathsf{STABLE}_{\leqslant N,D}(x)$ that can express partitioned communication graphs:

**Definition 7.3.3.** $\Diamond\mathsf{STABLE}_{\leqslant N,D}(x, y) := \mathsf{STICKY}_n(x) \cap \Diamond\mathsf{GOOD}_n(y) \cap \mathsf{DEPTH}_n(D)$.

It is not hard to see that, because $\mathsf{ROOTED}_n = \mathsf{STICKY}_n(1)$ by convention, we have $\Diamond\mathsf{STABLE}_{\leqslant N,D}(1, y) = \Diamond\mathsf{STABLE}_{\leqslant N,D}(y)$.

In order to extend the characterization of $\Diamond\mathsf{STABLE}_{\leqslant N,D}(x, y)$ to the entire range of its parameters, we need to investigate the cases where $x > 1$. When setting $x > 1$

in $\mathsf{STICKY}_n(x)$, however, we unfortunately cannot simply apply the techniques from Algorithm 7.1 and Theorem 6.3.1 again, but instead need to develop new algorithms and impossibility results. We will find that a significantly longer period of stability than $y = D + 1$ is required to solve consensus in this case: In Lemma 7.3.4 below, we find that increasing consecutive stability to $y = 2D$ still renders consensus impossible due to the partitioning that occurs when $x > 1$.

We prove this lower bound for $\mathsf{MA} = \bigcap_{i>1} \mathsf{STICKY}_n(i) \cap \Diamond\mathsf{GOOD}_n(2D) \cap \mathsf{DEPTH}_n(D)$, which suffices since $\mathsf{MA} \subseteq \Diamond\mathsf{STABLE}_{\leqslant N,D}(x, 2D)$ for $x > 1$. Glossing over many of the details for now, $\mathsf{MA}$ may generate $\sigma_1$ and $\sigma_4$ from Figure 7.1, resulting in admissible executions $\varepsilon_1 = \langle C^0, \sigma_1 \rangle$ and $\varepsilon_4 = \langle C^0, \sigma_4 \rangle$, where $C^0$ is an initial configuration in which the processes of $A = \{a, a'\}$ and the processes of $B = \{b, b'\}$ start with different inputs. The validity condition implies that in any correct consensus algorithm, by some round $\tau$, $a$ (and hence also $b$) decided a different value in $\varepsilon_1$ than $b$ (and hence also $a$) in $\varepsilon_4$. Furthermore, $\mathsf{MA}$ may generate $\sigma_2$ and $\sigma_3$ from Figure 7.1 where $\varepsilon_2 = \langle C^0, \sigma_2 \rangle$ and $\varepsilon_3 = \langle C^0, \sigma_3 \rangle$ result in the indistinguishabilities $\varepsilon_1 \sim_b \varepsilon_2 \sim_c \varepsilon_3 \sim_a \varepsilon_4$. This implies that $a$ and $b$ decided differently in $\varepsilon_2$ and $\varepsilon_3$ and hence $c \in C$ can never make a correct decision in $\varepsilon_2$ or $\varepsilon_3$.

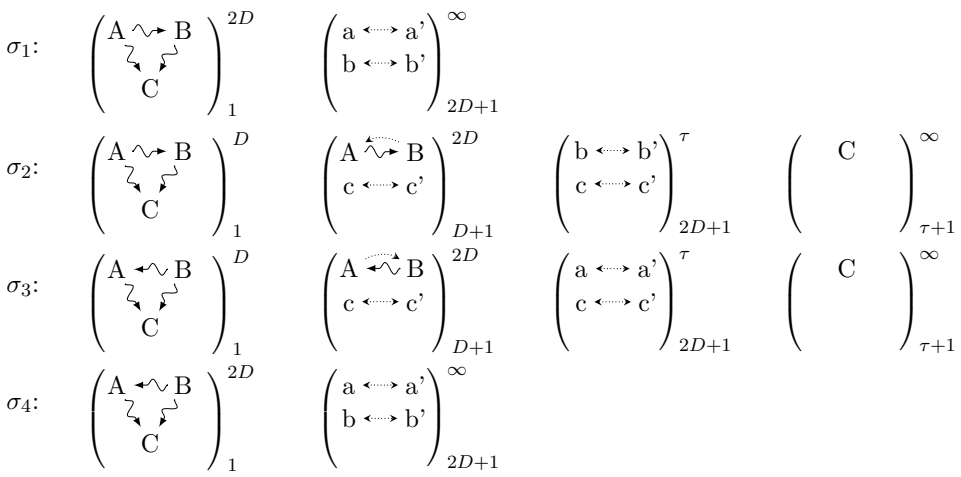


Figure 7.1: Sequences of Lemma 7.3.4. We use an uppercase letter $X$ to denote the graph $x \leftrightarrow x'$. For $D > 1$, a "wavy" edge $X \rightsquigarrow Y$ stands for a chain $X \to p_{i_1} \to \ldots \to p_{i_{D-1}} \to Y$ where $i_j \neq i_{j'}$ for $j \neq j'$. In this context, $X \to p$ (resp. $p \to X$) stands for the presence of the two edges $(x, p)$ and $(x', p)$ (resp. $(p, x)$ and $(p, x')$). The chain is unique wrt. $X$ and $Y$ and its vertex-set is disjoint from that of any other chain, denoted by a "wavy" line in the same graph. For $D = 1$, $X \rightsquigarrow Y$ stands for the collection of edges $(x, y), (x', y), (x, y'), (x', y')$. A dotted, unidirectional edge $x \dashrightarrow y$ represents an edge that is in $G_i$ iff it is not in $G_{i-1}$. A dotted, bidirectional edge $x \dashleftarrow\dashrightarrow y$ means that there is a single unidirectional edge between $x$ and $y$ that alternates its direction in every round. We assume that there is an edge from each process (or set of processes) depicted in the graph to every process not depicted in the graph.

**Lemma 7.3.4.** *Fix some integer $x > 1$. If $y \leq 2D$ then consensus is impossible under $\Diamond\mathsf{STABLE}_{\leqslant N,D}(x, y)$.*

*Proof.* As for $y' > y$, $\Diamond\mathsf{GOOD}_n(y') \subset \Diamond\mathsf{GOOD}_n(y)$, it suffices to show that consensus is impossible under message adversary $\mathsf{MA} = \bigcap_{i>1} \mathsf{STICKY}_n(i) \cap \Diamond\mathsf{GOOD}_n(2D) \cap \mathsf{DEPTH}_n(D)$.

For a contradiction, assume that a correct consensus algorithm $\mathcal{A}$ exists for $\mathsf{MA}$. Let $C^0$ be some initial configuration with input values $x_a = x_{a'} \neq x_b = x_{b'}$. We use the (admissible) executions $\varepsilon_1 = \langle C^0, \sigma_1 \rangle$, $\varepsilon_2 = \langle C^0, \sigma_2 \rangle$, $\varepsilon_3 = \langle C^0, \sigma_3 \rangle$, and $\varepsilon_4 = \langle C^0, \sigma_4 \rangle$ of $\mathcal{A}$ with $\sigma_1$, $\sigma_2$, $\sigma_3$, and $\sigma_4$ as shown in Figure 7.1 (using the notation explained in the caption of the figure). Inspecting our executions will reveal that process $a$ and hence also $b$ decides $x_a$ in $\varepsilon_1$, $b$ and hence also $a$ decides $x_b$ in $\varepsilon_4$, and $\varepsilon_1 \sim_b \varepsilon_2 \sim_c \varepsilon_3 \sim_a \varepsilon_4$.

By $\varepsilon_1 \sim_b \varepsilon_2$, since $\mathcal{A}$ is correct, $b$ decides on $x_a$ in $\varepsilon_2$. By $\varepsilon_2 \sim_c \varepsilon_3$, $c$ decides on $x_a$ also in $\varepsilon_3$. Furthermore, by $\varepsilon_3 \sim_a \varepsilon_4$, $a$ decides on $x_b$ in $\varepsilon_3$. This, however, contradicts the correctness of $\mathcal{A}$ due to the agreement condition and because $x_a \neq x_b$.

We finish the proof by arguing why the executions are admissible, why $a$ and $b$ decide $x_a$ in $\varepsilon_1$, why $a$ and $b$ decide $x_b$ in $\varepsilon_4$, and why the claimed indistinguishabilities hold.

$\varepsilon_1, \varepsilon_2, \varepsilon_3, \varepsilon_4$ are admissible under $\mathsf{DEPTH}_n(D)$, since their respective sequences adhere to the dynamic network depth $D$ by construction. Executions $\varepsilon_1, \varepsilon_4$ are admissible under $\Diamond\mathsf{GOOD}_n(2D)$, because the first $2D$ rounds of $\sigma_1$ are $\{a, a'\}$-rooted and the first $2D$ rounds of $\sigma_4$ are $\{b, b'\}$-rooted. Executions $\varepsilon_2, \varepsilon_3$ are admissible under $\Diamond\mathsf{GOOD}_n(2D)$, because their sequences have a suffix which is forever $\{c, c'\}$-rooted. Note that in all these executions, for $i > 1$, if a root component $R$ occurs in every graph of a subsequence of length $i$, then this subsequence is $R$-rooted. Hence, all executions are admissible under $\bigcap_{i>1} \mathsf{STICKY}_n(i)$.

Since $\varepsilon_1$ is admissible and $\mathcal{A}$ is correct, eventually $a$ will decide in $\varepsilon_1$. Since $a$ never received a message from a process $p$ with $x_p \neq x_a$, $\varepsilon_1$ is indistinguishable for $a$ from an execution where all processes started with input value $x_a$ and hence $a$ will decide $x_a$ due to the validity condition of consensus. Consequently, $b$ will also eventually decide $x_a$ in $\varepsilon_1$. The symmetric argument shows that $b$ and $a$ eventually decide $x_b$ in $\varepsilon_4$.

We now show the indistinguishability $\varepsilon_1 \sim_b \varepsilon_2$ (we can employ the symmetric argument to show $\varepsilon_3 \sim_a \varepsilon_4$). Recall that by round $\tau$ it is guaranteed that $b$ has already decided in $\varepsilon_1$. The existence of $\tau$ is guaranteed by the admissibility of $\varepsilon_1$. Let $P(r)$ denote the processes that have observed a difference between $\varepsilon_1$ and $\varepsilon_2$ by round $r$. We need to show that $P(r') \cap \mathrm{CP}_b^r(r') = \emptyset$, for all rounds $r' \leqslant r < \tau$. Since the executions are the same for the first $D$ rounds, we have $P(r) = \emptyset$ for $r \leqslant D$ and, clearly, $P(D+1) = A$. Since $B$ is reachable from $A$ during the interval $[D+2, 2D]$ only via $D$ hops, and $b$ receives messages only from $b'$ after round $2D$ until $\tau$, the claim follows by a simple induction.

We conclude the proof by showing $\varepsilon_2 \sim_c \varepsilon_3$. Let $P'(r)$ denote those processes that have observed a difference between $\varepsilon_2$ and $\varepsilon_3$ by the end of round $r$. Hence $p$ observes a

difference between the two executions iff there is some process $q$ and there are some rounds $r \geqslant r'$ s.t. $q \in P'(r')$ and $q \in \mathrm{CP}_p^r(r')$.

Clearly, $P'(1)$ contains the vertices of the chain from $A$ to $B$ as it changes the direction. Yet, according to Figure 7.1, the shortest path from $A$ to $C$ (resp. from $B$ to $C$) from round 2 to round $D$ of the execution is via $D$ hops. Afterwards, the only in-edges incident to $\{c, c'\}$ are from $\{c, c'\}$. Again a simple induction shows that because of this $P'(r') \cap \mathrm{CP}_c^r(r') = \emptyset$, for all rounds $r' \leqslant r$.

Assuming $n > 3D + 3$ allows the adversary to construct the executions from Figure 7.1. Hence, consensus is impossible if $N \geqslant 3D + 3$, i.e., for almost all values of $N$. □

## 7.4 Lower Bounds for Non-consecutive Stability

As we have just seen, even under a slightly partitioned message adversary, we require already a stability phase that is about twice as long as in the case where every communication graph is rooted. A natural question that arises in this context is whether this vertex-stability really needs to be consecutive, as we have required so far. In order to investigate this, i.e., to model the behavior of partly consecutive stability, we introduce a generalized variant of $\Diamond\mathsf{GOOD}_n(x)$ below.

**Definition 7.4.1.** *$\Diamond\mathsf{GOOD}_n(y, z)$ is the set of all infinite communication patterns $\sigma$ such that $|\Pi_\sigma| = n$ and there exists a set $R \subseteq \Pi_\sigma$ and an $R$-rooted $\sigma' \subseteq \sigma$ with $|\sigma'| \geqslant y$. In addition, the subsequence $\sigma'$ is followed by a finite subsequence $\sigma'' \subseteq \sigma$ such that at least $z$ communication graphs $G_r$ of $\sigma''$ satisfy $\mathrm{Root}(G_r) = R$.*

Definition 7.4.1 allows us to express that an $R$-rooted sequence of length at least $y$ is followed by $z$ not necessarily consecutive communication graphs with unique root component $R$. It leads to our final, most general definition of the eventually stabilizing message adversary:

**Definition 7.4.2.** $\Diamond\mathsf{STABLE}_{\leqslant N,D}(x, y, z) := \mathsf{STICKY}_n(x) \cap \Diamond\mathsf{GOOD}_n(y, z) \cap \mathsf{DEPTH}_n(D)$.

A particularly interesting value for the second parameter $z$ of $\Diamond\mathsf{GOOD}_n(y, z)$ seems to be $z = D$: According to Definition 6.2.1, this is just enough for the state information of the consecutively vertex-stable root component to propagate from the end of its stability phase to all other processes. Our lower bound, expressed in Lemma 7.4.3 below, states that consensus is impossible under $\Diamond\mathsf{STABLE}_{\leqslant N,D}(x, y, D)$ if $y < x$. The reason is that the information propagation during a stability phase of $y$ rounds, where $y < x$, is insufficient for some process to reliably detect that the stability window has occurred. Subsequently, we will show that this is a tight bound in the sense that consensus becomes solvable if $y \geq x$.

**Lemma 7.4.3.** *Fix integers $x > 1$ and $y > 0$. Consensus is impossible under the message adversary $\Diamond\mathsf{STABLE}_{\leqslant N,D}(x, y, D)$ if $y < x$.*

*Proof.* We show the claim for $n \geqslant 4$ and $D < n - 2$, which shows that consensus is impossible for almost all values of $n$ and $D$. Assume for a contradiction that some algorithm $\mathcal{A}$ exists that solves consensus under $\Diamond\mathsf{STABLE}_{\leqslant N,D}(x, y, D)$ for $y < x$, and hence also in the following execution $\varepsilon$ with graph sequence $\sigma$: Every process starts with input value 0 and, for the first $x - 1 \geq y$ rounds, $(G_r)_{r=1}^{x-1}$ is $R$-rooted. Then, the communication graphs alternate between being $R'$-rooted and $R$-rooted for some root component $R' \neq R$. Additionally, there are two distinct processes $p$ and $q$ that have only incoming edges from some process $\notin \{p, q\}$ and no outgoing edges throughout the entire execution $\varepsilon$. The actual communication graphs outside $R$ are such that they are in accordance with $\mathsf{DEPTH}_n(D)$.

Since every $G_r$ in $\sigma$ is rooted, $\sigma$ is feasible for $\Diamond\mathsf{STABLE}_{\leqslant N,D}(y, D)$ because $(G_r)_{r=1}^{x-1}$ is $R$-rooted and due to the communication graphs $G_{x+1}, G_{x+3}, \ldots, G_{x+2D-1}$ where $R$ re-appears $D$ times. In addition, as $\sigma$ does not contain an $R$-rooted subsequence with a duration $\geq x$ rounds, it trivially satisfies $\mathsf{STICKY}_n(x)$ as well. By the assumed correctness of $\mathcal{A}$ under $\Diamond\mathsf{STABLE}_{\leqslant N,D}(x, y, D)$, there is hence some round $\tau$ by which every process must have terminated correctly.

Now consider the following execution $\varepsilon'$, with graph sequence $\sigma'$: Each process of $\Pi \setminus \{p, q\}$ starts with input value 0, while $p$ and $q$ start with 1. For every $G_r$ of $(G_r)_{r=1}^{\tau}$ in $\sigma'$, the induced subgraph of $\Pi \setminus \{p, q\}$ is the same as in $\sigma$. By contrast, the processes $p$ and $q$ are now connected only with each other: There is an edge $(q, p)$ in every $G_r$ and, additionally, an edge $(p, q)$ in every $G_r$ where $r$ is even. Finally, the graph sequence $(G_r)_{r=\tau+1}^{\infty}$ forever repeats the star-graph $G$, where the center $p$ has no in-edges and an out-edge to every other process.

Clearly, $\sigma'$ is feasible for $\Diamond\mathsf{STABLE}_{\leqslant N,D}(y, D)$ due to the star-graph sequence $(G_r)_{r=\tau+1}^{\infty}$. Moreover, $(G_r)_{r=\tau+1}^{\infty}$ is the only $R$-rooted subsequence of $\sigma'$ with a consecutive duration $\geq x$. Since $(G_r)_{r=\tau+1}^{\infty}$ is $R$-rooted, $\sigma'$ is feasible for $\mathsf{STICKY}_n(x)$. Since also the dynamic diameter $D$ is adhered to in $\sigma'$, we have thus that $\sigma'$ is feasible for $\Diamond\mathsf{STABLE}_{\leqslant N,D}(x, y, D)$.

We observe that all processes of $\Pi \setminus \{p, q\}$ have the same state in both $\varepsilon$ and $\varepsilon'$ at the end of round $\tau$. Hence, all decide 0 in $\varepsilon'$ as they do in $\varepsilon$. For $p$ and $q$, $\varepsilon'$ is indistinguishable from the execution $\varepsilon''$, which applies $\sigma'$ to the initial configuration where every process started with input value 1. Consequently, $p$ cannot make a safe decision in $\varepsilon'$: If it decides 1, it violates agreement w.r.t. $\varepsilon$, if it decides 0, it violates validity w.r.t. $\varepsilon''$. This contradicts the assumption that $\mathcal{A}$ is a correct consensus algorithm. □

## 7.5 A Solution Algorithm

We now show that consensus is solvable under $\Diamond\mathsf{STABLE}_{\leqslant N,D}(z_1, z_2, D)$ for $z_2 \geq z_1 > D$ by providing Algorithm 7.2, which copes with this message adversary, despite its power to partition the communication graphs and providing only a partially consecutive stability phase. The algorithm reuses some of the supporting functions from Algorithm 6.1 and

employs the same mechanisms to approximate the graph sequence and store/forward the processes' initial values.

Informally, in a round $r$, the algorithm checks if it can reliably find a unique root component $\mathsf{R}$ in round $r - D$ (Lines 17 and 18). If this is the case, the algorithm searches for the first round $\mathsf{s}$ such that this root component consecutively occurred in $(G_i)_{i=\mathsf{s}}^{r}$ and locks the maximum proposal value $\mathsf{x}_q^{\mathsf{s}}$ a process $q \in \mathsf{R}$ held at time $\mathsf{s}$ (Lines 19 to 21). This ensures that if a unique root component occurs in a subsequence of at least $D + 1$ rounds, every process has the same lock value, forever after.

The criterion for a decision is straightforward in Algorithm 7.2: a process $p$ decides in Line 33 as soon as it detects a $\mathsf{R}'$-rooted subsequence of duration at least $z_2$ (Lines 28 to 31). Compared to our previous approaches, it is interesting to note that fixing the third parameter of $\Diamond\mathsf{STABLE}_{\leqslant N,D}(z_1, z_2, D)$ to $D$ here ensures that every process eventually must detect this subsequence, thereby ensuring termination. Again, the algorithm decides on the largest value of $\mathsf{x}_q^{\mathsf{t}}$ of $q \in \mathsf{R}'$, where $\mathsf{t}$ is the earliest round such that $\mathsf{R}'$ appeared as a root component in every communication graph $G_r$ for $r \in [\mathsf{t}, i]$ but not in $G_{\mathsf{t}-1}$ (Line 32). Since we have that $z_2 \geq z_1$, it is guaranteed that the first sequence detected in this way is locked by all processes forever after, according to Corollary 7.3.2 and the mechanism described in the previous paragraph. It follows that all future decisions are the same.

In the sequel, we prove the correctness of Algorithm 7.2 under the message adversary $\Diamond\mathsf{STABLE}_{\leqslant N,D}(z_1, z_2, D)$ if $z_2 > D$ and $z_2 \geq z1$. Before we begin, we introduce two lemmas that assert the correctness of the function `allRoots` from Algorithm 7.2. As the code of `searchRoot` and `allRoots` is almost the same, the proof of these lemmas is essentially the same as the proof of Lemmas 6.4.4 and 6.4.5 and we present them here without proof.

**Lemma 7.5.1.** $\forall r > s\ \forall p \in \Pi_\sigma$: *if* $R = \texttt{allRoots}_p^r(s) \neq \emptyset$, *then*

*(1) Every* $k \in R$ *is a source component*[1] *of* $(G_s)$, *i.e.,* $R \subseteq \mathrm{Sources}(G_s)$.

*(2)* $\forall k \in R\ \forall q \in k$: $q \in \mathrm{CP}_p^r(s)$, *and*

*(3) if* $\exists t \leqslant s$: $R \subseteq \mathrm{Sources}(G_t)$ *then* $R \subseteq \texttt{allRoots}_p^r(t)$.

**Lemma 7.5.2.** *Pick a* $\sigma \in \Diamond\mathsf{STABLE}_{\leqslant N,D}(z_1, z_2, D)$, *fix a round* $s$ *and let* $r > s$. *If any of the following holds:*

*(1)* $\{R\} = \mathrm{Sources}(G_s)$ *and* $R \subseteq \mathrm{CP}_p^r(s)$, *or*

*(2)* $(G_i)_{i=s}^{s+D}$ *is* $R$*-rooted and* $r \geqslant s + D$,

[1] Recall from Section 3.2 that a source component of $G$ is a connected component of $G$ that has no edges from outside of the component. If there is only one source component, we call this unique source component the root component of $G$.

**Algorithm 7.2:** Consensus algorithm for $\Diamond\mathsf{STABLE}_{\leqslant N,D}(z_1, z_2, D)$, code for process $p$. Uses function definitions from Algorithm 6.1.

```
1  Function allRoots(s):
2      R ← ∅
3      V ← {v ∈ P | ∃(s, ∗, v) ∈ A or ∃(s, v, ∗) ∈ A}
4      E ← {(u, v) ∈ P² | ∃(s, u, v) ∈ A}
5      Let SCC(V, E) denote the set of vertex sets of the strongly connected components
       (SCCs) of ⟨V, E⟩. A single node q may constitute a SCC only if (q, q) ∈ E.
6      foreach C ∈ SCC(V, E) do
7          if ∄v ∈ V \ C: (v, u) ∈ E for a u ∈ C then
8              R ← R ∪ {C}
9      return R
   Initialization:
10 r ← 0, x ← x_p, ℓ ← 0, A ← ∅, P ← ∅
11 S ← {(p, r, x, ℓ)}
12 r ← 1
   Round r communication:
13 Attempt to send (P, S, A) to all
14 Receive m_q from all q with (q, p) ∈ G_r
   Round r computation:
15 foreach m_q s.t. p received m_q = (P_q, S_q, A_q) in round r do
16     update(q, P_q, S_q, A_q)
17 R ← allRoots(r − D)
18 if |R| = 1 then
19     s ← r − D
20     while s > 1 and R ⊆ allRoots(s − 1) do s ← s − 1
21     x ← max{X(q, s) | q ∈ k ∧ k ∈ R}
22 R′ ← ∅
23 i ← 0
24 while y_p = ⊥ and i < r − 1 do
25     i ← i + 1
26     if |allRoots(i)| ≠ 1 then R′ ← ∅
27     else // |allRoots(i)| = 1
28         if allRoots(i) ≠ R′ then
29             R′ ← allRoots(i)
30             t ← i
31         if i − t + 1 ≥ z_2 then
32             while t > 1 and R′ ⊆ allRoots(t − 1) do t ← t − 1
33             y_p ← max{X(q, t) | q ∈ k ∧ k ∈ R′}
34 S ← S ∪ (p, r, x, ℓ)
35 r ← r + 1
```

*then* $\texttt{allRoots}_p^r(s) = \{R\}$ *for every process* $p$.

**Lemma 7.5.3.** *Algorithm 7.2 satisfies validity.*

*Proof.* Since the decision is always on $\mathtt{X}(q, \mathsf{t})$ by Line 33, validity follows from Lemma 6.4.3: Clearly, when executing Line 33, $\mathsf{R}' \neq \emptyset$. From $\mathsf{R}' \subseteq \mathtt{searchRoot}(\mathsf{t})$, it therefore follows that $q \in \mathrm{CP}_p^r(\mathsf{t})$ by Lemma 7.5.1 because $q \in k \wedge k \in \mathsf{R}'$. □

**Lemma 7.5.4.** *Algorithm 7.2 satisfies agreement.*

*Proof sketch.* We show that, if the first decision that happens in a given execution is by process $p$ in round $\tau$ on value $v$, then, if a process $p'$ decides in a round $\tau' \geq \tau$, this decision is on $v$ as well.

The code of the algorithm implies that a decision by $p$ in round $\tau$ can only happen when $p$ detected a subsequence $(G_r)_{r=a}^{b}$ of duration at least $b - a + 1 \geq z_2$ that contained exactly one vertex-stable root component $\mathsf{R}'$. Furthermore, it follows from the code that $v = \max(\mathsf{x}_q^{\mathsf{t}} | q \in \mathsf{R}')$, i.e., the largest proposal value of any member of $\mathsf{R}'$. Here, round $\mathsf{t}$ is such that $\mathsf{R}'$ is a root component of $G_k$ for all rounds $k \in [\mathsf{t}, b]$ but not of $G_{\mathsf{t}-1}$.

Let $\sigma$ denote the longest graph sequence in the current execution such that $\sigma \supseteq (G_r)_{r=a}^{b}$ and $\mathsf{R}$ is a root component of every $G_r \in \sigma$. Since $z_2 \geqslant z_1 > D$ by assumption, according to Definition 7.3.1, $\mathsf{STICKY}_n(z_1)$ ensures that there is a $\mathsf{R}$-rooted sequence $(G_r)_{r=a'}^{b'} \subseteq \sigma$ with $b' - a' + 1 > D$. As the first decision of the current execution is based on a subsequence of $\sigma$, for all $S \subseteq \Pi$, it follows that all $S$-rooted subsequences that are different from $\sigma$ and occur before $a'$, have a duration of less than $z_2$ due to Corollary 7.3.2. Pick an arbitrary $p' \in \Pi$. It follows from our previous argument that, if $p'$ already decided before time $a'$, it decided the same as $p$. If $p'$ has not yet decided before time $a'$, in round $a' + D$, it enters Line 21 and sets $\mathsf{x}_{p'} \leftarrow v$. A simple induction shows that in fact $\mathsf{x}_{p'}^{k} = v$ for all times $k \in [a' + D, b' + D]$ and, even more, as $b' - a' + 1 > D$, we have $\mathsf{x}_{p'}^{k} = v$ for all times $k \geq a' + D$. This shows our claim since any future decision made by $p'$ is either on $\max(\mathsf{x}_q^{\mathsf{t}} | q \in \mathsf{R}') = v$ or on $\mathsf{x}_{p''}^{k} = v$ for a $p'' \in \Pi$ and a time $k \geq a' + D$. □

**Lemma 7.5.5.** *Algorithm 7.2 terminates.*

*Proof sketch.* Let $\sigma'$ be the $R$-rooted subsequence with duration $y$ from Definition 7.4.1, as guaranteed by the message adversary. Furthermore, let $\tau$ be the round where the subsequence $\sigma''$, as introduced in Definition 7.4.1, ends. We can apply Definition 7.4.1 and Lemma 7.5.2 to show that, at the latest in round $\tau$, every process finds $\sigma'$ and thus decides via Line 33. □

Taken together, the previous three lemmas can be combined to formulate the correctness of Algorithm 7.2.

**Theorem 7.5.6.** *Algorithm 7.2 solves consensus under* $\Diamond\mathsf{STABLE}_{\leqslant N,D}(z_1, z_2, D)$ *if* $z_2 > D$ *and* $z_2 \geq z_1$.

Taken together with our previous possibility and impossibility results, this enables a characterization of the consensus solvability of $\Diamond\mathsf{STABLE}_{\leqslant N,D}(x, y, D)$ for all values of $x, y > 0$, which is shown in Figure 7.2.

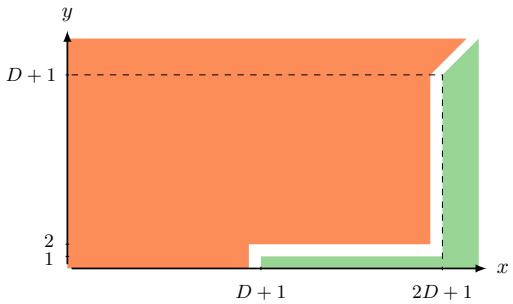


Figure 7.2: Characterization of the consensus solvability/impossibility border of $\Diamond\mathsf{STABLE}_{\leqslant N,D}(x, y, z)$, where $z = 0$ if $y \leq D + 1$ and $z = D$ otherwise: red (upper left) = impossible, green (right) = possible.

## 7.6 Lower Bounds for $k$-Set Agreement

So far in this chapter, we have seen that a partitioning of the communication graphs into multiple connected components does still allow us to solve consensus under certain circumstances. Preceding this, we have seen in Chapter 6 that communication patterns consisting of rooted communication graphs allow us to solve consensus, given enough stability. A natural question to ask here is: does this mean that we can solve $k$-set agreement if the communication graphs may partition into at most $k$ connected components, each containing a rooted spanning tree? Clearly, with a static partitioning that remains the same throughout the communication pattern and sufficient stability in each component, this is the case, as we can deploy the consensus algorithm from Algorithm 6.2 in each partition. Perhaps not too surprisingly, if the stability is not sufficient for consensus in one component, $k$-set agreement remains impossible; we will prove this formally in Theorem 7.6.3. Interestingly though, if the only restriction on the message adversary is such that each communication graph may consist of up to $k$ components, the partitioning becomes fixed eventually, and each component contains a vertex-stable source component eventually that lasts forever, $k$-set agreement is still impossible. We will show this result in Theorem 7.6.5 and provide an even stronger version, where $k$-set agreement is impossible despite significantly less than $k$ components being allowed in every round in Theorem 7.6.6.

For the purpose of deriving these impossibility results, Definition 7.6.1 below defines the message adversary $\Diamond\mathsf{STABLE}_n(k, d)$, which allows at most $k$ source components per round and guarantees a common window of vertex stability of duration at least $d$.

**Definition 7.6.1** (Message adversary $\Diamond\mathsf{STABLE}_n(k, d)$)**.** *For $k > 0$, $d > 0$, the message adversary $\Diamond\mathsf{STABLE}_n(k, d)$ is the set of all sequences of communication graphs $(G_r)_{r>0}$ on $n$ nodes, where*

*(i) for every round $r$, $G_r$ contains at most $k$ source components,*

*(ii) for each $(G_r)_{r>0}$, there exists some $r_{stab} > 0$ and an interval of rounds $J = [r_{stab}, r_{stab} + d - 1]$ with a set of $1 \leqslant \ell \leqslant k$ $J$-vertex-stable source components $S_1, \ldots, S_\ell$. Furthermore, each source component that occurs in a communication graph $G_r$ with $r \in J$ is in this set.*

Note that the message adversary $\Diamond\mathsf{STABLE}_n(k, 1)$ guarantees at most $k$ vertex-stable source components in every $G_r$, $r > 0$.

We will now prove that it is impossible to solve $k$-set agreement for $1 \leqslant k < n - 1$ under the message adversary $\Diamond\mathsf{STABLE}_n(k, n - 1)$, even under the slightly weaker version of this message adversary stated in Theorem 7.6.3 below. We will use the generic impossibility theorem provided in [BRS11, Thm. 1] for this purpose. In a nutshell, the latter exploits the fact that $k$-set agreement is impossible if $k$ sufficiently disconnected components may occur and consensus cannot be solved in some component.

We first introduce the required definitions: Two executions of an algorithm $\alpha, \beta$ are *indistinguishable until decision* for a set of processes $\mathcal{D}$, denoted $\alpha \sim_{\mathcal{D}} \beta$, if for any $p \in \mathcal{D}$ it holds that $p$ executes the same state transitions in $\alpha$ and in $\beta$ until it decides. Now consider a model of a distributed system $\mathcal{M} = \langle \Pi \rangle$ that consists of the set of processes $\Pi$ and a *restricted model* $\mathcal{M}' = \langle \mathcal{D} \rangle$ that is computationally compatible to $\mathcal{M}$ (i.e., an algorithm designed for a process in $\mathcal{M}$ can be executed on a process in $\mathcal{M}'$) and consists of the set of processes $\mathcal{D} \subseteq \Pi$. Let $\mathcal{A}$ be an algorithm that works in system $\mathcal{M} = \langle \Pi \rangle$, where $\mathcal{M}_{\mathcal{A}}$ denotes the set of runs of algorithm $\mathcal{A}$ on $\mathcal{M}$, and let $\mathcal{D} \subseteq \Pi$ be a nonempty set of processes. Given any restricted system $\mathcal{M}' = \langle \mathcal{D} \rangle$, the *restricted algorithm* $A_{|\mathcal{D}}$ for system $\mathcal{M}'$ is constructed by dropping all messages sent to processes outside $\mathcal{D}$ in the message sending function of $\mathcal{A}$. We also need the following similarity relation between runs in computationally compatible systems (cf. [BRS11, Definition 3]): Let $\mathcal{R}$ and $\mathcal{R}'$ be sets of runs, and $\mathcal{D}$ be a non-empty set of processes. We say that *runs $\mathcal{R}'$ are compatible with runs $\mathcal{R}$ for processes in $\mathcal{D}$*, denoted by $\mathcal{R}' \preccurlyeq_{\mathcal{D}} \mathcal{R}$, if $\forall \alpha \in \mathcal{R}'\ \exists \beta \in \mathcal{R} \colon \alpha \sim_{\mathcal{D}} \beta$.

**Theorem 7.6.2** ($k$-Set Agreement Impossibility [BRS11, Thm. 1])**.** *Let $\mathcal{M} = \langle \Pi \rangle$ be a system model and consider the runs $\mathcal{M}_A$ that are generated by some fixed algorithm $A$ in $\mathcal{M}$, where every process starts with a distinct input value. Fix some nonempty and pairwise disjoint sets of processes $\mathcal{D}_1, \ldots, \mathcal{D}_{k-1}$, and a set of distinct decision values $\{v_1, \ldots, v_{k-1}\}$. Moreover, let $\mathcal{D} = \bigcup_{1 \leq h < k} \mathcal{D}_h$ and $\overline{\mathcal{D}} = \Pi \setminus \mathcal{D}$. Consider the following two properties:*

***(dec-$\mathcal{D}$)*** *For every set $\mathcal{D}_h$, value $v_h$ was proposed by some $p \in \mathcal{D}$, and there is some $q \in \mathcal{D}_h$ that decides $v_h$.*

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

*Let $\mathcal{R}_{(\overline{\mathcal{D}})} \subseteq \mathcal{M}_A$ and $\mathcal{R}_{(\mathcal{D},\overline{\mathcal{D}})} \subseteq \mathcal{M}_A$ be the sets of runs of $A$ where (dec-$\overline{\mathcal{D}}$) respectively both, (dec-$\mathcal{D}$) and (dec-$\overline{\mathcal{D}}$), hold.*[2] *Suppose that the following conditions are satisfied:*

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

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

*In addition, consider a restricted model $\mathcal{M}' = \langle\overline{\mathcal{D}}\rangle$ such that the following properties hold:*

**(C)** *There is no algorithm that solves consensus in $\mathcal{M}'$.*

**(D)** $\mathsf{MA}' \preccurlyeq_{\overline{\mathcal{D}}} \mathcal{M}_A$.

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

The proof of Theorem 7.6.3 below utilizes Theorem 7.6.2 in conjunction with the impossibility of consensus under $\Diamond\mathsf{STABLE}_{\leqslant N,D}(D)$ established in Theorem 6.3.6.

**Theorem 7.6.3** (Impossibility of $k$-set agreement under $\Diamond\mathsf{STABLE}_n(k, n-k-1)$)**.** *There is no algorithm that solves $k$-set agreement with $n > k+1$ processes under the message adversary $\Diamond\mathsf{STABLE}_n(k, n-k-1)$, for any $1 \leqslant k < n-1$, even if there are $k-1$ source components $S_1, \ldots, S_{k-1}$ that are vertex-stable all the time (and only source component $S_k$ is vertex-stable for at most $n-k-1$ rounds).*

*Proof.* Suppose that there is a $k$-set algorithm $\mathcal{A}$ that works correctly under the assumptions of our theorem. The case $k = 1$ follows directly from Theorem 6.3.6.

To prove the theorem for $k > 1$, we will show that the conditions of the generic Theorem 7.6.2 are satisfied, thereby providing a contradiction to the assumption that $\mathcal{A}$ exists. Let $\mathcal{D}_i = \{p_i\}$ for $0 < i \leqslant k-1$ and let $\mathcal{D} = \bigcup_{i=1}^{k-1} \mathcal{D}_i$. Consequently, $\overline{\mathcal{D}} = \{p_k, p_{k+1}, \ldots, p_n\}$ and $|\overline{\mathcal{D}}| \geqslant 2$.

**(A)** The set of runs $\mathcal{R}_{(\overline{\mathcal{D}})}$ of $\mathcal{A}$ where no process in $\overline{\mathcal{D}}$ receives any message from $\mathcal{D}$ before it decides is nonempty: We choose the communication graph in every round to be such that $\overline{\mathcal{D}}$ has no incoming links from $\mathcal{D}$ until every process in $\overline{\mathcal{D}}$ has decided. Since any such sequence of communication graphs satisfies the assumptions of our theorem, $\mathcal{R}_{(\overline{\mathcal{D}})} \neq \emptyset$.

**(B)** The set of runs $\mathcal{R}_{(\mathcal{D},\overline{\mathcal{D}})}$ of $\mathcal{A}$ where both (i) some process in every $D_i$ decides $v_i$ and (ii) no process in $\overline{\mathcal{D}}$ receives any message from $\mathcal{D}$ before it decides satisfies $\mathcal{R}_{(\overline{\mathcal{D}})} \preccurlyeq_{\overline{\mathcal{D}}} \mathcal{R}_{(\mathcal{D},\overline{\mathcal{D}})}$: Let $\mathcal{H}$ be the set of runs where processes $p_i$ have unique input values $x_i = i$, $0 < i < k$, the communication graph in every round is such that $p_1, \ldots, p_{k-1}$ are isolated, and $p_k, \ldots, p_n$ are weakly connected (with a single source component) until every process has decided. By the assumptions of our theorem, $\mathcal{H}$ is non-empty. Since (i) the processes in $\overline{\mathcal{D}}$ never

[2] Note that $\mathcal{R}_{(\overline{\mathcal{D}})}$ is by definition compatible with the runs of the restricted algorithm $A_{|\overline{\mathcal{D}}}$.

receive a message from a process in $\mathcal{D}$ in both $\mathcal{R}_{(\overline{\mathcal{D}})}$ and $\mathcal{H}$, and (ii) the initial values of the processes in $\overline{\mathcal{D}}$ are not restricted in $\mathcal{H}$ in any way, it is easy to find, for any run $\rho \in \mathcal{R}_{(\overline{\mathcal{D}})}$, a run $\rho' \in \mathcal{H}$ such that $\rho \sim_{\overline{\mathcal{D}}} \rho'$. Because obviously $\mathcal{H} \subseteq \mathcal{R}_{(\mathcal{D},\overline{\mathcal{D}})}$, we have established $\mathcal{R}_{(\overline{\mathcal{D}})} \preccurlyeq_{\overline{\mathcal{D}}} \mathcal{R}_{(\mathcal{D},\overline{\mathcal{D}})}$.

**(C)** Consensus is impossible in $\mathcal{M}' = \langle\overline{\mathcal{D}}\rangle$: Let $\overline{\mathcal{D}}$ be the partition containing the $k^{\text{th}}$ source component $S_k$, which is perpetually changing in every round, except for some interval of rounds $I = [r_{\text{stab}}, r_{\text{stab}} + \ell - 1]$, where $\ell = n - k$, for some fixed $r_{\text{stab}}$.

Since $|\overline{\mathcal{D}}| = n - k + 1$, every communication pattern generated by the message adversary $\Diamond\mathsf{STABLE}_{|\overline{\mathcal{D}}|,\ell}(\ell)$, which is even stronger than the corresponding message adversary underlying Theorem 6.3.5, satisfies this criterion. Hence, consensus is impossible in $\overline{\mathcal{D}}$.

**(D)** $\mathcal{M}'_{\mathcal{A}_{|\overline{\mathcal{D}}}} \preccurlyeq_{\overline{\mathcal{D}}} \mathcal{M}_{\mathcal{A}}$: Fix any run $\rho' \in \mathcal{M}'_{\mathcal{A}_{|\overline{\mathcal{D}}}}$ and consider a run $\rho \in \mathcal{M}_{\mathcal{A}}$, where every process in $\overline{\mathcal{D}}$ has the same sequence of state transitions in $\rho$ as in $\rho'$. Such a run $\rho$ exists, since the processes in $\overline{\mathcal{D}}$ can be disconnected from $\mathcal{D}$ in every round in $\mathcal{M}_{\mathcal{A}}$, so $\rho \sim_{\overline{\mathcal{D}}} \rho'$. □

Since Theorem 7.6.3 tells us that no $k$-set agreement algorithm (for $1 \leqslant k < n - 1$) can *terminate* with insufficient concurrent stability of the at most $k$ source components in the system, it is tempting to assume that $k$-set agreement becomes solvable if a round exists after which all communication graphs remain the same. However, we will prove in Theorem 7.6.5 below that this is not the case for any $1 < k \leqslant n - 1$. We will again use the generic Theorem 7.6.2, this time in conjunction with the variant of the well-known impossibility of consensus with lossy links [SW89, SWK09] provided in Lemma 7.6.4, to prove that ensuring at most $k$ different decision values is impossible here, as too many decision values may originate from the unstable period.

**Lemma 7.6.4.** *Let $\mathcal{M}' = \langle p, q\rangle$ be a two-processor subsystem of our system $\mathcal{M} = \langle\Pi\rangle$. If the sequence of communication graphs $G_r$, $r > 0$, of $\mathcal{M}$ are restricted by the existence of a round $r' > 0$ such that (i) for $r < r'$, $(p, q) \in G_r$ or $(q, p) \in G_r$, and no other edges incident with $p$ or $q$ are in $G_r$, and (ii) for $r \geqslant r'$, there are no edges incident with $p$ and $q$ at all in $G_r$, then consensus is impossible in $\mathcal{M}'$.*

*Proof.* Up to $r'$, this is ensured by the impossibility of 2-processor consensus with a lossy but at least unidirectional link established in [SWK09, Lemma 3]. After $r'$, this result continues to hold (and is even ensured by the classic lossy link impossibility [SW89]). Hence, consensus is indeed impossible in $\mathcal{M}'$. □

**Theorem 7.6.5.** *There is no algorithm that solves $k$-set agreement for $n \geqslant k+1$ processes under the message adversary $\Diamond\mathsf{STABLE}_n(k, \infty)$, for every $1 < k < n$.*

*Proof.* Suppose again that there is a $k$-set algorithm $\mathcal{A}$ that works correctly under the assumptions of our theorem. We restrict our attention to runs of $\mathcal{M}_{\mathcal{A}}$ where, until $r_{\text{stab}}$,

(i) the same set of $k-1$ source components $\{\mathcal{D}_1, \dots, \mathcal{D}_{k-1}\}$ with $\mathcal{D} = \bigcup_{i=1}^{k-1} \mathcal{D}_i$ exists in every round, and (ii) two remaining processes $\overline{\mathcal{D}} = \Pi \setminus \mathcal{D} = \{p_1, p_2\}$ exist, which are (possibly only uni-directionally, i.e., via a lossy link) connected in every round, without additional edges to or from $\mathcal{D}$. After $r_{\text{stab}}$, the communication graph remains the same, except that the processes in $\overline{\mathcal{D}}$ are disconnected from each other and there is an edge from, say, $p_1$ to some process in $\mathcal{D}$ in every round. Note that these runs satisfy Definition 7.6.1 for $d = \infty$, as the number of source components never exceeds $k$.

Moreover, we let the adversary choose $r_{\text{stab}}$ sufficiently large such that the processes in $\mathcal{D}$ have decided. Since the processes in $\mathcal{D}_i$ $(i < 0 < k)$ never receive a message from the remaining system before $r_{\text{stab}}$, in which case they must eventually unilaterally decide, we can safely assume this.

We can now again employ the generic impossibility Theorem 7.6.2 in this modified setting. The proofs of properties (A), (B) and (D) remain essentially the same as in Theorem 7.6.3. It hence only remains to prove:

**(C)** Consensus is impossible in $\mathcal{M}' = \langle \overline{\mathcal{D}} \rangle$: This follows immediately from Lemma 7.6.4 with $r' = r_{\text{stab}}$.

□

The following Theorem 7.6.6 reveals that even (considerably) less than $k$ source components per round before stabilization and a single perpetually stable source component after stabilization are not sufficient for solving $k$-set agreement.

**Theorem 7.6.6.** *There is no algorithm that solves $k$-set agreement for $n \geqslant k+1$ processes under the message adversary $\Diamond\mathsf{STABLE}_n(\lceil k/2 \rceil + 1, \infty)$, for every $1 < k < n$, even if $G_r = G$, $r \geqslant r_{stab}$, where $G$ contains only a single source component.*

*Proof.* We show that, under the assumption that $\mathcal{A}$ exists, there is a sequence of communication graphs that is feasible for our message adversary that leads to a contradiction. We choose $x_i = i$ for all $p_i \in \Pi$ and let $\mathcal{D}_i = \{p_{1+2i}, p_{2+2i}\}$ for $0 \leqslant i < \lceil k/2 \rceil - 1$. If $k$ is even, let $\mathcal{D}_{k/2-1} = \{p_{k-1}, p_k\}$; if $k$ is odd, let $\mathcal{D}_{\lceil k/2-1 \rceil} = \{p_k\}$. In any case, let $\mathcal{D}_{\lceil k/2 \rceil} = \{p_{k+1}\}$. Finally, let $\overline{\mathcal{D}} = \{p_{k+2}, \dots, p_n\}$. Note that $\overline{\mathcal{D}}$ may be empty, while all $\mathcal{D}_i$ are guaranteed to contain at least one process since $n > k$. For all rounds, the processes in $\overline{\mathcal{D}}$ have an incoming edge from a process in one of the $\mathcal{D}_i$.

We split the description of the adversarial strategy into $\lceil k/2 \rceil + 1$ phases in each of which we will force some $\mathcal{D}_i$ to take $|\mathcal{D}_i|$ decisions. To keep processes $p_{1+2i}, p_{2+2i} \in \mathcal{D}_i$ with $|\mathcal{D}_i| = 2$ from deciding on the same value before their respective phase $i$, the adversary restricts $G_r$ such that (i) there are no links to $\mathcal{D}_i$ from any other $\mathcal{D}_j$ and (ii) either the edge $(p_{1+2i} \to p_{2+2i})$ or $(p_{1+2i} \leftarrow p_{2+2i})$ or both are in $G_r$, in a way that causes Lemma 7.6.4 to apply. Note carefully that any such $G_r$ indeed has no more than $\lceil k/2 \rceil + 1$ source components.

In the initial phase, $\mathcal{D}_{\lceil k/2 \rceil}$ is forced to decide: Since $p_{k+1}$ has no incoming edges from another node in $G_r$, this situation is indistinguishable from a run where $p_{k+1}$ became the single source component after $r_{\text{stab}}$. Thus, by the correctness of $\mathcal{A}$, $p_{k+1}$ must eventually decide on $x_{k+1} = k+1$. At this point, the initial phase ends, and we can safely allow the adversary to modify $G_r$ in such a way that $p_{k+1}$ has an incoming edge from some other process.

We now proceed with $\lceil k/2 \rceil - 1$ phases: In the $i^{\text{th}}$ phase, $0 \leqslant i < \lceil k/2 \rceil - 1$, the adversary drops any link between the processes $p_{1+2i}, p_{2+2i} \in \mathcal{D}_i$ (and does not provide an incoming link from any other process, as before) in any $G_r$. Since, for both processes this is again indistinguishable from the situation where they become the single source component after $r_{\text{stab}}$, both will eventually decide in some future round (if they have not already decided). Since the adversary may have chosen a link failure pattern in earlier phases that causes the impossibility (= forever bivalent run) of Lemma 7.6.4 to apply, as $\mathcal{M}'_{\mathcal{A}_{|\mathcal{D}_i}} \preccurlyeq_{\mathcal{D}_i} \mathcal{M}_{\mathcal{A}}$, it follows that $\mathcal{A}$ and hence $\mathcal{A}_{|\mathcal{D}_i}$ cannot have solved consensus in $\mathcal{D}_i$. Since $\mathcal{A}$ solves $k$-set agreement, $p_{1+2i}$ and $p_{2+2i}$ must hence decide on two *different* values. Moreover, since neither $p_{1+2i}$ nor $p_{2+2i}$ ever received a message from a process not in $\mathcal{D}_i$, their decision values must be different from the ones in all former phases.

Finally, after $p_{1+2i}$ and $p_{2+2i}$ have made their decisions, the adversary may again modify $G_r$ such that they have an incoming edge from some other process, thereby reducing the number of source components by two and preserving the maximum number $\lceil k/2 \rceil + 1$ of source components, and continue with the next phase.

If $k$ is even, then the final phase $\lceil k/2 \rceil - 1$ forces two more decisions just as described above; otherwise, $p_k$ provides one additional decision value (which happens concurrently with the initial phase here). In either case, we have shown that all $p_i$ with $1 \leq i \leq k+1$ have decided on different values, which contradicts the assumption that a correct algorithm $\mathcal{A}$ exists. □

Note that Theorem 7.6.6 reveals an interesting gap between 2-set agreement and 1-set agreement, i.e., consensus: It shows that 2-set agreement is impossible with $\lceil k/2 \rceil + 1 = 2$ source components per round before and a single fixed source component after stabilization. By contrast, if we reduce the number of source components per round to a single one before stabilization (and still consider a single fixed source component thereafter), even 1-set agreement becomes solvable [BRS12].

Chapter 8

# A Faster and Simpler Consensus Algorithm

So far in our study of characterizations of consensus solvability under a message adversary, our focus was on the mere existence of solution algorithms with little regard for the simplicity or efficiency of the algorithms. Thus, two points of interest remain:
Firstly, the consensus algorithms described so far are rather long, use complex algorithmic techniques and their correctness proofs often span several pages. To make these results more accessible and simplify their validation, we believe that it would be desirable to have easier solution algorithms.
Secondly, the question of optimality wrt. termination time still remains open for the algorithm described in Section 6.4. This makes sense, since the main concern of this section was to find out whether consensus was achievable at all. Still, it would be interesting if the quite significant termination times (quadratic in the number of processes for Algorithm 6.2) can be reduced.

In this section, we attempt to answer both of the above questions. We provide an easy, generic consensus algorithm that relies on a function $\Delta(.)$ that provides a deterministic, iterative labelling of communication patterns. We then show how this function can be implemented under the message adversary of Section 6.2 using a short, centralized algorithm. Finally, we show that our new algorithm is asymptotically optimal with respect to termination time by providing a matching lower bound.

## 8.1 Related Work

There are two related works that we would like to highlight for this section. The first, [WSS19], which introduces a consensus algorithm for the message adversary $\Diamond\mathsf{STABLE}_{n,D}$, was already presented in Chapter 6. The second is [FG11], where consensus solvability is characterized under a general message adversary with $n = 2$ processes. We will describe some of the main points of this result in Section 8.2.3.

## 8.2 An Easy Consensus Algorithm

### 8.2.1 The function Δ

Our algorithm will rely on a labelling function $\Delta(.)$, which is a member of the class of $\Delta$-functions, introduced in the following Definition 8.2.1. Fixing some admissible communication pattern $\sigma$ of a message adversary, $\Delta(\sigma|_r)$ assigns a label to every round $r$ prefix $\sigma|_r$ of $\sigma$ as follows: Going from shorter prefixes to longer ones, the label starts to be $\emptyset$ until it becomes a fixed, non-empty subset of the kernel of $\sigma$.

**Definition 8.2.1.** *Let MA be a message adversary and $\sigma \in$ MA be an arbitrary admissible communication pattern. The class of $\Delta$-functions contains those functions*

$$\Delta : \bigcup_{r\geq 1} \{\sigma|_r : \sigma \in \mathsf{MA}\} \to 2^{\Pi}$$

*for which there is a round $s$ (which may depend on $\sigma$) and a set $K \subseteq \mathrm{Ker}(\sigma)$ with $K \neq \emptyset$ such that*

$$\Delta(\sigma|_r) = \begin{cases} \emptyset, & \text{for } r \leq s \\ K & \text{for } r > s \end{cases}$$

As we show in Theorem 8.2.2, already the solvability of binary consensus implies the existence of a $\Delta$-function, thus its existence can be derived from the solvability of consensus for an arbitrary set of input values.

**Theorem 8.2.2.** *If binary consensus is solvable under MA, then there exists a function $\Delta(.)$ such that $\Delta(.)$ is a member of the class of $\Delta$-functions for MA.*

*Proof.* First, we show that if binary consensus is solvable under MA then every $\sigma \in \mathsf{MA}$ satisfies $\mathrm{Ker}(\sigma) \neq \emptyset$. Suppose that some algorithm $A$ solves consensus in a communication pattern $\sigma$ in spite of $\mathrm{Ker}(\sigma) = \emptyset$. By termination, for every input configuration $C$, there is a round $t$ such that all processes have decided when executing $A$ in $\langle C, \sigma|_t\rangle$. In fact, since there are only finitely many processes and hence input assignments, there exists a round $t'$ such that all processes have decided when running $A$ in $\langle C, \sigma|_{t'}\rangle$, irrespective of the input value assignment of the initial configuration $C$. Assuming that the set of input values is $\mathcal{I} = \{0, 1\}$, by validity, when all processes start with input 0, they must also decide 0, and when all processes start with 1, they must decide 1. An easy induction shows that therefore, by agreement, there must exist two input configurations $C', C''$ that differ in the input value assignment of a single process $p$ but all processes decide 0 in $\langle C', \sigma|_{t'}\rangle$, whereas all processes decide 1 in $\langle C'', \sigma|_{t'}\rangle$. Since $\mathrm{Ker}(\sigma|_{t'}) = \emptyset$ by assumption, there is a process $q$ such that $p \notin \mathrm{HO}_{\sigma|_{t'}}(q)$. But then $\langle C', \sigma|_{t'}\rangle \sim_q \langle C'', \sigma|_{t'}\rangle$ and $q$ decides the same in both executions, a contradiction.

We conclude the proof by showing that the existence of $\Delta(.)$ follows if every $\sigma \in \mathsf{MA}$ has $\mathrm{Ker}(\sigma) \neq \emptyset$, which implies that for every $\sigma \in \mathsf{MA}$ there is a round $r$ such that

$\mathrm{Ker}(\sigma|_r) \neq \emptyset$. Given $\sigma \in \mathsf{MA}$, let $s$ be the smallest such round where $K = \mathrm{Ker}(\sigma|_s) \neq \emptyset$. Then

$$\Delta(\sigma|_r) = \begin{cases} \emptyset, & \text{for } r \leq s \\ K & \text{for } r > s \end{cases}$$

is a $\Delta$-function according to Definition 8.2.1: Because $\mathrm{Ker}(\sigma|_s) = K \neq \emptyset$ and the Kernel of a communication pattern prefix monotonically increases with increasing prefix length, i.e., if $r \leq r'$ then $\mathrm{Ker}(\sigma|_r) \subseteq \mathrm{Ker}(\sigma|_{r'})$, every $\rho \in \mathsf{MA}$ with $\rho|_s = \sigma|_s$ satisfies $K \subseteq \mathrm{Ker}(\rho)$. □

One might argue that the core statement of Theorem 8.2.2 is not very strong, as the definition of $\Delta(.)$ is almost equivalent to the consensus problem specification itself. However, the point is not that this formulation is very deep, but rather that it is very useful, especially because of its simplicity. We show in the remaining chapter how, by means of encapsulating the computation of $\Delta(.)$, consensus can be solved rapidly and using a relatively simple algorithm under a message adversary for which, until now, only a slow and complex algorithms existed.

### 8.2.2 Solving consensus with $\Delta$

From the contraposition of Theorem 8.2.2, it immediately follows that the existence of such a $\Delta$-function $\Delta(.)$ is necessary to solve consensus. At first glance, it might seem that the existence of $\Delta(.)$ is also sufficient for consensus, i.e., that Theorem 8.2.2 can actually be extended to an equivalence, thus providing an efficient consensus solvability characterization. The reason why this is not the case, however, is that local uncertainty, i.e., the incomplete view a process has of the communication pattern, makes processes unsure of the actual communication pattern. Thus the existence of $\Delta(\sigma|_r)$ is not enough since a process may be uncertain of $\sigma|_r$. In order to make consensus solvable, we introduce an additional condition, stated in Assumption 1, on a $\Delta$-function $\Delta(.)$ that we will require in addition to its existence:

**Assumption 1.** $\forall \sigma \in \mathit{MA}\ \exists r \in \mathbb{N} : \sigma' \sim \sigma|_r \Rightarrow \Delta(\sigma') = \Delta(\sigma|_r) \neq \emptyset$ .

Assumption 1 states that every admissible communication pattern has a round after which the values of $\Delta(.)$ are the same for every indistinguishable prefix. While this is too strong to be truly necessary, it is still very useful because it is readily satisfied by a variety of different message adversaries from the literature that have a rather difficult algorithmic implementation. We can see that the predicate from Assumption 1 is sufficient by means of the following Algorithm 8.1.

The principal operation of Algorithm 8.1 is as follows: Each process $p$ waits until it detects that all prefixes $\rho$ that it still considers possible, i.e., those prefixes that do not contradict what $p$ observed so far, have the same non-empty value for $\Delta(\rho) = K$. When this occurs, $p$ waits until it hears from the process with the largest identifier in $K$ and decides on its input value.

We note that the algorithm makes use of $P$, the set of communication pattern prefixes process $p$ considers possible in a round of some execution, which it can compute from its view, given the message adversary specification. Furthermore, we make the simplifying assumption that processes know the input values of all processes that they heard from so far, which can easily be satisfied by each process attempting to broadcast its own input in every round and storing the input values (along with the corresponding process identifiers) it received so far. A book-keeping algorithm that computes $P$ more explicitly can be found in Algorithm 8.2.

**Algorithm 8.1:** Consensus algorithm, given $\Delta(.)$ for process $p$ and round $r$ when run under communication pattern $\sigma$ of message adversary $\mathsf{MA}$.

```
1 Let P = {ρ|_r : ρ ∈ MA ∧ ρ|_r ∼_p σ|_r} be the prefixes p considers possible
2 if y_p = ⊥ and ∃K ⊆ Π ∀ρ ∈ P : Δ(ρ) = K ≠ ∅ then
3 |   Let q ∈ Π be the process in K with the largest identifier
4 |   if heard from q then y_p ← x_q
```

The correctness of Algorithm 8.1 is shown in the following Theorem 8.2.3. In the proof, we use $v_p^r$ to denote the value of variable $v$ at process $p$ at time $r$.

**Theorem 8.2.3.** *Algorithm 8.1 solves consensus under a message adversary $\mathsf{MA}$, given that $\Delta(.)$ is a $\Delta$-Function for $\mathsf{MA}$ (according to Definition 8.2.1) that satisfies Assumption 1.*

*Proof.* We show that the consensus properties are satisfied by Algorithm 8.1 when it is run in an admissible communication pattern $\sigma \in \mathsf{MA}$.

Validity follows immediately, since every decision in Line 4 is on some other process' input value.

Termination follows because, when run under a communication pattern $\sigma \in \mathsf{MA}$, Assumption 1 guarantees that there is a set $K \neq \emptyset$ and a round $r$ such that $\Delta(\rho) = K$ for all $\rho$ with $\rho \sim \sigma|_r$. Furthermore, $K \subseteq \mathrm{Ker}(\sigma)$, thus the guard of Line 4 will eventually be passed.

In order to show agreement, let $r$ be the round where some process, say $p$, passes the guard in Line 2 for the first time. This implies that $K_p^r \neq \emptyset$ and $p$ has $\Delta(\rho) = K_p^r$ for all $\rho \sim_p \sigma|_r$ so, in particular, $\Delta(\sigma|_r) = K_p^r$. Definition 8.2.1 implies that for all $s \geq r$, $\Delta(\sigma|_s) = K_p^r$ as well. Thus, if some process $p'$ passes Line 2 again in round $s$, because it clearly has $\sigma|_s \in P_{p'}^s$, every decision via Line 4 must be on the input of the process with the largest identifier in $K_p^r = K_{p'}^s$. □

In Section 8.4 we will show how $\Delta(.)$ can be computed under the message adversary from Section 6.2. Whereas the computation of $\Delta(.)$ requires some effort there, we will show in Section 8.2.3 that the computation of $\Delta(.)$ is straightforward in the message adversary from [FG11].

A computation of $\Delta(.)$, together with Algorithm 8.1, gives us a solution algorithm for consensus that operates in two phases: When run in a communication pattern $\sigma$, in its round $r$ computation step, every process $p$ first pre-computes $\Delta(\rho)$ for all communication pattern prefixes $\rho$ that are compatible with its view, i.e., all $\rho \in P$, where $P$ is the set of round $r$ prefixes that $p$ considers possible, as in Line 1 of Algorithm 8.1. Process $p$ then proceeds to run Algorithm 8.1, using these pre-computed values instead of actual calls to $\Delta(.)$.

### Computing the Possible Prefixes

We now provide explicitly a short book-keeping algorithm that that suffices to compute the prefixes that process $p$ considers possible in round $r$ under a message adversary $\mathsf{MA}$ in Algorithm 8.2. It essentially computes the local view as the subgraph of the process-time graph $\mathcal{PT}$ that is induced[1] by all process-time pairs that influenced $p$. Then $p$ compares this local view to the possible round $r$ prefixes. For this, because we assume that the message adversary specification is common knowledge, we assume that $p$ knows the set of round $r$ prefixes that are allowed under $\mathsf{MA}$, which will be denoted by

$$\mathsf{MA}|_r := \{\mathcal{PT}_{\sigma|_r} : \sigma \in \mathsf{MA}\}$$

**Algorithm 8.2:** Computing $P$, the possible round $r$ prefixes, code for process $p$. We assume every process receives a message from itself in every round.

**Initialization**:

1 $V \leftarrow \{(p,0)\}$ $E \leftarrow \emptyset$ `// Sets for the` $\mathcal{PT}$`-approximation`

2 $r \leftarrow 1$

**Round $r$ communication**:

3 Attempt to broadcast $(\langle V, E\rangle, r)$

**Round $r$ computation**:

4 **foreach** $q$ *such that* $p$ *received* $(\langle V_q, E_q\rangle, r)$ *from* $q$ **do**

5 $\quad V \leftarrow V \cup V_q \cup (q, r-1)$

6 $\quad E \leftarrow E \cup E_q \cup ((q,r-1),(p,r))$

7 $P \leftarrow \{\mathcal{PT} \in \mathsf{MA}|_r : \mathcal{PT}[V] = \langle V, E\rangle\}$

8 $r \leftarrow r+1$

The correctness of Algorithm 8.2 can be formally established by showing that its code implies the following Corollary 8.2.4:

**Corollary 8.2.4.** *For every round $r$ and every $\sigma|_r$ such that $\sigma \in \mathsf{MA}$, we have:*

- *A pair $(q,r)$ is in $V$ at process $p$ at time $t$ if and only if $(q,r) \rightsquigarrow_{\sigma|_r} (p,t)$.*
- *An edge $e$ is in $E$ at process $p$ only if $e$ is in $PT_{\sigma|_r}$.*

[1] We denote the subgraph of a graph $G$, induced by a vertex-set $V$, as $G[V]$.

### 8.2.3 Computing $\Delta(.)$ in the fair/unfair message adversary

We now briefly describe the message adversary FAIR, which was introduced in [FG11]. A communication pattern is called *fair* if it is eventually distinguishable from every other communication pattern for every process. Conversely, a communication pattern is called *unfair* if it is forever indistinguishable from some other communication pattern for some process. In the case of a set of $\Pi = \{\circ, \bullet\}$ of $n = 2$ processes, communication graphs[2] $\circ\!\leftarrow\!\bullet$, $\circ\!\leftrightarrow\!\bullet$, and $\circ\!\rightarrow\!\bullet$, unfair communication patterns always come in pairs. This coined the term *special pair* of unfair communication patterns for two communication patterns that are forever indistinguishable to each other and, as was shown in [FG11], every other communication pattern is eventually distinguishable from the members of a special pair. Intuitively, a fair communication pattern can be understood as one where either a communication graph, containing the same single edge, is repeated forever (the perpetual continuation of the prefix on the very left, resp. the very right in Figure 8.1), or, from some round onwards, the communication graph where both edges appear is repeated forever. In contrast, an unfair communication pattern will eventually contain only a single edge forever, which is preceded by a communication graph containing a different edge.

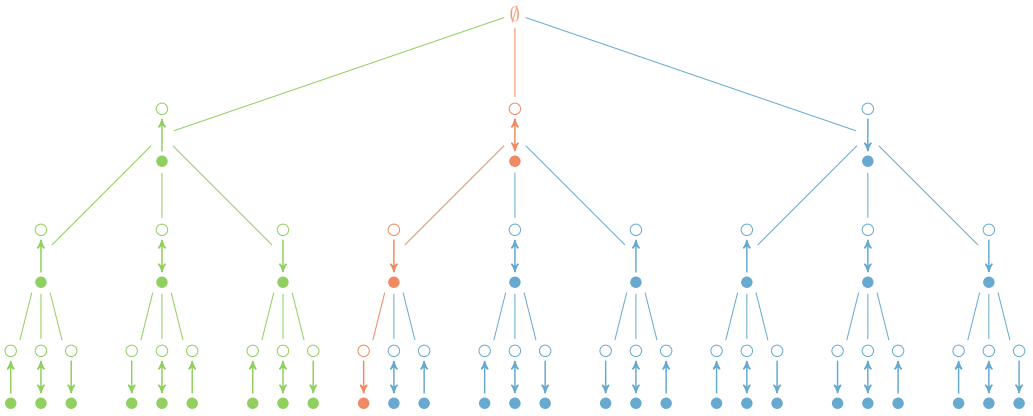


Figure 8.1: First three rounds of the communication patterns of FAIR (self-loops are assumed but not drawn to avoid clutter). With the root ($\emptyset$) at level 0, the $r^{\text{th}}$ level in the tree corresponds to the possible choices for $G_r$ and in this sense a direct path from the root to a leaf in the tree represents a communication pattern prefix. The red path exemplifies the prefix of an inadmissible communication pattern. We can observe that two nodes are adjacent on the same level precisely if the corresponding prefixes are indistinguishable for some process.

It was shown in [FG11] that for $n = 2$ consensus is solvable if and only if at least one fair communication pattern or one special pair of unfair communication patterns is not admissible. With this in mind, let FAIR be an arbitrary message adversary for $n = 2$

[2] Self-loops are assumed but not drawn to avoid clutter.

processes where at least one fair or a special case unfair communication patterns is inadmissible.

Computing a $\Delta$-function $\Delta(.)$ for FAIR is straightforward: Fix $F$ to be a set that consists of one concrete inadmissible fair communication pattern or one special pair of unfair communication patterns, which are both inadmissible (represented by the red prefix in Figure 8.1). Essentially, we can use this set $F$ to partition the prefix tree: For a given round $r$, we assign $\Delta(\sigma|_r) \leftarrow \{\bullet\}$ if there is a chain of prefixes $\rho_1 \sim \ldots \sim \rho_k$ such that no $\rho_i$ is in $F$ and $p$ never hears from $\circ$ in $\rho_1$. Analogously, $\Delta(\sigma|_r) \leftarrow \{\circ\}$ if there is a chain of prefixes $\rho'_1 \sim \ldots \sim \rho'_k$ such that no $\rho'_i$ is in $F$ and $\circ$ never hears from $\bullet$ in $\rho'_1$. Note that the only two prefixes that do not have $\{\circ, \bullet\}$ in their kernel are $\rho_1$ and $\rho'_1$ and, because they are in different partitions, we have $\Delta(\sigma|_r) \subseteq \mathrm{Ker}(\sigma|_r)$. In Figure 8.1, this corresponds to assigning $\Delta(\sigma|_r) \leftarrow \{\bullet\}$ to all green prefixes (left of the red prefix), respectively assigning $\Delta(\sigma|_r) \leftarrow \{\circ\}$ to all blue prefixes (right of the red prefix).

It can be shown that every admissible communication pattern $\notin F$ is eventually distinguishable from all elements of $F$, which implies that $\Delta(.)$, when computed in the above fashion, satisfies Assumption 1.

## 8.3 Basic Properties of Rooted Message Adversaries

In this section, we present some fundamental technical properties of rooted message adversaries that appear to be interesting in their own right. Whereas they may seem a bit simplistic at their core, with their proofs relying on arguments similar to the ones established in [KLO10], they will turn out to be essential in Section 8.4, however.

We commence by establishing that, if the root members of sufficiently many communication graphs in a sequence were influenced by a member of some set $P$ of processes, then the remaining processes are influenced by $P$ as well.

**Lemma 8.3.1.** *Let $P, Q$ be a partition of $\Pi$ with $P \neq \emptyset$, $Q \neq \emptyset$ and $\rho$ be a communication pattern that contains a subsequence $\sigma = (G_{r_i})_{i=1}^{|Q|}$ of rooted communication pattern such that, for each $G_r$ of $\sigma$, $\exists q \in \mathrm{Root}(G_r), \exists p \in P : (p, r_1 - 1) \rightsquigarrow_\rho (q, r - 1)$. Then, for all $q \in Q$ there is a $p \in P : (p, r_1 - 1) \rightsquigarrow_\rho (q, r_{|Q|})$.*

*Proof.* For $1 \leq i \leq |Q|$, let $P(r_i) = \{q \in \Pi : \exists p \in P, (p, r_1 - 1) \rightsquigarrow_\rho (q, r_i)\}$ denote those processes that heard from a member of $P$ by round $r_i$ in $\rho$. With $P(r_1 - 1) \supseteq P$ and $\overline{P}(r) = \Pi \setminus P(r)$, we show that $|P(r-1)| < n$ implies $|P(r)| > |P(r-1)|$. Since $\exists q \in \mathrm{Root}(G_r), \exists p \in P : (p, r_1 - 1) \rightsquigarrow_\rho (q, r - 1)$ by assumption, $\mathrm{Root}(G_r) \cap P(r-1) \neq \emptyset$ and there is a path in $G_r$ from some node in $P(r-1)$ to every node of $\overline{P}(r-1)$. Thus, if $|P(r-1)| < n$, there is an edge in $G_r$ from $P(r-1)$ to $\overline{P}(r-1)$ in the cut-set of $(P(r-1), \overline{P}(r-1))$ and we have $P(r) > P(r-1)$, as claimed.

Because $\sigma$ consists of $|Q|$ communication graphs and $P(r_1 - 1) \supseteq \Pi \setminus Q$, we have $|P(r_{|Q|})| = n$, which implies the lemma. □

We are now ready to prove formally in Lemma 8.3.2 that indistinguishability for a sufficiently long duration in rooted communication patterns implies that one process that is still unable to distinguish two prefixes was able to inform everyone of its initial inability to distinguish.

**Lemma 8.3.2.** *Let $\rho, \sigma$ be two communication patterns that consist of $r \geq n-1$ rooted communication graphs. If $\rho \sim \sigma$ then for all $p \in \Pi$, for all $\tau \in \{\rho, \sigma\}$, there is a $q \in \Pi$ such that $\rho|_{r-n+1} \sim_q \sigma|_{r-n+1}$ and $(q, r-n+1) \rightsquigarrow_\tau (p, r)$.*

*Proof.* Let $P = \{p \in \Pi : \rho|_{r-n+1} \sim_p \sigma|_{r-n+1}\}$. Note that because indistinguishability cannot be introduced, and we assume $\rho \sim \sigma$, we have $P \neq \emptyset$. If $P = \Pi$, the claim follows trivially since every process hears from itself, thus let $P \subset \Pi$. Pick an arbitrary $\tau \in \{\sigma, \rho\}$. With $\overline{P} = \Pi \setminus P$, consider the partition $P, \overline{P}$ of $\Pi$. We observe that at most $|P| - 1$ of the communication graphs $G_t = \tau(t)$, $r-n+1 < t \leq r$, have a member $q \in \mathrm{Root}(G_t)$ such that $\exists p' \in \overline{P} : (p', r-n+1) \rightsquigarrow_\sigma (q, t-1)$; otherwise, by Lemma 8.3.1, every $p \in \Pi$ would have $\mathrm{HO}_\tau(p) \cap \overline{P} \neq \emptyset$, which would contradict that $\rho \sim \sigma$: every process would contain in its view the view of some process that was able to distinguish $\rho|_{r-n+1}$ and $\sigma|_{r-n+1}$ and thus $\rho \not\sim \sigma$. Consequently, for at least $n-1-(|P|-1) = |\overline{P}|$ of the $G_t$, $r-n+1 < t \leq r$, we have $q \in \mathrm{Root}(G_r) \Rightarrow \nexists p \in \overline{P} : (p, r-n+1) \rightsquigarrow_\tau (q, t-1)$. Since every process has either heard from a member of $P$ or $\overline{P}$, we have instead for at least $|\overline{P}|$ of the $G_t$ that $q \in \mathrm{Root}(G_t) \Rightarrow \exists p \in P : (p, r-n+1) \rightsquigarrow_\tau (q, t-1)$. By Lemma 8.3.1, for all $q \in \Pi$ there is thus some $p \in P$ such that $(p, r-n+1) \rightsquigarrow_\tau (q, r)$, which completes the proof by the definition of $P$. □

An important consequence of Lemma 8.3.2, stated formally in Lemma 8.3.3, is that, after a sufficiently long period of indistinguishability, the kernel of the prefix propagates to all processes. This happens because the process that sends its inability to distinguish (Lemma 8.3.2) sends also its view to everyone and thus its HO-set to everyone.

**Lemma 8.3.3.** *Let $\rho, \sigma$ be two communication patterns that consist of $r \geq n-1$ rooted communication graphs. If $\rho \sim \sigma$ then $\mathrm{Ker}(\rho|_{r-n+1}) \cup \mathrm{Ker}(\sigma|_{r-n+1}) \subseteq \mathrm{Ker}(\rho) \cap \mathrm{Ker}(\sigma)$.*

*Proof.* Let $\rho' = \rho|_{r-n+1}$ and $\sigma' = \sigma|_{r-n+1}$, let $P' = \{p \in \Pi \colon \rho' \sim_p \sigma'\}$, and let $P = \{p \in \Pi \colon \rho \sim_p \sigma\}$. As indistinguishability cannot be introduced, we have $P \subseteq P'$.
From the definition of indistinguishability, it follows that

$$\forall p \in P' : \mathrm{HO}_{\sigma'}(p) = \mathrm{HO}_{\rho'}(p) \ . \tag{8.1}$$

By definition of Ker(.), we have

$$\forall p \in P' : \mathrm{Ker}(\sigma') \subseteq \mathrm{HO}_{\sigma'}(p) \wedge \mathrm{Ker}(\rho') \subseteq \mathrm{HO}_{\rho'}(p) \ . \tag{8.2}$$

Combining Equations (8.1) and (8.2) leads to

$$\forall p \in P' : \mathrm{Ker}(\rho') \cup \mathrm{Ker}(\sigma') \subseteq \mathrm{HO}_{\rho'}(p) \wedge \mathrm{Ker}(\rho') \cup \mathrm{Ker}(\sigma') \subseteq \mathrm{HO}_{\sigma'}(p) \tag{8.3}$$

$$\forall p \in P' : \mathrm{Ker}(\rho') \cup \mathrm{Ker}(\sigma') \subseteq \mathrm{HO}_{\rho'}(p) \cap \mathrm{HO}_{\sigma'}(p) \ . \tag{8.4}$$

Applying Lemma 8.3.2 implies

$$\forall q \in \Pi, \forall \tau \in \{\rho, \sigma\}, \exists p \in P' : (p, r-n+1) \rightsquigarrow_\tau (q, r) \ . \tag{8.5}$$

Combining Equation (8.4) and Equation (8.5) shows the claim. □

Finally, we show in Lemma 8.3.4 that sufficiently long indistinguishability, while not transitive in itself, implies a form of transitive indistinguishability for certain prefixes.

**Lemma 8.3.4.** *Let $\rho, \sigma, \tau$ be finite communication patterns that consist of $r \geq n-1$ rooted communication graphs. If $\rho|_{r-n+1} \not\sim \tau|_{r-n+1}$ then $\rho \not\sim \sigma$ or $\sigma \not\sim \tau$. Equivalently, if $\rho \sim \sigma$ and $\sigma \sim \tau$ then $\rho|_{r-n+1} \sim \tau|_{r-n+1}$.*

*Proof.* Let $\rho' = \rho|_{r-n+1}$, $\sigma' = \sigma|_{r-n+1}$, and $\tau' = \tau|_{r-n+1}$. Suppose that $\rho' \not\sim \tau'$, $\rho \sim \sigma$, and $\sigma \sim \tau$ hold. Let $P' = \{p \in \Pi \colon \rho' \sim_p \sigma'\}$, $Q' = \{p \in \Pi \colon \sigma' \sim_p \tau'\}$, $P = \{p \in \Pi \colon \rho \sim_p \sigma\}$, and $Q = \{p \in \Pi \colon \sigma \sim_p \tau\}$. By Lemma 8.3.2, because $\sigma \sim \tau$,

$$\forall p \in \Pi \, \exists \chi(p) \in Q' : (\chi(p), r-n+1) \rightsquigarrow_\sigma (p, r) \ . \tag{8.6}$$

Fix an arbitrary $p \in P$ and let $q = \chi(p)$. Equation (8.6) implies that the subgraph $S \subseteq \mathcal{PT}_\sigma(p)$ that is induced by the vertex-set $\{(k, r'') \colon (k, r'') \rightsquigarrow_\sigma (q, r-n+1)\}$, is exactly $S = \mathcal{PT}_{\sigma'}(q)$. Consider the subgraph $S' \subseteq \mathcal{PT}_\rho(p)$ that is induced by the vertex-set $\{(k, r'') \colon (k, r'') \rightsquigarrow_\rho (q, r-n+1)\}$. By $\rho \sim_p \sigma$ and Equation (8.6), we have that $(q, r-n+1) \rightsquigarrow_\rho (p, r)$ and thus $S' = \mathcal{PT}_{\rho'}(q)$. Furthermore, $\rho \sim_p \sigma$ implies that $S' = S$. However, from $\rho' \not\sim \tau'$ and $\sigma' \sim_q \tau'$, we have, since $\sim_q$ is clearly transitive, that $\rho' \not\sim_q \sigma'$ and so $\mathcal{PT}_{\rho'}(q) \neq \mathcal{PT}_{\sigma'}(q)$. But then $S \neq S'$, a contradiction. □

## 8.4 Computing $\Delta(.)$ for $\Diamond\mathsf{STABLE}_{n,n-1}(n)$

### Basic Definitions

We now briefly repeat the essential definitions from Chapter 6. The central object studied there is the vertex-stable root component, which is a root component that occurs as the root component in a subsequent sequence of communication graphs of some communication pattern. It can be understood as a form of stability in the sense that, even though the interconnect of the members of the root component may vary significantly from one communication graph to the next, it is nevertheless the same set of vertices that constitutes the root component of every communication graph in the sequence. Let $\Diamond\mathsf{STABLE}_{n,n-1}(x)$ denote the eventually stabilizing message adversary that can be specified as the set of those sequences $\sigma = S^r L S^\omega$ of communication graphs on $n$ nodes where $S^r$ denotes a sequence consisting of $r$ rooted (possibly different) communication graphs, $S^\omega$ denotes an infinite sequence of rooted communication graphs and $L$ denotes a sequence of $x$ consecutive rooted graphs with a vertex-stable root component, which we call the stability phase of the communication pattern. For $\Diamond\mathsf{STABLE}_{n,n-1}(x)$ and

$\sigma = S^r LS^\omega \in \Diamond\mathsf{STABLE}_{n,n-1}(x)$ we denote by $r_{stab}(\sigma) := r + x$ the last round of the stability phase.

We slightly abuse notation and denote this vertex-stable root component as $\mathrm{Root}(L)$, i.e., for each communication graph $G$ of $L$ we have $\mathrm{Root}(G) = \mathrm{Root}(L)$. Technically, the message adversary from Chapter 6 has an additional parameter, the dynamic diameter $D$, which we will neglect here for simplicity. As it was stated in Chapter 6 that $D < n$, we will simply assume $D = n-1$ subsequently and focus our investigation on the parametrization $\Diamond\mathsf{STABLE}_{n,n-1}(n)$. This has the downside of the lower bound established in Theorem 8.4.6 at the end of this section being weaker in those cases where $D = o(n)$, however it would be straight-forward to express this lower bound in terms of $D$ instead of $n$. The main result from Chapter 6 was that consensus is solvable under $\Diamond\mathsf{STABLE}_{n,n-1}(x)$ if and only if $x \geq D + 1$, i.e., $x \geq n$ for $D = n - 1$.

It is not hard to see that every round $r$ prefix $\sigma|_r$ of $\sigma \in \Diamond\mathsf{STABLE}_{n,n-1}(n)$ can be extended by a suffix that consists of a star-graph with $p$ in the center for all rounds $> r$ in order to yield an admissible communication pattern. Thus, we obtain:

**Corollary 8.4.1.** *Let $\rho, \sigma \in \Diamond\mathsf{STABLE}_{n,n-1}(n)$. If $\rho|_r \sim_p \sigma|_r$ then there exist $\rho', \sigma' \in \Diamond\mathsf{STABLE}_{n,n-1}(n)$ such that $\rho'|_r = \rho|_r$, $\sigma'|_r = \sigma|_r$ and $\rho' \sim \sigma'$.*

A rather immediate, yet essential corollary about a vertex-stable root component $R$ that occurs in some communication pattern of $\geq n-1$ rounds is that every process receives a message from every member of the vertex-stable root component. This can be expressed as:

**Corollary 8.4.2.** *With $r \geq 0$, let $\sigma = S^r LS^\omega$ be a communication pattern that is admissible under $\Diamond\mathsf{STABLE}_{n,n-1}(n)$ with stability period $L$. Then for all $q \in \mathrm{Root}(L)$, for all $p \in \Pi$ we have $(q, r+1) \rightsquigarrow_\sigma (p, r+n)$, which implies $R \subseteq \mathrm{Ker}(\sigma|_{r+n})$.*

*Proof.* Pick an arbitrary $q \in \mathrm{Root}(L)$ and, for $r+1 < s < r+n$, let $P_q(s) = \{p \in \Pi : (q, r+1) \rightsquigarrow_\sigma (p, s)\}$ and $\overline{P}_q(s) = \Pi \setminus P_q(s)$. Because every $G_s$ with $r+1 < s < r+n$ contains self-loops and, because $q \in \mathrm{Root}(G_s)$, there is a path from $q$ to every other node, we have $P_q(r+2) \geq 2$. Furthermore, there is an edge from $P_q(s)$ to $\overline{P}_q(s)$ in the cut-set $(P_q(s), \overline{P}_q(s))$, thus if $P_q(s) < n$ then $P_q(s+1) > P_q(s)$, from which we may derive the corollary. □

It can be shown with the help of Corollary 8.4.2 that two communication patterns $\rho, \sigma$ with overlapping stability phases cannot be indistinguishable.

**Corollary 8.4.3.** *Let $\rho = S^r LS^\omega$ and $\sigma = S^s L'S^\omega$ be two communication patterns that are admissible under $\Diamond\mathsf{STABLE}_{n,n-1}(n)$ with stability phases $L$ and $L'$, respectively. If $|r - s| \leq n$ and $\mathrm{Root}(L) \neq \mathrm{Root}(L')$ then $\rho|_{\max\{r,s\}+n} \not\sim \sigma|_{\max\{r,s\}+n}$.*

*Proof.* Assume the left side of the implication holds and let, w.l.o.g., $s \geq r$, that is, $\max\{r, s\} = s$. Suppose that $\rho|_{s+n} \sim \sigma|_{s+n}$. By Corollary 8.4.2, $\forall q \in \mathrm{Root}(L')\, \forall p \in \Pi : (q, s+1) \rightsquigarrow_\sigma (p, s+n)$, which implies that, for all $p \in \Pi$, the subgraph $S(p)$ of $\mathcal{PT}_\sigma(p)$ that is induced by the set $\{(q, s) : (q, s+1) \rightsquigarrow_\sigma (p, s+n)\}$ has $\mathrm{Root}(S(p)) = \mathrm{Root}(L')$. Since $\rho|_{s+n} \sim \sigma|_{s+n}$, there is a $p$ such that the subgraph $S'(p)$ of $\mathcal{PT}_\rho(p)$ that is induced by the set of pairs $\{(q, s) : (q, s+1) \rightsquigarrow_\rho (p, s+n)\}$ satisfies $S(p) = S'(p)$. We observe that by the definition of $PT_\rho(p)$, if $(q, s) \in S'(p)$ then $\mathrm{In}_{\rho(s)}(q) = \{q' : ((q', s-1), (q, s)) \in S'\}$ where $\mathrm{In}_{\rho(s)}(q)$ are the in-neighbors of $q$ in $\rho(s)$. Because of this and since $S(p) = S'(p)$, we have $\mathrm{Root}(S'(p)) = \mathrm{Root}(L') = \mathrm{Root}(\rho(s))$. This, however is a contradiction, since we have $s \leq r + n$ and hence $\mathrm{Root}(\rho(s)) = \mathrm{Root}(L) \neq \mathrm{Root}(L')$ by assumption. □

### Algorithm for Computing $\Delta(.)$

We show now that $\Delta(\sigma)$ can be computed under $\Diamond\mathsf{STABLE}_{n,n-1}(n)$ by the centralized algorithm given in Algorithm 8.3. It operates by assigning a value $\Delta(\sigma)$ to a prefix $\sigma$ that has no value assigned yet if (i) there is a prefix $\rho$ that is still indistinguishable to $\sigma$ for $2(n-2)$ rounds after the stability phase of $\rho$ ended or (ii) $2(n-2)$ rounds after the stability phase of $\sigma$ itself ended. In case (i), $\sigma$ adopts the assignment of $\Delta(\rho)$, in case (ii) it assigns to $\Delta(\sigma)$ the vertex-stable root component of its own stability phase.

**Algorithm 8.3:** Computing $\Delta(\sigma|_r)$ for $\sigma \in \Diamond\mathsf{STABLE}_{n,n-1}(n)$.

```
1  Let X = {ρ|_r : ρ ∈ ◇STABLE_{n,n−1}(n)} be the set of all r-round prefixes.
2  if r < n + 2(n − 1) then
3  |  Δ(σ|_r) ← ∅
4  else
5  |  foreach σ ∈ X do Δ(σ) ← Δ(σ|_{r−1})
6  |  foreach σ ∈ X with Δ(σ) = ∅ do
7  |  |  if ∃ρ ∼ σ : Δ(ρ) ≠ ∅ ∧ ρ = S^{r−n−2(n−1)} L S^{2(n−1)} then
8  |  |  |  Δ(σ) ← Δ(ρ)
9  |  foreach σ ∈ X with σ = S^{r−n−2(n−1)} L S^{2(n−1)} and Δ(σ) = ∅ do
10 |  |  Δ(σ) ← Root(L)
```

Before providing a formal proof for the correctness of Algorithm 8.3 in Theorem 8.4.4, we explain briefly the intuition of why the computed function $\Delta(\sigma|_r)$ satisfies Definition 8.2.1 and Assumption 1. Since $\Delta(\sigma|_r)$ outputs $\emptyset$ initially (Line 3), is always assigned the value of the last round (Line 5), and will get assigned a value $\neq \emptyset$ at most once (as is evident from the guards preceding lines 8 and 10), we have that there is some $K$ such that $\Delta(\sigma|_r)$ outputs $\emptyset$ until it outputs $K$ eventually.

In order to see that $K \neq \emptyset$ and $K \subseteq \mathrm{Ker}(\sigma|_r)$, we note that $K$ is eventually assigned via Line 10, at the latest in round $r_{stab} + 2(n-1)$. In this case, $K \subseteq \mathrm{Ker}(\sigma|_r)$ follows from Corollary 8.4.2. As this is the mechanism by which the first assignment of $\Delta(.)$ happens to a prefix, an inductive argument shows that $K \subseteq \mathrm{Ker}(\sigma|_r)$ also holds when $K$

is assigned via Line 8: $\sigma|_r$ was indistinguishable to some $\rho$, whereas the stability phase of the latter already occurred long enough ago for us to be able to apply Lemma 8.3.2 in conjunction with Corollary 8.4.2 to show that $\Delta(\rho) \subseteq \mathrm{Ker}(\sigma|_r)$.

Finally, to realize that Assumption 1 holds, we first note that every communication pattern $\sigma$, at the latest $2n-2$ rounds after its stability phase ends, gets assigned a value $K \neq \emptyset$ to $\Delta(.)$. This implies that all indistinguishable prefixes also get assigned a value $K' \neq \emptyset$. It thus suffices to show that an assignment $K$ to $\Delta(\sigma)$ implies that all communication patterns $\rho$ that are indistinguishable to $\sigma$ and have $\Delta(\rho) \neq \emptyset$, have in fact $\Delta(\rho) = K$. This can be accomplished by using Lemma 8.3.4, as an assignment with $K \neq K'$ would imply the indistinguishability of two communication patterns with overlapping stability phase, which is impossible according to Corollary 8.4.3.

**Theorem 8.4.4.** *Algorithm 8.3 computes a $\Delta$-function $\Delta(.)$ for $\Diamond\mathsf{STABLE}_{n,n-1}(n)$ that satisfies Assumption 1. Furthermore, for a given $\sigma \in \Diamond\mathsf{STABLE}_{n,n-1}(n)$, the computed function satisfies $\Delta(\sigma') \neq \emptyset$ for all $\sigma' \sim \sigma|_r$ for $r > r_{stab} + 2n - 2$.*

*Proof.* Pick an arbitrary $\sigma \in \Diamond\mathsf{STABLE}_{n,n-1}(n)$. First, we show that there is a round $s$ and a non-empty set $K \subseteq \Pi$ such that, for all rounds $r \leqslant s$, $\Delta(\sigma|_r) = 0$ and, for all $r > s$, $\Delta(\sigma|_r) = K$. It is clear from line Line 5, that $\Delta(\sigma|_r) = \emptyset$ until either Line 8 or Line 10 is executed. After this, Line 5 ensures that $\Delta(\sigma|_r) = K \neq \emptyset$. Furthermore, if $\Delta(\sigma|_{r_{stab}(\sigma)+2n-3}) = \emptyset$, then, during the computation of round $r_{stab}(\sigma) + 2n - 2$, the guard of Line 7, resp. Line 9 is passed and Line 8, resp. Line 10 is executed, setting $\Delta(\sigma|_{r_{stab}(\sigma)+2n-2}) \leftarrow K \neq \emptyset$.

Next, we show by induction on $r$ that, for all $\rho \in \Diamond\mathsf{STABLE}_{n,n-1}(n)$, if $\Delta(\rho|_r) = K$ then $K \subseteq \mathrm{Ker}(\rho|_t)$ where $t = \min\{r_{stab}(\rho), r - n + 1\}$. For the induction base, consider the smallest $r$ such that $\Delta(\rho|_r) \leftarrow K \neq \emptyset$ is assigned to $\rho \in \Diamond\mathsf{STABLE}_{n,n-1}(n)$ for the first time. Note that $r \geq n + 2(n-1)$ because of Line 3. We can see from the code that it must be assigned via Line 10. In this case, the claim follows immediately from Corollary 8.4.2.

For the induction step, if $K$ is assigned to some $\rho|_r$ in round $r$ via Line 10, the claim follows again directly from Corollary 8.4.2, thus assume that it is assigned via Line 8 due to the existence of some $\tau \in \Diamond\mathsf{STABLE}_{n,n-1}(n)$ such that $\rho|_r \sim \tau|_r$, $\tau|_r = S^{r-3n+2}LS^{2(n-1)}$, and $\Delta(\tau|_r) = K \neq \emptyset$. This, together with Corollary 8.4.3, implies that $r_{stab}(\rho) \geq r - (n-1)$, hence $t = r - (n-1)$. By the induction hypothesis, $K \subseteq \mathrm{Ker}(\tau|_{r_{stab}(\tau)}) = \mathrm{Ker}(\tau|_{r-2(n-1)})$, thus, by Lemma 8.3.2, $K \subseteq \mathrm{Ker}(\rho|_{r-(n-1)}) = \mathrm{Ker}(\rho|_t)$.

Finally, we show that Assumption 1 holds. According to Line 10, for every $\sigma \in \Diamond\mathsf{STABLE}_{n,n-1}(n)$, at latest in round $r = r_{stab}(\sigma) + 2n - 2$, we have $\Delta(\sigma|_r) \neq \emptyset$. Furthermore, by Line 8, for every $\rho \in \Diamond\mathsf{STABLE}_{n,n-1}(n)$ with $\rho|_r \sim \sigma|_r$, we have $\Delta(\rho) \neq \emptyset$ as well. In order to show the claim, it thus suffices to show the following lemma:

**Lemma 8.4.5.** *For two arbitrary round $r$ prefixes $\sigma, \rho$, if $\sigma \sim \rho$, and $\Delta(\sigma) \leftarrow K \neq \emptyset$ is assigned, and $\Delta(\rho) = K' \neq \emptyset$ then $K = K'$.*

*Proof.* Consider an assignment $\Delta(\sigma) \leftarrow K \neq \emptyset$ to some round $r$-prefix $\sigma$. It follows from Lines 8 and 10 that there exists $\sigma' = S^{r-3n+2}LS^{2n-2}$ and $\sigma' \sim \sigma$. Suppose there is an $r$-prefix $\rho$ such that $\sigma \sim \rho$, $\Delta(\rho) = K'$, and $\emptyset \neq K' \neq K$. Again by Lines 8 and 10, there is an $r$-prefix $\rho' = S^{r'-3n+2}L'S^{2n-2+r-r'}$ with $r' \leq r$ and, due to Corollary 8.4.1, $\rho' \sim \rho$. Thus, because indistinguishability cannot be introduced, we have

$$\rho' \sim \rho \sim \sigma \quad \Rightarrow \quad \rho'|_{r-n+2} \sim \sigma|_{r-n+2}$$

where we used Lemma 8.3.4 for the implication. By similar arguments, we get

$$\rho'|_{r-n+2} \sim \sigma|_{r-n+2} \sim \sigma'|_{r-n+2} \Rightarrow \quad \rho'|_{r-2n+3} \sim \sigma'|_{r-2n+3} \ .$$

Applying Corollary 8.4.3 reveals that $r \geq r'+n$. But since $\rho'|_{r-n+2} \sim \sigma|_{r-n+2}$, $\Delta(\sigma) \leftarrow K'$ was already assigned in round $r - n + 2$, a contradiction. □

□

Plugging the computation of $\Delta$ from Algorithm 8.3 into Algorithm 8.1 reveals that our algorithm terminates a linear number of rounds after the end of the stability phase. The lower bound below shows that termination cannot occur in a sublinear number of rounds after the end of the stability phase, by providing a suitable counter-example, thus revealing that our algorithm has an asymptotically optimal termination time.

**Theorem 8.4.6.** *Solving binary consensus under* $\Diamond\mathsf{STABLE}_{n,n-1}(n)$ *takes at least* $r_{stab} + \Omega(n)$ *rounds.*

*Proof.* Fix an arbitrary input assignment $C^0$ and suppose some binary consensus algorithm $A$ terminates in round $r_{stab}(\sigma) + n - 3$ in every run $\langle C^0, \sigma\rangle$. Let $\sigma$ be the communication pattern where each communication graph consists of the same cycle, thus $r_{stab}(\sigma) = n$. Starting from the initial configuration, where all input values are 0 and toggling, one at a time, the input assignments to 1 until we arrive at the initial configuration where all inputs are 1, the validity condition shows that there are two initial configurations $C$, $C'$ that differ only in the input value of a single process $p$, yet all processes decide 0 in $\langle C, \sigma|_{2n-3}\rangle$ and 1 in $\langle C', \sigma|_{2n-3}\rangle$.

Consider the graph sequence $\rho$ that is $\sigma|_{n-2}$, followed by at least $n-1$ repetitions of a directed line graph that starts from the in-neighbor $p'$ of $p$ in the cycle, i.e., from $p'$, such that $(p', p)$ is in the cycle. It is not hard to see that $\rho|_{2n-3} \sim_q \sigma|_{2n-3}$, where $q$ is the process at the end of the line graph. Thus $q$, and by agreement, all processes, decide 0 in $\varepsilon = \langle C, \rho\rangle$ and 1 in $\varepsilon' = \langle C', \rho\rangle$. This is a contradiction, however, since $p \notin \mathrm{HO}_\varepsilon(p') \cup \mathrm{HO}_{\varepsilon'}(p')$ and thus $\varepsilon \sim_{p'} \varepsilon'$. □

Chapter 9

# Message Adversary Simulations

A prominent formalism when regarding the solvability of consensus, especially in asynchronous systems prone to crashes, are *failure detectors* [CT96]. Informally (see Section 9.3 for a formal definition), a failure detector is an oracle whose output depends on the failure pattern, i.e., the occurrence of the (crash) failures so far in the run. Ideally, a failure detector precisely captures what is needed for consensus to become solvable in a given system as an abstract object, called the *weakest failure detector* for consensus in a given system. The basic idea is that, given a failure detector $\mathsf{FD}$, if we can simulate the output of a failure detector $\mathsf{FD}'$, then $\mathsf{FD}'$ is weaker than $\mathsf{FD}$. If it can be shown that some failure detector $\mathsf{FD}'$ can be simulated by an arbitrary failure detector $\mathsf{FD}$ that makes consensus solvable, then $\mathsf{FD}'$ is the weakest failure detector for consensus. In this chapter, we investigate how this approach can be applied in the message adversary model.

We will find that, even though a crash failure of process $p$ at time $t$ is indistinguishable from the outside to $p$ stopping to send messages at time $t$, there are some intricacies when applying the failure detector approach directly to the message adversary setting. For this purpose, we formally introduce message adversary simulations, which should, in analogy to the hierarchy of failure detectors, yield a *strongest message* adversary that still allows to solve consensus. Note that in the case of a failure detector, we usually desire to find the weakest failure detector, i.e., the failure detector that provides the least amount of information to the algorithm and still makes consensus solvable, whereas in the case of a message adversary, we are interested mostly in the strongest, i.e., the most powerful message adversary that still permits a solution algorithm. We remark that even though our solvability results from the previous chapters enclose the impossibility/possibility border of consensus quite tightly under some message adversaries, they did not immediately yield a "strongest" message adversary for consensus. A complete characterization will be given using a topological framework in Chapter 10.

Raynal and Stainer [RS13] already established an interesting relation between synchronous message passing systems (abbreviated $\mathcal{SMP}$) with message adversaries and asynchronous message-passing systems (abbreviated $\mathcal{AMP}$) with process crashes augmented by failure

detectors. Among other results, they showed that $\mathcal{AMP}$ in conjunction with the weakest failure detector $(\Omega, \Sigma)$ for consensus with an arbitrary number of crashes [DGFG+04] can be simulated in $\mathcal{SMP}$ with the message adversary $\mathsf{SOURCE} \cap \mathsf{QUORUM}$ (and *vice versa*). This message adversary guarantees communication graphs such that, for every pair of processes $p$, $q$ and every pair of rounds $r$, $r'$, there is some process $\ell$ such that $(\ell, p) \in G_r$ and $(\ell, q) \in G_{r'}$. Additionally, there is a process $p$ and some round $r_0$ such that for all processes $q$ and all rounds $r \geq r_0$ we have $(p, q) \in G_r$. Consequently, every $(\Omega, \Sigma)$-based consensus algorithms for $\mathcal{AMP}$ can be employed atop of this simulation in $\mathcal{SMP}$ with $\mathsf{SOURCE} \cap \mathsf{QUORUM}$.

Despite these results, the question remains whether failure-detector-based consensus algorithms on top of failure detector implementations can indeed compete with specifically designed consensus algorithms for $\mathcal{SMP}$ for some given message adversary. In particular, is it always possible to implement the weakest failure detector $(\Omega, \Sigma)$ atop of $\mathcal{SMP}$ with a message adversary that permits a consensus algorithm? Conversely, given that $(\Omega, \Sigma)$ is a weakest failure detector for consensus, is it somehow possible to simulate $\mathcal{SMP}$ with a strong(est) message adversary for consensus atop of $\mathcal{AMP}$ augmented with $(\Omega, \Sigma)$? Interestingly, it was shown in [BRS+18] that neither is possible: $\Sigma$ cannot be implemented in $\mathcal{SMP}$ with some message adversary $\Diamond\mathsf{STABLE}_{n,D}(\infty)$ that admits a consensus algorithm, and the latter cannot be implemented in $\mathcal{AMP}$ equipped with $(\Omega, \Sigma)$ either. Essentially, the reason is that all properties achievable in $\mathcal{AMP}$ with failure detectors are inherently time-free, i.e., of eventually-forever-type, whereas $\mathcal{SMP}$ with message adversaries facilitates time-dependent properties: The latter are sometimes too short-lived to guarantee eventual-forever properties, however, and, conversely, cannot be extracted from eventual-forever properties either.

In this chapter, we thus avoid the detour via failure detectors and introduce the concept of *message adversary simulations* in $\mathcal{SMP}$, using the HO model [CBS09], in particular the emulations of HO-predicates established there, as a basis. Inspired by the definition of a weakest failure detector, we also define a notion for a *strongest message adversary*: A strongest message adversary $\mathsf{MA}$ for a given problem $\mathcal{P}$ is such that (i) it admits a solution algorithm for the problem $\mathcal{P}$ in $\mathcal{SMP}$ subject to $\mathsf{MA}$, and (ii) every message adversary $\mathsf{MA}'$ that admits a solution algorithm for $\mathcal{P}$ in $\mathcal{SMP}$ under $\mathsf{MA}'$ allows to simulate $\mathcal{SMP}$ subject to $\mathsf{MA}$.

Using message adversary simulations, we prove that the message adversary $\mathsf{STAR}$ (which generates an infinite sequence of identical star graphs $G, G, \dots$) is a strongest message adversary for consensus. Moreover, we show that every message adversary $\mathsf{MA}$ that satisfies $\mathsf{STAR} \subseteq \mathsf{MA}$ and allows to solve consensus is also a strongest message adversary. It hence turns out that both $\mathsf{SOURCE} \cap \mathsf{QUORUM}$ and $\Diamond\mathsf{STABLE}_{n,D}(\infty)$ are strongest message adversaries, even though neither one is contained in the other. Moreover, there are interesting and sometimes apparently paradoxical consequences resulting from our findings: $\mathcal{AMP}$ with $(\Omega, \Sigma)$ allows to simulate $\mathcal{SMP}$ with the strongest message adversary $\mathsf{SOURCE} \cap \mathsf{QUORUM}$ and, in addition, the former can be implemented in the latter. $\mathcal{SMP}$ with the strongest message adversary $\mathsf{SOURCE} \cap \mathsf{QUORUM}$, in turn, can

be simulated in $\mathcal{SMP}$ with the strongest message adversary $\Diamond\mathsf{STABLE}_{n,D}(\infty)$ and vice versa. Interestingly, however, $\mathcal{AMP}$ with $(\Omega, \Sigma)$ cannot be implemented in $\mathcal{SMP}$ with $\Diamond\mathsf{STABLE}_{n,D}(\infty)$. We will elaborate on this apparent paradox in Section 9.5.

**Chapter organization:** We commence by providing the computational details of the synchronous message passing model $\mathcal{SMP}$ with message adversaries in Section 9.2. Subsequently, asynchronous message passing models $\mathcal{AMP}$ with failure detectors are introduced in Section 9.3, along with a reiteration of some failure-detector-related definitions and results. In Section 9.4, we introduce the key simulation relation between message adversaries and prove that the message adversary STAR is strongest for solving consensus in $\mathcal{SMP}$. We also show that this result generalizes to a fairly large class of strongest message adversaries. In Section 9.5, we discuss the consequences arising from the fact that STAR is a strongest message adversary.

## 9.1 Related Work

In [RS13], Raynal and Stainer related message adversaries to asynchronous systems with failure detectors, which stimulated our interest in the problems addressed in this chapter. There, it was shown, among other insightful results, that the message adversary SOURCE + QUORUM allows to simulate an asynchronous message passing system with process crashes augmented by the failure detector $(\Sigma, \Omega)$. A failure detector is an oracle that can be queried by all processes in order to gain some information on failures (traditionally process crashes) that could otherwise not be obtained. Since $(\Sigma, \Omega)$ is a weakest failure detector for consensus [DGFG$^+$04], it is possible to use classic consensus algorithms on top of this simulation. Furthermore, as $\Sigma$ is the weakest failure detector to simulate shared memory on top of wait-free asynchronous message passing [DGFG$^+$04], even shared memory algorithms that rely on $\Omega$ could be employed. Related to this, it was shown in [BRS$^+$18] that $\Sigma$ cannot be implemented in $\mathcal{SMP}$ with message adversary $\mathrm{VSSC}_{D,E}(\infty)$ that admits a consensus algorithm, and the latter cannot be implemented in $\mathcal{AMP}$ equipped with $(\Omega, \Sigma)$ either. Because of these difficulties, [SSW18] introduced a message adversary simulation that was intended to provide a suitable counterpart to the failure detector hierarchy. We will discuss these results in detail in the remaining chapter.

## 9.2 The Computational Model $\mathcal{SMP}$

In order to introduce message adversary simulations, we need to study the details of how computation proceeds in the model $\mathcal{SMP}$ in a little more depth than so far in this thesis. Because of this, we reuse significant parts of the formalism of the HO model introduced in [CBS09], which provides a solid basis for this purpose. In addition to our usual set $\Pi = \{p_1, \ldots, p_n\}$ of $n > 1$ processes with unique identifiers, we now assume a set of messages $M$, which includes a *null* placeholder indicating the absence of a message. Each

process $p \in \Pi$ consists of the following components: a set of states denoted by $\mathsf{states}_p$, a set $\mathsf{init}_p \subseteq \mathsf{states}_p$ of initial states, and, for each positive integer $r > 0$, a message sending function $S_p^r$ mapping $\mathsf{states}_p \times \Pi$ to a unique (possibly *null*) message $m_p$, and a state-transition function $T_p^r$ mapping $\mathsf{states}_p$ and partial vectors (indexed by $\Pi$) $z_p$ of elements of $M$ to $\mathsf{states}_p$. The message sending function and state-transition function of the processes for every round $r > 0$, together with the initial states, represents the algorithm under consideration.

Computations in the HO model are composed of infinitely many rounds, which are communication-closed layers in the sense that every message sent in a round can be received only in that round. In each round $r$, process $p$ first applies $S_p^r$ to the current state $(p, r-1) \in \mathsf{states}_p$ and every processes $q$ emits the "messages" to be sent to $q$, and then, for a subset $\mathrm{HO}(p,r) = \mathrm{In}_p(G_r)$ of $\Pi$ (indicating the processes which $p$ hears of, i.e., the in-neighbors of $p$ in $G_r$), applies $T_p^r$ to its current state and the partial vector of incoming messages whose support is $\mathrm{In}_p(G_r)$ to compute $(p, r)$. As before, we call the collection of the round $r$ states $((p_1, r), \ldots, (p_n, r))$ at the end of the round, i.e., after application of the state-transition transition function, the round $r$ configuration.

Following the notation of [RS13], the abbreviation we use for such models is $\mathcal{SMP}_n[adv : \mathsf{MA}]$, where $n$ is the number of processes and $\mathsf{MA}$ is the name assigned to a set of admissible graph sequences. For example, $\mathcal{SMP}_n[adv : \mathsf{SOURCE}, \mathsf{QUORUM}]$ denotes the synchronous model with message adversary $\mathsf{SOURCE} \cap \mathsf{QUORUM}$ defined below.

The next two message adversaries have been introduced in [RS13]: $\mathsf{SOURCE}$ requires that, eventually, there is a round after which some process successfully sends a message to every process in the system:

**Definition 9.2.1** ($\mathsf{SOURCE}$)**.** *The message adversary* $\mathsf{SOURCE}$ *is the set of all sequences of communication graphs* $(G_r)_{r>0}$*, where* $\exists p \in \Pi : \exists r_0 \geqslant 1 : \forall r \geqslant r_0 : \forall q \in \Pi : (p, q) \in G_r$*.*

$\mathsf{QUORUM}$ requires that every two processes $p, q \in \Pi$ hear from some common process $\ell \in \Pi$, in every pair of rounds $r_p, r_q$. Moreover, it requires that the set of processes that appear strongly correct (formally introduced in Definition 9.3.5) is not empty:

**Definition 9.2.2** ($\mathsf{QUORUM}$)**.** *The message adversary* $\mathsf{QUORUM}$ *is the set of all sequences of communication graphs* $(G_r)_{r>0}$*, where* $\forall p, q \in \Pi, \forall r_p, r_q : (\{\ell : (\ell, p) \in G_{r_p}\} \cap \{\ell : (\ell, q) \in G_{r_q}\} \neq 0)]$*, and the set of strongly correct processes is not empty.*

$\mathcal{SMP}_n[adv : \mathsf{SOURCE}, \mathsf{QUORUM}]$ and $\mathcal{SMP}_n[adv : \Diamond\mathsf{STABLE}_n(\infty, ])$ allow a solution algorithm for consensus according to [RS13] and [BRS+18], respectively.

## 9.3 Failure Detectors in Asynchronous Systems

In an asynchronous message passing system, abbreviated $\mathcal{AMP}$, processes may take steps at any time, and the time between two steps must be finite and larger than 0.

Given some initial configuration $C^0 = ((p_1, 0), \ldots, (p_n, 0))$ consisting of the initial states of all processes, a run (also called execution) in $\mathcal{AMP}$ is a sequence of infinitely many steps of every process starting from $C^0$, where every message sent must eventually be received and processed in finite time (except when failures occur, see below). Note that the end-to-end delay of a single message, from the time it is broadcast to the time it is received and processed, can be different for different recipients. Conceptually, we assume a (non-observable) clock with domain $\mathcal{T} = \{0, 1, 2, \ldots\}$ and require all computing steps in a run to occur synchronized with this clock. A commonly considered failure mode for asynchronous systems are crash failures. There, a faulty process may simply stop to take points at an arbitrary point in time.

**Definition 9.3.1** (Process crashes)**.** *We say that process $p_i$ crashes at time $t \geqslant 0$, if it stops executing its computing step at time $t$ (possibly leaving it incomplete) and does not execute further steps at time $t' > t$.*

A failure detector [CT96] is an oracle that can be queried by a process during its computing step. Formally, a failure detector $\mathcal{D}$ with range $\mathcal{R}$ maps each failure pattern $F$ to a non-empty set of histories with range $\mathcal{R}$, where a history $H$ with range $\mathcal{R}$ is a function $H : \Pi \times \mathcal{T} \to \mathcal{R}$. The failure pattern is a function $F : \mathcal{T} \to 2^{\Pi}$ that maps each $t \in \mathcal{T}$ to the processes that have crashed by $t$. The set of all possible failure patterns is called the environment. Finally, $\mathcal{D}(F)$ denotes the set of possible failure detector histories permitted by $\mathcal{D}$ for the failure pattern $F$.

**Definition 9.3.2.** *$\mathcal{AMP}_{n,x}[fd : FD]$ denotes the asynchronous message passing model consisting of $n$ processes, at most $x$ of which may crash in a run, augmented by failure detectors $FD$.*

Two important failure detectors for consensus in $\mathcal{AMP}$ are $\Sigma$ and $\Omega$, as their combination $(\Omega, \Sigma)$ is known to be a weakest failure detector in the wait-free environment [DGFG$^+$04].

**Definition 9.3.3.** *The eventual leader failure detector $\Omega$ has range $\Pi$. For each failure pattern $F$, for every history $H \in \Omega(F)$, there is a time $t \in \mathcal{T}$ and a correct process $q$ s.t. for every process $p$ and for every $t' \geqslant t$, $H(p, t') = q$.*

**Definition 9.3.4.** *The quorum failure detector $\Sigma$ has range $2^{\Pi}$. For each failure pattern $F$, for every $H \in \Sigma(F)$, two properties hold: (1) for every $t, t' \in \mathcal{T}$ and $p, q \in \Pi$ we have $H(p, t) \cap H(q, t') \neq \emptyset$ and (2) there is a time $t \in \mathcal{T}$ s.t. for every process $p$, for every $t' \geqslant t$, $H(p, t') \subseteq \Pi \setminus \bigcup_{t \in \mathcal{T}} F(t)$.*

In order to relate such failure detector models to our message adversaries, we use the simple observation that the externally visible effect of a process crash can be expressed in our setting: Since correct processes in asynchronous message passing systems perform an infinite number of steps, we can assume that they send an infinite number of (possibly empty) messages that are eventually received by all correct processes. As in [RS13], we

hence assume that the correct (= non-crashing) processes in the simulated $\mathcal{AMP}$ are the *strongly correct processes*. Informally, a strongly correct process is able to disseminate its state to all other processes infinitely often.

**Definition 9.3.5** (Faulty and strongly correct processes)**.** *Given an infinite communication pattern $\sigma$, process $p$ is* faulty *in a run with $\sigma$ if there is a round $r$, s.t., for some process $q$, for all $r' > r$: $p \notin \mathrm{CP}_q^{r'}(r)$.*
*Let $\mathcal{C}(\sigma) = \left\{p \in \Pi \mid \forall q \in \Pi, \forall r \in \mathbb{N}, \exists r' > r : p \in \mathrm{CP}_q^{r'}(r)\right\}$ denote the strongly correct (= non-faulty) processes in any run with $\sigma$.*

If a given process influences just one strongly correct process infinitely often, it would transitively influence all processes in the system, hence would also be strongly correct. Therefore, in order not to be strongly correct, a faulty process must not influence *any* strongly correct process infinitely often. We can hence define failure patterns as follows:

**Definition 9.3.6** (Failure Pattern)**.** *The failure pattern associated with communication pattern $\sigma$ is a function $F_\sigma : \mathbb{N} \to 2^\Pi$ s.t. $p \in F_\sigma(r)$ if, and only if, for all processes $q \in \mathcal{C}(\sigma)$, for all $r' > r$: $p \notin \mathrm{CP}_q^{r'}(r)$.*

We note that $F_\sigma(r) \subseteq F_\sigma(r+1)$, as required.

The following lemmas have originally been proven for the message adversary $\mathrm{VSSC}_{D,E}(d)$ in [BRS+18]. Fortunately, these proofs translate almost literally to the weaker message adversaries $\Diamond\mathsf{STABLE}_{n,D}(d)$ and $\Diamond\mathsf{STABLE}_n(k,d)$ given in Definitions 6.2.4 and 7.6.1.

**Lemma 9.3.7.** *$\mathcal{AMP}_{n,n-1}[fd : \Sigma]$ cannot be built in $\mathcal{SMP}_n[adv : \Diamond\mathsf{STABLE}_{n,D}(\infty)]$.*

*Proof.* We will prove our lemma for $n = 2$ for simplicity, as it is straightforward to generalize the proof for arbitrary $n$. Suppose that, for all rounds $r$ and every processes $p$, some algorithm $\mathcal{A}$ computes $out(p,r)$ s.t. for an admissible failure pattern $F$, $out \in \Sigma(F)$. Consider the graph sequence $\sigma = (p \to q)_{r \geqslant 1}$. Clearly, the failure pattern associated with $\sigma$ is $F_\sigma(r) = \{q\}$. Hence, in the run $\varepsilon$ starting from some initial configuration $C^0$ with sequence $\sigma$, there is some round $r'$ s.t. $out(p,r) = \{p\}$ for any $r > r'$ by Definition 9.3.4. Let $\sigma' = (p \to q)_{r=1}^{r'}(p \leftarrow q)_{r>r'}$. By similar arguments as above, in the run $\varepsilon'$ that starts from $C^0$ with sequence $\sigma'$, there is a round $r''$ such that $out(q,r) = \{q\}$ for any $r > r''$. Finally, for $\sigma'' = (p \to q)_{r=1}^{r'}(p \leftarrow q)_{r=r'+1}^{r''}(p \leftrightarrow q)_{r>r''}$, let $\varepsilon''$ denote the run starting from $C^0$ with graph sequence $\sigma''$. Until round $r'$, $\varepsilon'' \sim_p \varepsilon$, hence, as shown above, $out(p,r') = \{p\}$ in $\varepsilon''$. Similarly, until round $r''$, $\varepsilon'' \sim_q \varepsilon'$ and hence $out(q,r'') = \{q\}$ in $\varepsilon''$. Clearly, $\sigma, \sigma', \sigma'' \in \Diamond\mathsf{STABLE}_{n,D}(\infty)$ and $F_{\sigma''}(r) = \{\}$, that is, no process is faulty in $\sigma''$. However, in $\varepsilon''$, $out(p,r') \cap out(q,r'') = \emptyset$, a contradiction to Definition 9.3.4. □

We continue with the definitions of generalized failure detectors for the $k$-set agreement setting in crash-prone asynchronous message passing systems.

**Definition 9.3.8.** *The range of the failure detector $\Omega_k$ is all $k$-subsets of $2^\Pi$. For each failure pattern $F$, for every history $H \in \Omega_k(F)$, there exists a set $LD = \{q_1, \ldots, q_k\} \in 2^\Pi$ and $t \in \mathcal{T}$ such that $LD \cap \mathcal{C} \neq \emptyset$ and for all $t' \geqslant t, p \in \mathcal{C} : H(p, t') = LD$.*

**Definition 9.3.9.** *The failure detector $\Sigma_k$ has range $2^\Pi$. For each failure pattern $F$, for every $H \in \Sigma_k(F)$, two properties must hold: (1) for every $t, t' \in \mathcal{T}$ and $S \in \Pi$ with $|S| = k + 1$, $\exists p, q \in S : H(p, t) \cap H(q, t') \neq \emptyset$, (2) there is a time $t \in \mathcal{T}$ s.t. for every process $p$, for every $t' \geqslant t$: $H(p, t') \subseteq \mathcal{C}$.*

$k$-set agreement in our lock-step round model with link failures allows non-temporary partitioning, which in turn makes it impossible to use the definition of crashed and correct processes from the previous section: In a partitioned system, every process $p$ has at least one process $q$ such that $\forall r' > r : p \in \mathrm{CP}_q^{r'}(r)$, but no $p$ usually reaches all $q \in \Pi$ here. Definition 9.3.1 hence implies that there is no correct process in this setting. Hence, we employ the following generalized definition:

**Definition 9.3.10.** *Given an infinite graph sequence $\sigma$, let a* minimal source set *$S$ in $\sigma$ be a set of processes with the property that $\forall q \in \Pi, \forall r > 0$ there exists $p \in S, r' > r$ such that $p \in \mathrm{CP}_q^{r'}(r)$. The set of* weakly correct *processes $\mathcal{WC}(\sigma)$ of a sequence $\sigma$ is the union of all minimal source sets $S$ in $\sigma$.*

This definition is a quite natural extension of correct processes in a model, which allows perpetual partitioning of the system. Based on this definition of weakly correct processes, it is possible to generalize some of our results for consensus solvability (obtained for $\Sigma$ and $\Omega$): First, we show that $\Sigma_k$ cannot be implemented in $\Diamond\mathsf{STABLE}_n(k, \infty)$, since the latter allows the system to partition into $k$ isolated connected components.

**Lemma 9.3.11.** *$\mathcal{AMP}_{n,n-1}[fd : \Sigma_k]$ cannot be built in $\mathcal{SMP}_n[adv : \Diamond\mathsf{STABLE}_n(k, \infty)]$.*

*Proof.* For $k = 1$, we can rely on Lemma 9.3.7, as every $\sigma \in \Diamond\mathsf{STABLE}_{n,D}(\infty)$ is also admissible in $\Diamond\mathsf{STABLE}_n(k, \infty)$. Hence, $\Sigma_1 = \Sigma$ cannot be implemented in $\Diamond\mathsf{STABLE}_n(1, \infty) \supseteq \Diamond\mathsf{STABLE}_{n,D}(\infty)$.

The impossibility can be extended to $k > 1$ by choosing some $\sigma$ that (i) perpetually partitions the system into $k$ components $\widetilde{P} = \{P_1, \ldots, P_k\}$ that each have a single source component and consist of the same processes throughout the run, and (ii) demands eventually a vertex stable source component in every partition forever. Pick an arbitrary partition $P \in \widetilde{P}$. If $|P| > 1$, such a sequence does not allow to implement $\Sigma$ in $P$ (e.g., the message adversary could emulate the graph sequence used in Lemma 9.3.7 in $P$). We hence know that $\exists p, p' \in P$ and $\exists r, r'$ such that $out(p, r) \cap out(p', r') = \emptyset$. Furthermore, and irrespective of $|P|$, as, for every $p \in P$, it is indistinguishable whether any $q \in \widetilde{P} \setminus P$ is faulty in $\sigma$ or not, $p$ has to assume that every process $q \in \widetilde{P} \setminus P$ is faulty. Hence, for every $p \in P$, we must eventually have $out(p, r_i) \subseteq P$ for some sufficiently large $r_i$.

We now construct a set $S$ of $k + 1$ processes that violates Definition 9.3.9: fix some $P \in \widetilde{P}$ with $|P| > 1$ and add the two processes $p, p' \in P$, as described above, to $S$. For

every partition $P_j \in \widetilde{P} \setminus P$, add one process $p_i$ from $P_j$ to $S$. Since there exist $r, r'$ such that $out(p, r) \cap out(p', r') = \emptyset$, and $\forall P_j \in \widetilde{P} \setminus P, \forall p \in P_j, \exists r_i \colon out(p_i, r_i) \subseteq P_i$ and, by the construction of $S$, we have that $\forall p, q \in S$, $\exists r_i, r_j$ such that $out(p, r_i) \cap out(q, r_j) = \emptyset$. This set $S$ clearly violates Definition 9.3.9, as required. □

**Lemma 9.3.12.** *For $k > 1$, $\mathcal{AMP}_{n,n-1}[fd : \Omega_k]$ cannot be implemented in $\mathcal{SMP}_n[adv : \Diamond\mathsf{STABLE}_n(k, \infty)]$.*

*Proof.* We prove the claim for $k = 2$ and $n = 3$, as it is straightforward to derive the general case from this. We show that supposing some algorithm could implement $\Omega_k$ under $\Diamond\mathsf{STABLE}_n(k, \infty)$ leads to a contradiction. The following graph sequences (a)–(e) are all admissible sequences under $\Diamond\mathsf{STABLE}_n(k, \infty)$ (we assume that processes not appearing in the sequences are isolated):

(a) $(p_3 \leftarrow p_1 \rightarrow p_2)_{r>0}$

(b) $(p_3 \leftarrow p_2 \rightarrow p_1)_{r>0}$

(c) $(p_2 \leftarrow p_3 \rightarrow p_1)_{r>0}$

(d) $(p_1 \rightarrow p_2)_{r>0}$

(e) $(p_1 \rightarrow p_3)_{r>0}$

Let $\varepsilon_a, \ldots, \varepsilon_e$ be the runs resulting from the above sequences applied to the same initial configuration. By Definitions 9.3.8 and 9.3.10, $LD$ has to include $p_1$ in $\varepsilon_a$, $p_2$ in $\varepsilon_b$, and $p_3$ in $\varepsilon_c$. By Definition 9.3.8, in $\varepsilon_d$, because $\varepsilon_a \sim_{p_1} \varepsilon_d$ and $\varepsilon_c \sim_{p_3} \varepsilon_d$ in all rounds, for some $t > 0$, for all $t' > t$, $out(p_1, t') = \{p_1, p_3\}$. A similar argument shows that in $\varepsilon_e$, for some $t > 0$, for all $t' > t$, $out(p_1, t') = \{p_1, p_2\}$, because $\varepsilon_a \sim_{p_1} \varepsilon_e$ and $\varepsilon_b \sim_{p_2} \varepsilon_e$. The indistinguishability $\varepsilon_d \sim_{p_1} \varepsilon_e$ provides the required contradiction, as for some $t > 0$, for all $t' > t$, $out(p_1, t')$ should be the same in $\varepsilon_d$ and $\varepsilon_e$. □

Lemma 9.3.11 may come as a surprise, since the proof of the necessity of $\Sigma_k$ for $k$-set agreement (hence the necessity of $\Sigma = \Sigma_1$ for consensus) developed by Raynal et. al. [BR11] only relies on the availability of a correct $k$-set agreement algorithm. However, their reduction proof works only in $\mathcal{AMP}_{n,n-1}$, i.e., crash-prone asynchronous message passing systems: It relies crucially on the fact that there cannot be a safety violation (i.e., a decision on a value that eventually leads to a violation of $k$-agreement) in any finite prefix of a run. This is not the case in the simulation running atop of $\mathcal{SMP}_{n,0}[adv : \Diamond\mathsf{STABLE}_{n,D}(\infty)]$, however, as we cannot ensure the crash failure semantics of faulty processes (that is needed for ensuring safety in arbitrary prefixes) here. Hence, we cannot apply their result (or adapt their proof) in our setting.

From these negative results, we conclude that, given $\mathcal{SMP}$ with some message adversary, looking out for simulations of $\mathcal{AMP}_{n,n-1}[fd : (\Sigma, \Omega)]$ in order to be able to run standard

failure detector-based consensus algorithms is not a viable alternative to the development of a tailored consensus algorithm, and is hence also no substitute for the chase for strongest message adversaries in $\mathcal{SMP}$. We hence need a different way for approaching the latter, which will be presented in the following section.

## 9.4 Message Adversary Simulations and the Strongest Message Adversary for Consensus

The lemmas in the previous section showed that there exists a message adversary $\mathsf{MA}$ that makes consensus solvable but despite this, $\mathcal{AMP}_{n,n-1}[fd : \Sigma, \Omega]$ cannot be simulated atop of $\mathcal{SMP}_n[adv : \mathsf{MA}]$. Even though failure detectors cannot hence be used directly to find a strongest message adversary, the concept of comparing models with different restrictions in terms of their computational power is nevertheless attractive. This idea was already used in [CBS09] to structure communication predicates in the HO model, even though the "general translations" introduced for this purpose suffered from the fact that one would need to solve consensus repeatedly. In sharp contrast, the message adversary simulations introduced below only need to run several instances of (one-shot) consensus in parallel.

Our counterpart to the failure detector simulation is the *message adversary simulation*, of message adversary $\mathsf{MA}$ atop of $\mathsf{MA}'$, using a suitable simulation algorithm $\mathcal{X}$ running in $\mathcal{SMP}_n[adv : \mathsf{MA}']$ that emulates $\mathcal{SMP}_n[adv : \mathsf{MA}]$. Note that $\mathcal{X}$ is parameterized by the algorithm that is to be run in $\mathcal{SMP}_n[adv : \mathsf{MA}]$, as $\mathcal{X}$ has to know its message sending and state-transition function. If such a simulation exists, then $\mathsf{MA}'$ has at least the computational power of $\mathsf{MA}$, i.e., all problems solvable in $\mathsf{MA}$ are also solvable in $\mathsf{MA}'$. We will now describe the details of our message adversary simulation. First, we perform some minor adaptations to make the notion of HO predicate emulation from [CBS09] compatible with the message adversary model. Based on this, we introduce our new notion of a message adversary simulation.

Consider $\mathcal{SMP}_n[adv : \mathsf{MA}']$, and let $\mathcal{X}$ be a still to-be-defined algorithm that maintains a variable $\mathsf{NewIn}_p \subseteq \Pi$ at every process $p$, along with the set of received messages. For some positive integer $k$, let the macro-round $\rho \geqslant 1$ for process $p$ be the sequence of the $k$ consecutive rounds $r_1 = k(\rho-1)+1, \ldots, r_k = k\rho$. Note that $k = k(p,\rho)$ may be different for different (receiver) processes $p$ and macro rounds $\rho$ here. We say that $\mathcal{X}$ *emulates* (macro-)rounds $\rho \in \{1, 2, \ldots\}$ of $\mathcal{SMP}_n[adv : \mathsf{MA}]$, if, in any run of the latter, the value of $\mathsf{NewIn}_p^{(\rho)}$ computed at the end of macro-round $\rho$ satisfies:

(E1) If $q \in \mathsf{NewIn}_p^{(\rho)}$ then $q \in \mathrm{CP}_p^{r_k}(r_1 - 1)$

(E2) The communication pattern $(G_{(\rho)})_{\rho>0}$ with $G_{(\rho)} = \langle \Pi, E(\rho) \rangle$ and $E(\rho) = \bigcup_{p\in\Pi} \bigcup_{q\in\mathsf{NewIn}_p^{(\rho)}} (q,p)$ satisfies $(G_{(\rho)})_{\rho>0} \in \mathsf{MA}$.

Property (E1) guarantees that $p$ has $q$ as an in-neighbor in the emulation only if $p$ was influenced by $q$ during $\rho$ and (E2) guarantees that the resulting collection of $\mathsf{NewIn}_p^{(\rho)}$ sets satisfy $\mathsf{MA}$. Together, they ensure well-defined and correct emulations, respectively.

Implementing the above emulation, i.e., the emulation algorithm $\mathcal{X}$, is straightforward: Let $m^r_{q'\to p'}$ represent the message sent by $q'$ to $p'$ in round $r$ in $\mathcal{SMP}_n[adv : \mathsf{MA}']$, and $m^{(\rho)}_{q\to p}$ the message sent in macro-round $\rho$ in the simulated $\mathcal{SMP}_n[adv : M]$. In $\mathcal{X}$, each $q'$ piggy-backs every $m^{(\rho)}_{q\to p}$ it knows of (either because $q' = q$ or else because it has received some $m^{r'}_{x\to q'}$ that contained $m^{(\rho)}_{q\to p}$) on all outgoing messages $m^j_{q\to p'}$, for every $p' \in \Pi$, $j \in [(\rho-1)k+1, \rho k]$, and delivers $m^{(\rho)}_{q\to p}$ in $z_p^{(\rho)}$ in macro-round $\rho$, along with maintaining $\mathsf{NewIn}_p^{(\rho)}$. Note that the message chain stipulated in (E1) ensures that the latter delivery will happen in some round.

Unfortunately, however, this emulation is too restrictive for our purpose.

Our next step will hence be to define a more abstract *message adversary simulation* of $\mathcal{SMP}_n[adv : \mathsf{MA}]$, by relaxing (E1) in a way that still guarantees well-defined simulations. We recall that, by definition, $q \in \mathrm{In}(p, r)$ iff $m^r_{q\to p} \in z^r_p$, and that $m^r_{q\to p} = S^r_q((q, r-1), p)$. Now consider the following relaxed variant of (E1), where we replace the requirement of $p$ having *received* the message $m^r_{q\to p}$ by the requirement of $q$ having attempted to *send* $m^r_{q\to p}$ and $p$ being able to derive this fact:

(E1') If $q \in \mathsf{NewIn}_p^{(\rho)}$, then there exists at least one $j$, $(\rho-1)k+1 \leqslant j \leqslant \rho k$, for which $p$ can infer the messages $S^j_q((q, j-1), p)$ for all $q \in \Pi$.

We say that $\mathcal{X}$ is a message adversary simulation of (macro-)rounds $\rho > 0$ of $\mathcal{SMP}_n[adv : \mathsf{MA}]$, if, in any run of the latter, the value of $\mathsf{NewIn}_p^{(\rho)}$, computed at the end of macro-round $\rho$ satisfies (E1') and (E2). At the first glance, (E1') may appear to be equivalent to (E1), as it has the same outcome in the case where a chain of messages from $q$ to $p$, as specified in (E1), exists. However, the essential difference is revealed only in the case where such a chain does not exist, but where it is still possible for process $p$ to locally simulate the execution of an arbitrary algorithm in $\mathcal{SMP}_n[adv : \mathsf{MA}']$ at process $q$. In this case, $p$ is able to locally compute the messages $S^j_q((q, j-1), p)$, i.e., the messages that $q$ intended to send in macro round $\rho$ without any actual communication in this macro round!

Using this type of message adversary simulations, we will prove in Lemma 9.4.3 below that consensus solvability and the ability to simulate the message adversary $\mathsf{STAR}$, which contains all perpetual repetitions of star-graphs, are equivalent.

**Definition 9.4.1.** *For $k \in [1, n]$, the message adversary $\mathsf{STAR}(k)$ contains all sequences $(G_r)_{r>0}$ for which there is a $k' \in [1, k]$ and a partitioning $P_1 \cup \ldots \cup P_{k'} = \Pi$ such that for all rounds $r$ and all $i \in [1, k']$, the induced graph $G_r[P_i]$ is a star graph with self-loops and there are no edges between nodes of distinct partitions in $G_r$. Let $\mathsf{STAR} := \mathsf{STAR}(1)$.*

**Algorithm 9.1:** Multi-valued consensus for initial value $x_i$ and decision variable $y_i$ using a binary consensus algorithm $\mathcal{C}$. Code for process $p_i$.

```
1  Initially, let r := 1, K := ∅, X_i[i] := {x_i}, X_i[j] := ⊥ for all j ∈ [1, n] \ {i}, and
2  D[j] := ⊥ for all j ∈ [0, 2^{n−1}]
   Loop over rounds 1, 2, . . . :
3  send X_i to all
4  receive X_j from all p_j ∈ In_{p_i}(G_r)
5  foreach X_j received from p_j do
6  |  foreach k with X_i[k] = ⊥ and X_j[k] ≠ ⊥ do
7  |  |  X_i[k] ← X_j[k]            /* maintain array of known input values */
8  for j = 0 to 2^n − 1 do
9  |  Let z be the bit at position i of the n-bit binary representation of j
10 |  C_j(in: z, out: D[j])          /* perform round r of consensus instance j */
11 if for all j ∈ [0, 2^n − 1] : D[j] ≠ ⊥ then
12 |  foreach j, k ∈ [0, 2^n − 1] whose n-bit binary representations differ in exactly one bit
   |  (at position ℓ) and D[j] ≠ D[k] do
13 |  |  K ← K ∪ {ℓ}
14 |  c ← max(K) + 1
15 |  y_i ← X[c]
```

Before proceeding with Lemma 9.4.3, we show that solvability of binary consensus and solvability of multi-valued consensus are equivalent in any model $\mathcal{SMP}_n[adv : \mathsf{MA}]$. For this reason, we introduce Algorithm 9.1, which, given a binary consensus algorithm, solves consensus for an arbitrary input domain $V$. While there is a classic result that proves this equivalence in a Byzantine setting [TC84], to the best of our knowledge, it was never formally established in a message adversary setting before.

Algorithm 9.1 roughly operates as follows (see Lemma 9.4.2 for the correctness proof): Each process attempts to flood its input value $x_i \in V$, i.e., keeps track of all the input values it has received so far in an array $\mathsf{X}_i$ and broadcasts its array in every round (Lines 3 to 7). Furthermore, the processes execute $2^n$ instances of binary consensus in parallel, one for every possible $n$-bit binary string representing the binary input values of the processes (Lines 8 to 10). Once all these instances have terminated at a process, it checks which alternation of a single process' input value caused a different decision value, and records the identifier of the according process in the set $\mathsf{K}$ (Lines 12 and 13). As we will prove in Lemma 9.4.2, $\mathsf{K}$ holds a non-empty subset of the kernel of the communication graph sequence, i.e., a subset of $\Pi$ whose input value was received by every process. Finally, each process decides on the input of the process with the maximum identifier from the set $\mathsf{K}$ (Lines 14 and 15).

**Lemma 9.4.2.** *Binary consensus can be solved in a model $\mathcal{SMP}_n[adv : \mathsf{MA}]$ if and only if multi-valued consensus can be solved in $\mathcal{SMP}_n[adv : \mathsf{MA}]$.*

*Proof.* Since the solvability of multi-valued consensus immediately implies the solvability

of binary consensus, we only need to prove the converse direction. Assume that $\mathcal{C}$ is a binary consensus algorithm for $\mathcal{SMP}_n[adv : \mathsf{MA}]$. We show that Algorithm 9.1 solves consensus under $\mathcal{SMP}_n[adv : \mathsf{MA}]$ for an arbitrary input domain. It is apparent immediately from the code that the algorithm maintains all known input values in the array $\mathsf{X}_i$ at every process $p_i$ (Lines 3 to 7), runs $2^n$ parallel binary consensus instances to check which changes of a single input value causes different decisions (Lines 8 to 13), and finally decides on the input value of the process with the largest identifier whose change of inputs caused a change of the decision (Lines 14 and 15). Since we assume that $\mathcal{C}$ solves binary consensus, it follows that processes have agreement on $\mathsf{K}$ and hence $\mathsf{c}$ when executing Line 15. It thus suffices to show that, when executing Line 15, (i) $\mathsf{K} \neq \emptyset$ and (ii) $\mathsf{X}_i[\mathsf{c}] = x_{\mathsf{c}}$.

Property (i) follows, since the decision of the instance $\mathcal{C}_0$ must be $D[0] = 0$ while the decision of $\mathcal{C}_{2^n-1}$ must be $D[2^n - 1] = 1$ by the validity property of consensus. Changing one input at a time from 0 to 1 when starting from the input of instance $\mathcal{C}_0$ shows that there exist two instances $\mathcal{C}_a, \mathcal{C}_b$ with different decisions $D[a] \neq D[b]$ that differ only in the input of a single process. It follows that $\mathsf{K} \neq \emptyset$. Note that the above reasoning, which is solely based on the initial states of the parallel consensus instances, is justified, since all these instances are subjected to the *same* communication pattern in the given run.

For property (ii), let $t$ be the round where process $p_i$ executes Line 15 and pick an arbitrary $p_j \in \mathsf{K}$. If $p_j \in \mathrm{CP}^t_{p_i}(0)$, the claim follows since the processes store and forward their input values in every round. Thus, suppose that $p_j \notin \mathrm{CP}^t_{p_i}(0)$. As we have elaborated above, $p_j \in \mathsf{K}$ implies that there are two instances $\mathcal{C}_a$, $\mathcal{C}_b$ that differ only in the input $x_j$ of $p_j$ but their decision values are different. Because we supposed that $p_j \notin \mathrm{CP}^t_{p_i}(0)$, process $p_i$ did not learn $x_j$ by time $t$. By construction, for $p_i$, $\mathcal{C}_a$ are $\mathcal{C}_b$ indistinguishable up to round $t$, i.e., the state $(p_i, t)$ is the same in both instances. But then $p_i$ decided the same in $\mathcal{C}_a$ and $\mathcal{C}_b$ by round $t$, which contradicts that different decisions are reached in $\mathcal{C}_a$ and $\mathcal{C}_b$. □

We are now ready to show that consensus solvability and the ability to perform a message adversary simulation of the message adversary STAR are equivalent. In order to do so, we introduce Algorithm 9.2, which performs the desired message adversary simulation: It constructs the set $\mathsf{NewIn}^{(\rho)}_{p_i}$, corresponding to STAR, for every macro round $\rho$ in accordance with the definitions (E1') and (E2), given an arbitrary STAR-based algorithm (specified by its transition functions $T$ and message sending functions $S$) and an arbitrary initial state. Algorithm 9.2 does this in two steps: First, it executes the multi-valued consensus algorithm and uses the given initial state as input value. Upon termination of the consensus, it commences with the computation of the macro rounds, using the decided-upon initial state as the initial state of the center process of the star graph for STAR.

**Lemma 9.4.3.** *The following are equivalent:*

*(1) There is an algorithm that solves (binary) consensus in $\mathcal{SMP}_n[adv : \mathsf{MA}]$.*

**Algorithm 9.2:** Simulating STAR for an algorithm with initial state $(p, 0)$, transition/message sending functions $S, T$, on top of a message adversary where a multi-valued consensus algorithm $\mathcal{M}$ exists; code for process $p_i$.

```
1  Initially, let ρ := 0, cenState := ⊥, myState := (p_i, 0)
   Loop over rounds 1, 2, ...:
2  if ρ = 0 then
3  |  M(in: myState, out: cenState)              /* next round of consensus */
4  |  if cenState ≠ ⊥ then
5  |  |  c ← the identifier of the process whose state is stored in cenState
6  |  |  ρ ← 1
7  if ρ > 0 then
8  |  Let z be an n-dimensional vector with z_j = null for all j ∈ [1, n]
9  |  z_c ← S^ρ_{p_c}(cenState, p_c)
10 |  cenState ← T^ρ_{p_c}(cenState, z)
11 |  if c ≠ i then
12 |  |  z_i ← S^ρ_{p_i}(myState, p_i)
13 |  |  myState ← T^ρ_{p_i}(myState, z)
14 |  NewIn^(ρ)_{p_i} ← {p_i, p_c}
15 |  ρ ← ρ + 1
```

*(2) A message adversary simulation for $\mathcal{SMP}_n[adv : \mathsf{STAR}]$ on top of $\mathcal{SMP}_n[adv : \mathsf{MA}]$ exists.*

*Proof.* The direction (2) ⇒ (1) follows since the following simple algorithm solves consensus in $\mathcal{SMP}_n[adv : \mathsf{STAR}]$: Every process $p$ attempts to broadcast its input value $x_p$ and then decides on the input value $x_q$ received in the first round from a process $q \neq p$. If no $x_q$ was received in the first round for any $q \neq p$, $p$ decides on its own input $x_p$. We can run this algorithm in the simulated $\mathcal{SMP}_n[adv : \mathsf{STAR}]$ provided by (2) to obtain a consensus algorithm for $\mathcal{SMP}_n[adv : \mathsf{MA}']$.

To show the direction (1) ⇒ (2), we show that Algorithm 9.2 is a message adversary simulation of the model $\mathcal{SMP}_n[adv : \mathsf{STAR}]$ on top of $\mathcal{SMP}_n[adv : \mathsf{MA}]$ for any given algorithm $\mathcal{A}$, specified by its transition functions $S, T$ and its initial state. Since consensus is solvable in $\mathcal{SMP}_n[adv : \mathsf{MA}]$ by assumption, there is a multi-valued consensus algorithm $\mathcal{M}$ for $\mathcal{SMP}_n[adv : \mathsf{MA}]$ by Lemma 9.4.2.

Property (E2) follows since $\mathcal{M}$ satisfies agreement, thus all processes choose the same cenState and thus the same identifier c in Line 5. It follows that for every macro round $\rho$, $G_{(\rho)} = \langle \Pi, E(\rho) \rangle$ with $E(\rho) = \bigcup_{p \in \Pi} \bigcup_{q \in \mathsf{NewIn}_p^{(\rho)}} (q, p)$ being the star graph with self-loops and center node $p_c$. Thus, $(G_{(1)}, G_{(2)}, \ldots) \in \mathsf{STAR}$, as required.

Property (E1') follows by induction on $\rho$. The induction hypothesis is that at an arbitrary process $p_i$, at the end of macro-round $\rho$, cenState actually holds the state $(p_c, \rho)$ (that results when running algorithm $\mathcal{A}$ in $\mathcal{SMP}_n[adv : \mathsf{STAR}]$, starting from an initial

configuration where the initial state of process $p_{\mathsf{C}}$ is $(p_{\mathsf{C}}, 0)$). Additionally, myState holds the state $(p_i, \rho)$ (that results from running $\mathcal{A}$ in $\mathcal{SMP}_n[adv : \mathsf{STAR}]$, starting from an initial configuration where the initial state of $p_{\mathsf{C}}$ is $(p_{\mathsf{C}}, 0)$ and the initial state of $p_i$ is $(p_i, 0)$). These two statements imply (E1') since, by assumption, $p_i$ knows the algorithm $\mathcal{A}$ and thus the message sending functions $S_{p_i}$ and $S_{p_{\mathsf{C}}}$.

For the base case, we consider the "virtual" macro round $\rho = 0$ that ends after process $p_i$ has just passed Line 5 for the first time. Here, the hypothesis holds since the consensus algorithm $\mathcal{M}$ returned the initial state $(p_{\mathsf{C}}, 0)$ of $p_{\mathsf{C}}$, thus cenState $= (p_{\mathsf{C}}, 0)$ and, due to the initial assignment in Line 1, myState $= (p_i, 0)$.

For $\rho \geqslant 1$, we consider the end of macro round $\rho$, i.e., we assume that $p_i$ just passed Line 14. The induction hypothesis ensures that, at the end of round $\rho - 1$, cenState holds $(p_{\mathsf{C}}, \rho - 1)$ and myState holds $(p_i, \rho - 1)$. After executing Line 9, $\mathbf{z}_{\mathsf{C}}$ thus holds the message $p_{\mathsf{C}}$ sends to itself in macro round $\rho$, by definition of the message sending function $S$. Because of this and since all other components $\mathbf{z}$ were set to *null* in Line 8, the definition of the state-transition function $T$ ensures that, after executing Line 10, cenState holds the state $(p_{\mathsf{C}}, \rho)$. A very similar argument shows that myState holds $(p_i, \rho - 1)$.

□

With these preparations, we will now define and discuss our notion of a *strongest message adversary*:

**Definition 9.4.4** (Strongest message adversary)**.** *A message adversary* MA *is a strongest message adversary for a problem* $\mathcal{P}$*, if* $\mathcal{P}$ *is solvable in* $\mathcal{SMP}_n[adv : \mathsf{MA}]$ *and there is a message adversary simulation for* $\mathcal{SMP}_n[adv : \mathsf{MA}]$ *atop of every* $\mathcal{SMP}_n[adv : \mathsf{MA}']$ *in which* $\mathcal{P}$ *is solvable.*

A property that follows directly from Definition 9.4.4 is:

**Corollary 9.4.5.** *If a strongest message adversary for consensus allows to solve some problem* $\mathcal{P}$*, it holds that every message adversary that allows to solve consensus also allows to solve* $\mathcal{P}$*.*

By Lemma 9.4.3, STAR is a strongest message adversary for consensus. Since every other message adversary that contains STAR and allows to solve consensus is also a strongest message adversary by definition, we finally obtain the following Corollary 9.4.6:

**Corollary 9.4.6** (Class of strongest message adversaries for consensus)**.** *Every message adversary that includes* STAR *and allows to solve consensus is a strongest message adversary for consensus.*

Examples for such message adversaries are (SOURCE, QUORUM), $\Diamond\mathsf{STABLE}_{n,D}(\infty)$, and STAR itself: It is not hard to see that they all contain STAR. Furthermore, the solvability

of consensus in $\mathcal{SMP}_n[adv : (\mathsf{SOURCE}, \mathsf{QUORUM})]$ follows since it was shown in [RS13] that it is task equivalent to $\mathcal{AMP}_{n,n-1}[fd : \Sigma, \Omega]$, the solvability in $\mathcal{SMP}_n[adv : \Diamond\mathsf{STABLE}_n(k, \infty)]$ was provided in Section 9.2, and the solvability in $\mathcal{SMP}_n[adv : \mathsf{STAR}]$ was shown in Lemma 9.4.3.

Interestingly, the findings above can easily be adapted for $k$-set agreement as well, as Algorithm 9.2 can be used to simulate any perpetually repeated graph that consists of $k$ star graphs, atop of a message adversary $\mathsf{MA}'$ that allows to solve $k$-set agreement: As any $k$-set agreement algorithm guarantees at most $k$ different decision values, Algorithm 9.2 indeed allows to simulate any perpetually repeated graph that consists of at most $k$ star graphs, with the $k$ decisions as the centers. Hence:

**Corollary 9.4.7** (Class of strongest message adversaries for $k$-set agreement)**.** *Every message adversary that includes* $\mathsf{STAR}(k)$ *and allows to solve $k$-set agreement, is a strongest message adversary for $k$-set agreement.*

Examples for strongest message adversaries for $k$-set agreement are $\Diamond\mathsf{STABLE}_n(k, \infty)$ and the message adversary $\mathrm{VSSC}_{D,H}(n, \infty) + \mathrm{MAJINF}(k)$ introduced in [BRS+18].

## 9.5 Discussion

The results of Sections 9.3 and 9.4, in particular Lemma 9.3.7 and Corollary 9.4.6, let us obtain the following insights:

(i) Since $\Diamond\mathsf{STABLE}_{n,D}(\infty)$ does not allow to implement $\Sigma$, we cannot hope to run $(\Sigma, \Omega)$-based consensus algorithms on top if it.

(ii) There are message adversaries that are incomparable in terms of graph sequence inclusion, even though they belong to the class of strongest message adversaries, e.g. $(\mathsf{SOURCE}, \mathsf{QUORUM})$ and $\Diamond\mathsf{STABLE}_{n,D}(\infty)$: They are incompatible w.r.t. graph sequence inclusion as $(\mathsf{SOURCE}, \mathsf{QUORUM})$ contains the graph sequence consisting of two processes $p, q$ with self-loops where the edge $(p, q)$ is always there and the edge $(q, p)$ appears only in every odd round but this sequence is not in $\Diamond\mathsf{STABLE}_{n,D}(\infty)$. Conversely, a perpetually repeated, unidirectional chain of $n > 2$ processes is in $\Diamond\mathsf{STABLE}_{n,D}(\infty)$ but not in $(\mathsf{SOURCE}, \mathsf{QUORUM})$.

(iii) There are message adversaries like the one introduced in [SWS16], which (unlike $\Diamond\mathsf{STABLE}_{n,D}(\infty)$) do not even guarantee a single strongly correct process in some runs. Implementing $\Sigma$ subject to Definition 9.3.4 atop of such message adversaries is trivially impossible, as its specification becomes void.

On the other hand, the results of Section 9.4, in particular, Lemma 9.4.3, reveals that it is possible to simulate the message adversary $\mathsf{STAR}$ atop of any message adversary (hence also $\Diamond\mathsf{STABLE}_{n,D}(\infty)$) that allows to solve consensus. However, it is straightforward to

simulate $(\Sigma, \Omega)$ in $\mathcal{AMP}$ in $\mathcal{SMP}_n[adv : \mathsf{STAR}]$: Initially, process $p_i$ outputs itself as the leader and $\Pi$ as the quorum. At the end of macro-round 1, both the leader and the quorum is set to $\{p_i\}$ if $|\mathsf{NewIn}_{p_i}^{(1)}| = 1$ and $\mathsf{NewIn}_{p_i}^{(1)} \setminus \{p_i\}$ otherwise. Therefore, we seem to have arrived at a contradiction of Lemma 9.3.7!

This seemingly paradoxical result is traceable to the fact that the set $\mathsf{NewIn}_{p_i}^{(1)}$ provided by the message adversary simulation of $\mathsf{STAR}$ need *not* contain a strongly correct process! Indeed, recall that the infinite repetition of $G$ can also be achieved by letting every process $p_i$ in the system *locally* simulate the behavior of some $p_{\mathsf{C}}$'s algorithm. This is possible, since $p_i$ knows both $p_{\mathsf{C}}$'s deterministic algorithm and its initial state, from the star graph in macro-round 1.

Hence, it finally turns out that the impossibility of implementing $\Sigma$ established in Lemma 9.3.7 depends crucially on the assumption of considering strongly correct processes as correct in the simulated $\mathcal{AMP}$. In principle, it might be possible to implement $\Sigma$ (and also $\Omega$) atop of any message adversary that allows to solve consensus if a weaker alternative of Definition 9.3.1 of correct processes in $\mathcal{AMP}$, was used. A candidate for such a definition would be to equate the correct processes to the processes in the kernel of a given sequence, i.e., to those processes that manage to influence everyone eventually, as given in Definition 9.5.1 below.

**Definition 9.5.1** (Faulty and strongly correct processes redefined)**.** *Given an infinite communication pattern $\sigma$, process $p$ is* faulty *in a run with $\sigma$ if there is a process $q$, s.t., for all $r > 0$: $p \notin \mathrm{CP}_q^r(0)$.*
*Let $\mathcal{C}(\sigma) = \left\{p \in \Pi \mid \forall q \in \Pi, \exists r > 0 \colon p \in \mathrm{CP}_q^r(0)\right\}$ denote the strongly correct (= non-faulty) processes in any run with $\sigma$.*

Such a definition should be sufficient to circumvent our impossibility results, e.g., the one from Lemma 9.3.7: Since there is a consensus algorithm $\mathcal{M}$ for $\Diamond\mathsf{STABLE}_{n,D}(\infty)$, we can run $\mathcal{M}$ with input value $x_i = i$ for every process $p_i$ to find a correct process according to Definition 9.5.1. To see this, we could use an argument similar to the proof of Lemma 9.4.2 to show that the decision value $y_i$ in such a run is actually the identifier of a process that is a member of the kernel. Inspired by [CBHW10], however, we conjecture that the inability of correct processes to reach the other processes more often than once in a run would be fundamentally incompatible with the eventually-forever-type specifications of classic failure detectors, and would hence invalidate the wealth of known results.

Chapter 10

# The Topological Space of a Message Adversary

So far in this thesis, we have seen how the solvability of consensus can be characterized in limit-closed message adversaries (Chapter 5) and in a particular non-closed message adversary (Chapter 6). Even though we found a concise condition that was sufficient to solve consensus in Chapter 8, which turned out to be quite useful for developing better algorithms, we were not able to weaken it significantly enough to make it also necessary. This raises the following question: If the central object under investigation are sets of communication patterns in both cases, why does the characterization for non-closed message adversaries seem so elusive? In this section we will answer precisely this question.

Our strategy will be to directly investigate the space of the communication patterns (actually, the corresponding process-time graphs) and define a suitable topological structure for this space, which will allow us to make statements about consensus solvability. We will see that the topological space of a limit-closed message adversary is compact (leading us to call them compact message adversaries from now on) and thus amenable to the combinatorial characterization in Chapter 5. We will further find that non-compact message adversaries have a much less benign topological structure, which makes their combinatorial analysis challenging. Still, we manage to give a topological characterization theorem for all message adversaries: We show that the space of the admissible process-time graphs must constitute a separation that respects the validity condition in order for consensus to become solvable.

Rather than utilizing classic distributed computing techniques, we employ a novel modeling and analysis approach based on point-set topology [AS85]. We had to add several new topological ideas, as detailed below, which at the end provided us with a very powerful "toolbox" that, for example, allowed us to provide a topological explanation of bivalence [FLP85] and bipotence [MR02] proofs. As our approach can be adapted to different distributed computing models, we believe that it is of independent interest.

(i) We define two new topologies on the execution space, which allow to reason about sequences of local views of a certain process, rather than about configuration sequences. If this process is a fixed one ($p$), the resulting $\{p\}$-*view topology* is induced by a pseudo-metric $d_{\{p\}}(\alpha, \beta)$ based on the common prefix of $p$'s local views in the executions $\alpha$, $\beta$. Alternatively, this process can be the last one to notice a difference between executions, which gives rise to the *minimum topology* induced by the pseudo-semi-metric $d_{\min}(\alpha, \beta) = \min_{p\in[n]} d_{\{p\}}(\alpha, \beta)$.

(ii) Since the set of possible views is not necessarily finite, the product topologies built on the execution space are not compact *a priori*. Fortunately, the space of sequences of process-time graphs [BZM14], which are finite, comes to our rescue: Since the local transition function $\tau$, which maps sequences of admissible process-time graphs to the corresponding admissible executions $\Gamma$, is continuous, the resulting product subspaces *are* compact.

(iii) We show that consensus can be modeled as a continuous decision function $\Delta$, which maps an admissible execution to its unique decision value. In conjunction with the above results, this allows us to prove that consensus is solvable if and only if all the decision sets, i.e., the pre-images $\Gamma_v = \Delta^{-1}[\{v\}]$ resp. $PS_v = \tau^{-1}[\Delta^{-1}[\{v\}]]$ for every decision value $v$, are separated in our topologies. We also provide a universal consensus algorithm, which relies on this separation. Moreover, we show that separability is equivalent to broadcastability of the connected components (in the sense that there is a process that is heard by all processes in every sequence in a connected component).

(iv) Using some properties of the pseudo-metric $d_{\{p\}}$, we provide a topological definition of fair and unfair sequences [FG11]. They turn out to be the limits of two infinite sequences of executions lying in two different decision sets, which have distance 0, and happen to coincide with the forever bivalent/bipotent executions constructed in bivalence proofs. We show that a message adversary must not include such fair and unfair sequences for consensus to be solvable.

(v) We use our generic results to give a complete characterization of consensus solvability for both compact and non-compact message adversaries. For the former (like [SW89, SWK09, CGP15]), we introduce a simple $\varepsilon$-approximation of a decision set, which is equivalent to the set of the $(\log \frac{1}{\varepsilon})$-prefixes of the sequences contained therein. We prove that consensus can be solved, using our universal algorithm, if and only if the $\varepsilon$-approximations are broadcastable for some $\varepsilon$. For non-compact message adversaries (like [BRS$^+$18, WSS19, FG11]), the $\varepsilon$-approximation does not work, so one needs to apply our universal algorithm based on the connected components directly.

**Chapter organization:** In Section 10.2 and 3.5, we define the elements of the spaces that are endowed with our new topologies in Section 10.3. Section 10.4 introduces the consensus problem in topological terms and provides our abstract characterization result (Theorem 10.4.4, which also provides a universal algorithm). We refine our characterization according to the properties of the minimum topology (Theorem 10.4.3) and the

process-view topology (Corollary 10.4.14). Section 10.5 is devoted to the application of our generic results to compact (Theorem 10.5.1) and non-compact (Theorem 10.5.6) message adversaries.

## 10.1 Related Work

Apart from early work of Nowak on characterizing consensus solvability in standard compact models [Now10], and a topological study of the strongly dependent decision problem [BR19], which both use the classic common prefix metric [AS85], the only distributed computing work we are aware of that utilizes point-set topology is [LM95]. In this paper, Lubitch and Moran introduced a construction for schedulers that lead to compact submodels of classic non-compact distributed computing models (like up to $t$ crash failures). Whereas this greatly simplifies impossibility proofs, it does not lead to a precise characterization of solvability in non-compact models, however. Note that, in a similar spirit, [KRH18] allows to reason, in the setting of combinatorial topology, about non-compact models by considering equivalent affine tasks that are compact.

## 10.2 Notation

In order for us to develop a topological understanding of a message adversary, we require a more expressive notation than in the previous sections. While conceptually compatible to all previous considerations, there are some additional intricacies, which we introduce below.

An algorithm $\mathcal{A}$, an initial configuration $C^0$ of $\mathcal{A}$, and a graph sequence $\mathbf{G} = (G_t)_{t\geqslant 1}$ uniquely determine an *execution* in a round-by-round fashion: During round $t$, every process $p$ updates its previous local state $C_p^{t-1}$ to its new state $C_p^t$ in a deterministic way according to $\mathcal{A}$, based on the messages it received during round $t$. Rounds advance synchronously in a send–receive–compute order. Messages in round $t$ are delivered according to communication graph $G_t$: Process $q$ receives process $p$'s message in round $t$ if and only if $(p, q)$ is an edge of $G_t$.

An algorithm is defined by the local transition function and a set of initial states for every process. Since we are only interested in consensus algorithms, we stipulate that processes have an initial state for every initial value $v$ of the finite input domain $\mathcal{V}_I$. Note carefully that we assume that a process running $\mathcal{A}$ does not know $n$ a priori, and cannot always infer $n$ from the messages it receives in an execution either. By contrast, the message adversary need not be oblivious w.r.t. the algorithm, i.e., it may know $\mathcal{A}$ and choose its graph sequences accordingly.

A *configuration* is a tuple $(C_1, \dots, C_n)$ of process states. Whenever $\mathcal{A}$ is clear from the context, we denote its set of possible configurations by $\mathcal{C}$. $\mathcal{C}^\omega$ denotes the set of all infinite sequences of configurations, $\Gamma \subseteq \mathcal{C}^\omega$ is the set of admissible executions resulting from admissible graph sequences. Executions are represented by Greek letters $\alpha, \beta, \dots$.

## 10.3 Topological Structure

In this section, we will endow the sets from the previous section and the process-time graphs from Section 3.5 with suitable topologies. We first recall briefly the basic topological notions that are needed for our exposition. For a more thorough introduction, however, the reader is advised to refer to a textbook [Mun00].

A topology for a set $X$ is a family $\mathcal{T}$ of subsets of $X$ such that $\emptyset \in \mathcal{T}$, $X \in \mathcal{T}$, and $\mathcal{T}$ contains all arbitrary unions as well as all finite intersections of its members. We call $X$ endowed with $\mathcal{T}$, often written as $(X, \mathcal{T})$, a (topological) space and the members of $\mathcal{T}$ open sets. The complement of an open set is called closed and sets that are both open and closed, such as $\emptyset$ and $X$ itself, are called clopen. A metric on $X$ is a function $d : X \times X \to \mathbb{R}_+$ such that for all $x, y, z \in X$ we have $d(x, y) = 0$ if and only if $x = y$, $d(x, y) = d(y, x)$, and $d(x, z) \leq d(x, y) + d(y, z)$. The topology induced by a metric $d$ is the collection of sets $U$ such that for each $u \in U$, there is an $\varepsilon$-ball $B_\varepsilon(u) = \{v \mid d(u, v) < \varepsilon\}$ with $\varepsilon > 0$ and $u \in B_\varepsilon(u) \subseteq U$.

A function from space $X$ to space $Y$ is continuous if the pre-image of every open set in $Y$ is open in $X$. Given a space $(X, \mathcal{T})$, $Y \subseteq X$ is called a subspace of $X$ if $Y$ is equipped with the subspace topology $\{Y \cap U \mid U \in \mathcal{T}\}$. Given $A \subseteq X$, the closure of $A$ is the intersection of all closed sets containing $A$. For a space $X$, if $A \subseteq X$, we call $x$ a limit point of $A$ if it belongs to the closure of $A \setminus \{x\}$. It can be shown that the closure of $A$ is the union of $A$ with all limit points of $A$. A space $X$ is called compact if every union of open sets that covers $X$ contains a finite union of open sets that covers $X$. Note that our topologies are also sequentially compact, which ensures that every infinite sequence of elements has a convergent subsequence.

In previous work on point-set topology [Now10], the set of configurations $\mathcal{C}$ of some fixed algorithm $\mathcal{A}$ was endowed with the discrete topology, induced by the discrete metric $d_{\max}(C, D) = 1$ if $C \neq D$ and 0 otherwise (for configurations $C, D \in \mathcal{C}$)[1] Moreover, $\mathcal{C}^\omega$ was endowed with the corresponding product topology[2], which happens to be induced by the *common prefix metric*

$$d_{\max}(\alpha, \beta) = 2^{-\inf\{t \geqslant 0 \mid \alpha^t \neq \beta^t\}} \quad . \tag{10.1}$$

Our generalization will focus on the local views of the processes, which are obtained by suitable projection functions: For a tuple $x = (x_1, \ldots, x_n)$ and any $\emptyset \neq P = \{p_1, \ldots, p_k\} \subseteq [n]$, the $P$-projection function is defined as $\pi_P(x) = (x_{p_1}, \ldots, x_{p_k})$ where $p_1 \leqslant \cdots \leqslant p_k$. Similarly, for any infinite sequence $\theta = (\theta^0, \theta^1, \ldots)$ and any $t \geqslant 0$, the $t$-projection function is defined as $\pi^t(\theta) = \theta^t$.

The views of a fixed non-empty subset of the processes in a configuration can be defined as follows:

[1] The notation $d_{\max}$ stems from the fact that it is equal to the maximum of the $P$-pseudo-metrics defined in the next subsection.

[2] The product topology on a product $\Pi_{\iota \in I} X_\iota$ of topological spaces is defined as the coarsest topology such that all projections $\pi_i : \Pi_{\iota \in I} X_\iota \to X_i$ are continuous.

**Definition 10.3.1** (Views)**.** *For any configuration $C \in \mathcal{C}$, the view of the processes in $\emptyset \neq P \subseteq [n]$ is $V_P(C) = \pi_P(C)$. The set of all possible $P$-views is defined as* $\text{Views}_P = \{V_P(C) \mid C \in \mathcal{C}\}$.

### 10.3.1 Process-View Topologies

We will now introduce a topology on the set $\mathcal{C}$ of configurations that relies on the corresponding set of views $\text{Views}_P$ of a set of processes $P$.

**Definition 10.3.2** ($P$-view topology for configurations)**.** *Let $P \subseteq [n]$ be a nonempty set of processes. The* $P$-view topology $\mathcal{T}_P^{\mathcal{C}}$ *on $\mathcal{C}$ is defined as the topology induced by the subbasis $\{V_P^{-1}[V] \mid V \subseteq \text{Views}_P\}$, i.e., is the arbitrary union of finite intersections of the elements $V_P^{-1}[V]$ for an arbitrary set $V \subseteq \text{Views}_P$.*

The topology $\mathcal{T}_P^{\mathcal{C}}$ is induced by the pseudo-metric[3]

$$d_P(C, D) = \begin{cases} 0 & \text{if } V_P(C) = V_P(D) \\ 1 & \text{else } . \end{cases}$$

Note that the topology $\mathcal{T}_P^{\mathcal{C}}$ is much coarser than the discrete topology on $\mathcal{C}$, as it does not distinguish the local states of processes outside of $P$.

The corresponding product topology on the set $\mathcal{C}^\omega$ of sequences of configurations, also denoted *$P$-view topology*, can be defined as follows: The $P$-view topology $\mathcal{T}_P^{\mathcal{C}^\omega}$ on $\mathcal{C}^\omega$ is defined as the topology induced by the subbasis $\{(\pi^t)^{-1}[U] \mid t \geqslant 0, U \in \mathcal{T}_P^{\mathcal{C}}\}$, i.e., is an arbitrary union of finite intersections of the elements $(\pi^t)^{-1}[U]$, where $U$ is an arbitrary open set in $\mathcal{T}_P^{\mathcal{C}}$.

We will show that the $P$-view topology is induced by the following pseudo-metric on $\mathcal{C}^\omega$:

$$d_P(\alpha, \beta) = 2^{-\inf\{t \geqslant 0 \mid V_P(\alpha^t) \neq V_P(\beta^t)\}}.$$

We call this the *$P$-pseudo-metric* on $\mathcal{C}^\omega$. Figure 10.1 shows an example for different sets $P$.

It follows immediately from the definition that, if $\alpha, \beta \in \mathcal{C}^\omega$ satisfy $d_P(\alpha, \beta) < 2^{-t}$, then the processes in $P$ have the same view of the first $t$ configurations in $\alpha$ and $\beta$. Moreover, the $P$-pseudo-metric satisfies the following properties:

**Theorem 10.3.1** (Properties of $P$-pseudo-metric)**.** *The $P$-pseudo-metric $d_P(\alpha, \beta)$ on $\mathcal{C}^\omega$ satisfies*

$$\begin{aligned} d_P(\alpha, \beta) &= d_P(\beta, \alpha) && \textit{(symmetry)}, \\ d_P(\alpha, \gamma) &\leqslant d_P(\alpha, \beta) + d_P(\beta, \gamma) && \textit{(triangle inequality)}, \\ d_P(\alpha, \beta) &\leqslant d_Q(\alpha, \beta) && \textit{(monotonicity for } P \subseteq Q\textit{)}, \\ d_{[n]}(\alpha, \beta) &= d_{\max}(\alpha, \beta) && \textit{(common prefix metric)}. \end{aligned}$$

[3] A pseudo-metric has the same properties as a metric, except that it lacks definiteness, i.e., we can have $d(x, y) = 0$ for $x \neq y$.

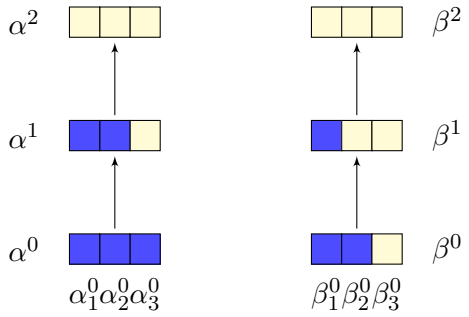


Figure 10.1: Comparison of the $P$-view, minimum, and common prefix topologies. The first three configurations of each of the two executions $\alpha$ and $\beta$ with three processes and two different possible local states (dark blue and light yellow) are depicted. We have $d_{\max}(\alpha,\beta) = d_{\{3\}}(\alpha,\beta) = 1$, $d_{\{2\}}(\alpha,\beta) = 1/2$, and $d_{\min}(\alpha,\beta) = d_{\{1\}}(\alpha,\beta) = 1/4$.

*Proof.* Symmetry follows immediately from the definition. As for the triangle inequality, it is apparent that if $d_P(\alpha,\beta) = 2^{-s}$ and $d_P(\beta,\gamma) = 2^{-t}$, then $d_P(\alpha,\gamma) \leqslant 2^{-\min\{s,t\}} \leqslant 2^{-s} + 2^{-t} = d_P(\alpha,\beta) + d_P(\beta,\gamma)$. Monotonicity for $P \subseteq Q$ and equality with the common prefix metric (10.1) follow easily from the definition. □

Despite of the lack of definiteness, most properties of metric spaces, including compactness, hold also in pseudo-metric spaces [Fre14]. What is obviously lost is the uniqueness of the limit of a convergent sequence of executions, however: if $\alpha_k \to \hat{\alpha}$ and $d_P(\hat{\alpha},\hat{\beta}) = 0$, then $\alpha_k \to \hat{\beta}$ as well.

**Theorem 10.3.2** ($P$-view topology induced by $P$-pseudo-metric)**.** *The $P$-view topology $\mathcal{T}_P^{\mathcal{C}^\omega}$ on $\mathcal{C}^\omega$ is induced by the $P$-pseudo-metric.*

*Proof.* Let $A \subseteq \mathcal{C}^\omega$ be open in the topology induced by the $P$-pseudo-metric. Then for every $\gamma \in \mathcal{C}^\omega$ there is some $\varepsilon(\gamma) > 0$ such that $A = \bigcup_{\gamma\in A} B_{\varepsilon(\gamma)}(\gamma)$. Since this a union, we only need to prove that every $B_{\varepsilon(\gamma)}(\gamma)$ is open in $\mathcal{T}_P^{\mathcal{C}^\omega}$. Choosing $T$ such that $2^{-T-1} < \varepsilon(\gamma) \leqslant 2^{-T}$ and abbreviating $V^t = V_P(\gamma^t) \in \mathrm{Views}_P$, we observe

$$\begin{aligned} B_{\varepsilon(\gamma)}(\gamma) &= \{\delta \in \mathcal{C}^\omega \mid V_P(\delta^t) = V^t \text{ for } 0 \leqslant t \leqslant T\} \\ &= \bigcap_{t=0}^{T} (\pi^t)^{-1}[V_P^{-1}([\{V_s\}])] \ . \end{aligned}$$

The pre-image $V_P^{-1}([\{V^t\}])$ is open in $\mathcal{T}_P^{\mathcal{C}}$ by definition. Hence $B_{\varepsilon(\gamma)}(\gamma)$ is open in $\mathcal{T}_P^{\mathcal{C}^\omega}$ as a finite intersection of subbasis elements for $\mathcal{T}_P^{\mathcal{C}^\omega}$.

Conversely, we need to show that every subbasis element of the form $(\pi^t)^{-1}[U]$ for any $t \geqslant 0$ and $U \in \mathcal{T}_P^{\mathcal{C}}$ is open in the topology induced by the $P$-pseudo-metric. We may write

$$(\pi^t)^{-1}[U] = \bigcup_{\gamma\in(\pi^t)^{-1}[U]} B_{2^{-t}}(\gamma), \tag{10.2}$$

as every $\delta \in B_{2^{-t}}(\gamma)$ satisfies $\pi^t(V_P(\gamma)) = \pi^t(V_P(\delta))$. The set (10.2) is a union of open sets and hence open in the $P$-pseudo-metric. □

We could use exactly the same machinery as above for endowing the set $G$ of communication graphs and the set $\mathcal{G}^\omega$ of sequences of graph sequences, like $\mathcal{C}$ and $\mathcal{C}^\omega$, with an analogous $P$-view topology. It turns out, however, that this does not lead to a transition function that is continuous w.r.t. the $P$-pseudo-metric for all algorithms.

Fortunately, however, the $P$-view projection function $V_P(\cdot)$ is also meaningful for any process-time graph $PT^t$: It just represents the causal past of the processes in $P$ at the end of round $t$, i.e., the sub-graph induced by all process-time nodes $(q, t')$ that have a path to node $(p, t)$ for some $p \in P$ in the process-time graph $PT^t$.

Moreover, we can define a $P$-view topology on $\mathcal{PT}^\omega$, analogous to that on $\mathcal{C}^\omega$. The same is true for the proof of the correspondence Theorem 10.3.2, which hence also holds here.

In sharp contrast to the set of configurations $\mathcal{C}$, however, the set of process-time graphs $\mathcal{PT}^t$ is finite for any time $t$ since $\mathcal{V}_I^n \times G^t$ is finite. Tychonoff's theorem[4] hence implies compactness of the $P$-view topology on $\mathcal{PT}^\omega$, which is not necessarily the case for $\mathcal{C}^\omega$.

Since the algorithms take decisions based on local states only, we can define the *transition function* $\tau : \mathcal{PT}^\omega \to \mathcal{C}^\omega$ that provides $C^t = \tau(PT^t)$ for every $t \geqslant 0$, i.e., maps process-time graphs to the corresponding configurations. The following Lemma 10.3.3 shows that $\tau$ is continuous w.r.t. the $P$-pseudo-metric.

**Lemma 10.3.3** (Continuity of $\tau$ w.r.t. $d_P$)**.** *Let $P \subseteq [n]$. The transition function $\tau : \mathcal{PT}^\omega \to \mathcal{C}^\omega$ is continuous when both $\mathcal{PT}^\omega$ and $\mathcal{C}^\omega$ are endowed with $d_P$.*

*Proof.* Let $U \subseteq \mathcal{C}^\omega$ be open with respect to $d_P$, and let $a \in \tau^{-1}[U]$. Since $U$ is open and $\tau(a) \in U$, there exists some $\varepsilon > 0$ such that $B_\varepsilon(\tau(a)) \subseteq U$. Let $t \in \mathbb{N}$ such that $2^{-t} \leqslant \varepsilon$. We will show that $B_{2^{-t}}(a) \subseteq \tau^{-1}[U]$. For this, it suffices to show that $\tau[B_{2^{-t}}(a)] \subseteq U$. From the definition of the function $\tau$ it follows that $V_P(a^t) = V_P(b^t)$ implies $V_P(\tau(a)^t) = V_P(\tau(b)^t)$. But this means that

$$\tau[B_{2^{-t}}(a)] \subseteq B_{2^{-t}}(\tau(a)) \subseteq B_\varepsilon(\tau(a)) \subseteq U \ ,$$

which proves that $\tau^{-1}[U]$ is open. This completes the proof of continuity of $\tau$ with respect to $d_P$. □

Since the image of a compact space under a continuous function is compact, it hence follows that the set $\tau[\mathcal{PT}^\omega] \subseteq \mathcal{C}^\omega$ of admissible executions is a compact subspace of $(\mathcal{C}^\omega, \mathcal{T}_P^{\mathcal{C}^\omega})$.

[4] Tychonoff's theorem states that any product of compact spaces is compact (with respect to the product topology).

The common structure of $\mathcal{PT}^\omega$ and its image under the transition function $\tau$, implied by the continuity of $\tau$, allows us to reason in either of these spaces. We will usually reason in $\mathcal{PT}^\omega$ or in its subspace $PS$. Note also that any sequence $a \in \mathcal{PT}^\omega$ or $\gamma \in \tau[\mathcal{PT}^\omega]$ can also be identified by specifying a vector $x \in \mathcal{V}_I^n$ of input values and a graph sequence $\mathbf{G} \in \mathcal{G}^\omega$ that leads to $a$ and hence $\gamma$.

### 10.3.2 Minimum Topology

For our characterization of consensus solvability, we endow the set $\mathcal{C}^\omega$ of configuration sequences and the set of process-time graphs $\mathcal{PT}^\omega$ with a slightly different but related topology, induced by the distance function

$$d_{\min}(\alpha, \beta) = \min_{p \in [n]} d_{\{p\}}(\alpha, \beta), \tag{10.3}$$

which reflects the common prefix length of the process $p$ that is the *last* one to distinguish its $q$-view in $\alpha$ and $\beta$.

Note carefully, however, that $d_{\min}$ only satisfies symmetry and nonnegativity but not the triangle inequality, i.e., is only a *pseudo-semi-metric*: There may be sequences with $d_{\{p\}}(\alpha, \beta) = 0$ and $d_{\{q\}}(\beta, \gamma) = 0$ but $d_{\{q\}}(\alpha, \gamma) > 0$ for all $q \in [n]$. Hence, the topology $\mathcal{T}_{\min}^{\mathcal{C}^\omega}$ on $\mathcal{C}^\omega$ induced by $d_{\min}$ lacks many of the properties of (pseudo-)metric spaces, but will turn out to be already sufficient for the characterization of the possibility/impossibility of consensus (see Theorem 10.4.1). Alternatively, however, one can also consider all individual pseudo-metric spaces induced by $d_{\{p\}}$, $p \in [n]$, separately and compute the minimum afterwards.

Pseudo-semi-metrics induce a topology just like (pseudo)-metrics. In fact, a much more general result holds:

**Lemma 10.3.4** (Pseudo-semi-metrics induce topologies)**.** *Let $X$ be a nonempty set and $d : X \times X \to \mathbb{R}$ be a function. Define $\mathcal{T} \subseteq 2^X$ by setting $U \in \mathcal{T}$ if and only if for all $x \in U$ there exists some $\varepsilon > 0$ such that*

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

*Then $\mathcal{T}$ is a topology on $X$.*

*Proof.* Firstly we show that $\mathcal{T}$ is closed under unions. So let $\mathcal{U} \subseteq \mathcal{T}$. We will show that $\bigcup \mathcal{U} \in \mathcal{T}$. Let $x \in \bigcup \mathcal{U}$. Then, by definition of the set union, there exists some $U \in \mathcal{U}$ such that $x \in U$. But since $U \in \mathcal{T}$, there exists some $\varepsilon > 0$ such that

$$B_\varepsilon(x) \subseteq U \subseteq \bigcup \mathcal{U} \ ,$$

which shows that $\bigcup \mathcal{U} \in \mathcal{T}$.

Secondly we show that $\mathcal{T}$ is closed under finite intersections. Let $U_1, U_2, \ldots, U_k \in \mathcal{T}$. We will show that $\bigcap_{\ell=1}^k U_\ell \in \mathcal{T}$. Let $x \in \bigcap_{\ell=1}^k U_\ell$. Then, by definition of the set intersection,

$x \in U_\ell$ for all $1 \leqslant \ell \leqslant k$. Because all $U_\ell$ are in $\mathcal{T}$, there exist $\varepsilon_1, \varepsilon_2, \ldots, \varepsilon_k > 0$ such that $B_{\varepsilon_\ell}(x) \subseteq U_\ell$ for all $1 \leqslant \ell \leqslant k$. If we set $\varepsilon = \min\{\varepsilon_1, \varepsilon_2, \ldots, \varepsilon_k\}$, then $\varepsilon > 0$. Since we have $B_\gamma(x) \subseteq B_\delta(x)$ whenever $\gamma \leqslant \delta$, we also have

$$B_\varepsilon(x) \subseteq B_{\varepsilon_\ell}(x) \subseteq U_\ell$$

for all $1 \leqslant \ell \leqslant k$. But this shows that $B_\varepsilon(x) \subseteq \bigcap_{\ell=1}^k U_\ell$, which means that $\bigcap_{\ell=1}^k U_\ell \in \mathcal{T}$. Since it is easy to check that $\emptyset, X \in \mathcal{T}$ as well, $\mathcal{T}$ is indeed a topology. □

Denote by $\mathcal{T}_{\min}^{\mathcal{C}}$ the topology on the set $\mathcal{C}$ of configurations induced by

$$d_{\min}(C, D) = \min_{p \in [n]} d_{\{p\}}(C, D) \ .$$

Then the function $d_{\min}$ defined on $\mathcal{C}^\omega$ induces the product topology $\mathcal{T}_{\min}^{\mathcal{C}^\omega}$, where every copy of $\mathcal{C}$ is endowed with $\mathcal{T}_{\min}^{\mathcal{C}}$:

**Lemma 10.3.5** (Pseudo-semi-metric for product topologies)**.** *Let $X$ be a nonempty set and let $d : X \times X \to \{0, 1\}$ be a function. Then the product topology of $X^\omega$, where every copy of $X$ is endowed by the topology $\mathcal{T}^\omega$ induced by $d$, is induced by*

$$d^\omega : X^\omega \times X^\omega \to \mathbb{R} \quad , \quad d^\omega(\alpha, \beta) = 2^{-\inf\{t \geqslant 0 \mid d(\alpha^t, \beta^t) > 0\}} \ .$$

*Proof.* We first show that all projections $\pi^t : X^\omega \to X$ are continuous when endowing $X^\omega$ with $\mathcal{T}^\omega$: Let $U \subseteq X$ be open, i.e., $C \in U$ and $d(C, D) = 0$ implies $D \in U$. Let $\alpha \in (\pi^t)^{-1}[U]$. Set $\varepsilon = 2^{-t}$. Then

$$\begin{aligned} B_\varepsilon(\alpha) &= \{\beta \in X^\omega \mid \forall 0 \leqslant s \leqslant t \colon d(\alpha^s, \beta^s) = 0\} \\ &\subseteq \{\beta \in X^\omega \mid d(\alpha^t, \beta^t) = 0\} \\ &= (\pi^t)^{-1}[\{D \in X \mid d(C, D) = 0\}] \subseteq (\pi^t)^{-1}[U], \end{aligned}$$

where the last inclusion follows from the openness of $U$. Since $(\pi^t)^{-1}[U]$ is hence open, the continuity of $\pi_t$ follows.

Let now $\mathcal{T}_0$ be any topology on $X^\omega$ such that all projections $\pi^t$ are continuous. We will show that $\mathcal{T}^\omega \subseteq \mathcal{T}_0$. Let $E \in \mathcal{T}^\omega$ and let $\alpha \in E$. There exists some $\varepsilon > 0$ such that $B_\varepsilon(\alpha) \subseteq E$. Choose $t \in \mathbb{N}_0$ such that $2^{-t} \leqslant \varepsilon$, and set

$$F = \prod_{s=0}^{t} B_1(\alpha^s) \times X^\omega = \bigcap_{s=0}^{t} (\pi^s)^{-1}[B_1(\alpha^s)] \ .$$

Then $F$ is open with respect to $\mathcal{T}_0$ as a finite intersection of open sets. But since $F \subseteq B_\varepsilon(\alpha) \subseteq E$, this shows that $E$ contains a $\mathcal{T}_0$-open neighborhood for each of its points, i.e., $E \in \mathcal{T}_0$. □

Specializing to $X = \mathcal{C}$ and $d = d_{\min}$, we get:

**Lemma 10.3.6** (Minimum topology in terms of process-view topologies)**.** *We have*

$$d^\omega(\alpha,\beta) = \min_{p\in[n]} d_{\{p\}}(\alpha,\beta) = d_{\min}(\alpha,\beta)$$

*for all* $\alpha,\beta \in \mathcal{C}^\omega$.

*Proof.* We calculate

$$\begin{aligned}
-\log_2 d^\omega(\alpha,\beta) &= \inf\{t \geqslant 0 \mid \min_{p\in[n]} d_{\{p\}}(\alpha^t,\beta^t) \neq 0\}\\
&= \inf\{t \geqslant 0 \mid \forall p \in [n]\colon V_{\{p\}}(\alpha^t) \neq V_{\{p\}}(\beta^t)\}\\
&= \inf \bigcap_{p\in[n]} \{t \geqslant 0 \mid V_{\{p\}}(\alpha^t) \neq V_{\{p\}}(\beta^t)\}\\
&= \max_{p\in[n]} \inf\{t \geqslant 0 \mid V_{\{p\}}(\alpha^t) \neq V_{\{p\}}(\beta^t)\}\\
&= \max_{p\in[n]} \inf\{t \geqslant 0 \mid d_{\{p\}}(\alpha^t,\beta^t) \neq 0\}\\
&= \max_{p\in[n]} \big(-\log_2 d_{\{p\}}(\alpha,\beta)\big) \ .
\end{aligned}$$

This concludes the proof. □

Like for $\mathcal{T}_P^{\mathcal{PT}^\omega}$, we get from Tychonoff's theorem and the finiteness, and hence compactness, of every $\mathcal{PT}^t$ that $(\mathcal{PT}^\omega, \mathcal{T}_{\min}^{\mathcal{PT}^\omega})$ is compact.

We finally show that the function $\tau : \mathcal{PT}^\omega \to \mathcal{C}^\omega$ is continuous also for the distance $d_{\min}$.

**Lemma 10.3.7** (Continuity of $\tau$ w.r.t. $d_{\min}$)**.** *The local transition function* $\tau : \mathcal{PT}^\omega \to \mathcal{C}^\omega$ *is continuous when* $\mathcal{PT}^\omega$ *and* $\mathcal{C}^\omega$ *are endowed with* $d_{\min}$.

*Proof.* Continuity with respect to $d_{\min}$ is shown by the same reasoning as in the proof of Lemma 10.3.3, using Lemma 10.3.6 in the last step. □

Like for the $P$-view topologies, the continuity of $\tau$ also implies the compactness of $\tau[\mathcal{PT}^\omega]$ in the minimum-topology.

## 10.4 Consensus

In this section, we will develop a topological condition for consensus under message adversaries. Unlike in [Now10], where the admissible graph sequences needed to be compact in the space induced by the common prefix metric, this is usually not the case here: The set of admissible graph sequences need not be limit-closed in general, see Section 10.5.2.

We will consider the consensus problem, which is defined as follows:

**Definition 10.4.1** (Consensus)**.** *Every process $p \in [n]$ has an input value $x_p \in \mathcal{V}_I$, taken from a finite input domain $\mathcal{V}_I$, which is set in the initial state, and an output value $y_p \in \mathcal{V}_O \cup \{\perp\}$, with a finite output domain $\mathcal{V}_O \supseteq \mathcal{V}_I$, initially $y_p = \perp$. In every admissible execution, a correct consensus algorithms $\mathcal{A}$ must ensure the following properties:*

*(T) Eventually, every $p$ must decide, i.e., change to $y_p \neq \perp$, exactly once (termination).*

*(A) If $p$ and $q$ have decided, then $y_p = y_q$ (agreement).*

*(V) If $x_p = v$ for all $p \in [n]$, then $v$ is the only possible decision value (validity).*

Note that our framework can be easily adapted to different validity conditions, like strong validity, where every decision value must satisfy $y_p = x_q$ for some $q \in [n]$.

If we endow the set $\mathcal{V}_O$ with the discrete topology, it turns out that consensus can be described by a continuous map from the set of admissible executions, endowed with any $P$-view topology or the minimum topology, to $\mathcal{V}_O$.

**Lemma 10.4.2** (Continuity of consensus)**.** *Let $\Gamma \subseteq \mathcal{C}^\omega$ be the set of admissible executions of some consensus algorithm $\mathcal{A}$. Define the map $\Delta : \Gamma \to \mathcal{V}_I$ such that $\Delta(\gamma)$ is the common decision value of algorithm $\mathcal{A}$ in $\gamma$. Then, $\Delta$ is continuous, both with respect to any $P$-view topology and the minimum topology.*

*Proof.* We show that $\Delta$ is locally constant, i.e., for all $\alpha \in \Gamma$, there is some neighborhood $\mathcal{N}$ of $\alpha$ such that $\Delta$ is constant on $\mathcal{N}$. Let $P' = P$ in case of the $P$-view topology and $P' = [n]$ for the minimum topology, and define $t$ to be the latest decision time of the processes in $P'$ in execution $\alpha$. For the $P$-view topology, we choose

$$\begin{aligned}\mathcal{N} = B_{2^{-t}}(\alpha) &= \{\beta \in \Gamma \mid d_P(\alpha, \beta) < 2^{-t}\} \\ &= \{\beta \in \Gamma \mid \forall s \leqslant t \colon V_P(\alpha^s) = V_P(\beta^s)\} \\ &\subseteq \{\beta \in \Gamma \mid V_P(\alpha^t) = V_P(\beta^t)\} \ .\end{aligned}$$

Analogously, for the minimum topology, we use

$$\begin{aligned}\mathcal{N} = B_{2^{-t}}(\alpha) &= \{\beta \in \Gamma \mid \min_{p \in [n]} d_{\{p\}}(\alpha, \beta) < 2^{-t}\} \\ &= \{\beta \in \Gamma \mid \exists p \in [n] \forall s \leqslant t \colon V_{\{p\}}(\alpha^s) = V_{\{p\}}(\beta^s)\} \\ &\subseteq \{\beta \in \Gamma \mid \exists p \in [n] \colon V_{\{p\}}(\alpha^t) = V_{\{p\}}(\beta^t)\} \ .\end{aligned}$$

Since all processes in $P$ have decided the value $\Delta(\alpha)$ in configuration $\alpha^t$ in both cases, they also have decided the same value in configuration $\beta^t$. Hence *all* executions in $\mathcal{N}$ have the decision value $\Delta(\alpha)$ by agreement as asserted. This concludes the proof of Lemma 10.4.2. □

In the following definition, we introduce the sets of process-time graphs and executions that lead to a given decision value:

**Definition 10.4.3** (Decision sets)**.** *For every output value $v \in \mathcal{V}_O$, let $PS(v) = \tau^{-1}[\Delta^{-1}[\{v\}]] \subseteq PS$ and $\Gamma(v) = \Delta^{-1}[\{v\}]$ be the set of admissible process-time graph sequences and admissible executions that lead to a common decision value $v$, respectively.*

We need a few more basic topological terms: A set in a topological space is *clopen*, if it is both closed and open. A topological space is *disconnected*, if it contains a nontrivial clopen set, which means that it it can be partitioned into two disjoint open sets. It is *connected*, if it is not disconnected.

With these preparations, we can already provide a topological consensus impossibility result:

**Theorem 10.4.1** (Consensus impossibility)**.** *If an algorithm solves consensus, then all of its decision sets $\Gamma(v) = \Delta^{-1}[\{v\}]$, $v \in \mathcal{V}_O$, and $PS(v) = \tau^{-1}[\Delta^{-1}[\{v\}]]$ are clopen in the subspace topology of $\Gamma$ and $PS$, respectively, both w.r.t. any $d_P$ and $d_{\min}$.*

*In particular, consensus is impossible if the set $\Gamma$ of admissible executions or the set $PS$ of admissible process-time graphs is connected.*

*Proof.* By Lemma 10.4.2, every decision set $\Gamma(v)$ is closed since $\{v\}$ is closed in the discrete topology. But this means that $\Gamma(v)$'s complement

$$\Gamma \setminus \Gamma(v) = \bigcup_{w \neq v} \Gamma(w)$$

is closed as a finite union of closed sets. Hence, $\Gamma(v)$ is open. This carries over to $PS(v)$, since $\Gamma(v) = \tau[PS(v)]$ and $\tau$ is continuous by Lemma 10.3.3 (for $d_P$) and Lemma 10.3.7 (for $d_{\min}$). This concludes the proof of the first statement.

The second statement follows immediately from the definition of connectivity. □

### 10.4.1 Characterization in the Minimum Topology

We call a process-time graph $z_v$, for $v \in \mathcal{V}_O$, $v$-valent, if it starts from an initial configuration where all processes $p \in [n]$ have the same input value $x_p(z_v) = v$. Let $PS_{z_v}$ denote the connected component of $PS$ that contains the $v$-valent $z_v \in PS$. In the minimum topology $\mathcal{T}_{\min}^{\mathcal{PT}^\omega}$, we get the following characterization of consensus solvability:

**Theorem 10.4.4** (Consensus characterization)**.** *Consensus is solvable with a message adversary generating the set of admissible process-time graph sequences $PS$ if and only if there exists a partition of $PS$ into sets $PS(v), v \in \mathcal{V}_O$ such that the following holds:*

1. *Every $PS(v)$ is open in $PS$ with respect to the minimum topology.*

*2. Every admissible $v$-valent $z_v \in PS$ satisfies $z_v \in PS(v)$.*

*Proof.* ($\Rightarrow$): Define $PS(v) = (\Delta \circ \tau)^{-1}[\{v\}] \subseteq PS$ using the functions $\tau$ and $\Delta$ defined by an algorithm that solves consensus. This is clearly a partition of $PS$ by the termination and validity property of consensus. The validity condition of the algorithm also implies property (2). It thus remains to show openness of the $PS(v)$, which follows from the continuity of $\Delta \circ \tau : PS \to \mathcal{V}_O$, as every singleton $\{v\}$ is open (and closed) in the discrete topology.

($\Leftarrow$): We construct an algorithm as follows. Each process's state contains a variable $V$ whose value is equal to the projection of process-time graphs onto its own view. State updates happen in such a way that process $p$'s variable at the end of round $t$ in the execution with process-time graph $a$ is equal to $V = \pi_{\{p\}}(a^t)$. Process $p$ decides value $v$ in round $t$ if the ball of radius $\varepsilon = 2^{-t}$ around the set of sequences of process-time graphs $\pi^{-1}_{\{p\}}[\{V\}]$ compatible with its locally recorded view $V$ is contained in $PS(v)$, i.e., if

$$\{b \in PS \mid \pi_{\{p\}}(b^t) = V\} \subseteq PS(v) \ .$$

We first show termination of the resulting algorithm. Let $a \in PS$ and let $v \in \mathcal{V}_O$ such that $a \in PS(v)$. Since $PS(v)$ is open w.r.t. $d_{\min}$, there exists some $\varepsilon > 0$ such that

$$\{b \in PS \mid d_{\min}(b, a) < \varepsilon\} \subseteq PS(v) \ .$$

Let $t \geqslant 0$ such that $2^{-t} \leqslant \varepsilon$. We hence have

$$\{b \in PS \mid d_{\min}(b, a) < 2^{-t}\} \subseteq \{b \in PS \mid d_{\min}(b, a) < \varepsilon\} \subseteq PS(v) \ .$$

By Lemma 10.3.6, we have

$$\begin{aligned}\{b \in PS \mid d_{\{p\}}(b, a) < 2^{-t}\} &\subseteq \{b \in PS \mid d_{\min}(b, a) < 2^{-t}\} \\ &\subseteq PS(v)\end{aligned}$$

for every process $p \in [n]$. By the definition of the distance $d_{\{p\}}$, we have

$$\{b \in PS \mid d_{\{p\}}(b, a) < 2^{-t}\} = \{b \in PS \mid \pi_{\{p\}}(b^t) = \pi_{\{p\}}(a^t)\} \ .$$

Since, by construction, process $p$'s variable $V$ at the end of round $t$ is equal to $\pi_{\{p\}}(a^t)$, this shows that process $p$ decides in or before round $t$.

To show agreement, assume by contradiction that there exists some $a \in PS$ such that process $p_1$ decides $v_1$ in round $t_1$ and process $p_2$ decides $v_2$ in round $t_2$ in execution $\tau(a)$ with $q_1 \neq q_2$. By the definition of the algorithm, we have

$$\{b \in PS \mid \pi_{\{p_1\}}(b^{t_1}) = \pi_{\{p_1\}}(a^{t_1})\} \subseteq PS(v_1)$$

and

$$\{b \in PS \mid \pi_{\{p_2\}}(b^{t_2}) = \pi_{\{p_2\}}(a^{t_2})\} \subseteq PS(v_2) \ .$$

In particular, $a \in PS(v_1)$ and $a \in PS(v_2)$, a contradiction to the fact that the $PS(v)$ form a partition of $PS$.

Validity is an immediate consequence of property (2). □

This characterization gives rise to the following meta-procedure for determining whether consensus is solvable and constructing an algorithm if it is, which will be instantiated for some examples in Section 10.5. It requires knowledge of the connected components of the space $PS$ with respect to the minimum topology:

1. Initially, start with an empty set $PS(v)$ for every value $v \in \mathcal{V}_O$.
2. Add to $PS(v)$ every connected component $PS_{z_v}$ of every $v$-valent $z_v \in PS$.
3. Add every remaining connected component of $PS$ to an arbitrarily chosen set $PS(v)$ (i.e., decide on default value $v$).
4. If the sets $PS(v)$ are pairwise disjoint, then consensus is solvable. In this case, the sets $PS(v)$ determine a consensus algorithm via the universal construction in the proof of Theorem 10.4.4. If the $PS(v)$ are not pairwise disjoint, then consensus is not solvable.

In particular, this meta-procedure gives rise to the following succinct characterization of consensus solvability.

**Corollary 10.4.5.** *Consensus is solvable with a message adversary messageadversary generating the set of admissible process-time graph sequences $PS$ if and only if none of its connected components with respect to the minimum topology contains $z_v$ and $z_w$, $v, w \neq v \in \mathcal{V}_O$ that are $v$-valent and $w$-valent, respectively.*

We will now develop another characterization of consensus solvability, with rests on broadcastability of the $PS_{z_v}$.

**Definition 10.4.6** (Diameter of a set)**.** *For $A \subseteq \mathcal{PT}^\omega$ and $P'$ denoting either $P \subseteq [n]$ or* min*, define $A$'s diameter as $d_{P'}(A) = \sup\{d_{P'}(a, b) \mid a, b \in A\}$.*

**Definition 10.4.7** (Broadcastability)**.** *We call a subset $A \subseteq PS$ of admissible process-time graphs* broadcastable *by the broadcaster $p \in [n]$, if for every $a \in A$ there is some round $T(a) < \infty$ where every process $q \in [n]$ knows $p$'s input value $x_p(a)$ in $a$, i.e., $(p, 0, x_p(a))$ is in $V_{\{q\}}(a^{T(a)})$.*

We will now prove the essential fact that *connected* broadcastable sets have a diameter strictly smaller than 1:

**Theorem 10.4.2** (Diameter of broadcastable connected sets)**.** *If a connected set $A \subseteq PS$ of admissible process-time graph sequences is broadcastable by some process $p$, then $d_{\min}(A) \leqslant d_{\{p\}}(A) \leqslant 1/2$, i.e., $p$'s input value $x_p(a)$ is the same for all $a \in A$.*

*Proof.* Broadcastability by $p$ implies that, for any $a \in A$, every process $q$ has $(p, 0, x_p(a))$ in its local view $V_{\{q\}}(a^{T(a)})$ for some $T(a) < \infty$. Abbreviating $t = T(a)$, consider any $b \in B_{2^{-t}}(a) \cap A$ in the minimum topology. By definition of $B_{2^{-t}}(a)$, there is some process $q$ such that $V_{\{q\}}(b^t) = V_{\{q\}}(a^t)$, which together with $(p, 0, x_p(a)) \in V_{\{q\}}(a^t)$ implies $(p, 0, x_p(a)) \in V_{\{q\}}(b^t)$. So $b$ must have started from the *same* input value $x_p(b) = x_p(a) = V_{\{p\}}(a^0) = V_{\{p\}}(b^0)$.

We show now that this argument can be continued to reach every $b \in A$. For a contradiction, suppose that this is not the case and let $U(a)$ be the union of all these balls, recursively defined as follows: Let $U_0(a) = \{a\}$, for $m > 0$, $U_m(a) = \bigcup_{b \in U_{m-1}(a)} (B_{2^{-T(b)}}(b) \cap A)$, and finally $U(a) = \bigcup_{m \geqslant 0} U_m(a)$. As an arbitrary union of open balls intersected with $A$, which are all open in $A$, both $U_m(a)$ for every $m > 0$ and $U(a)$ is hence open in $A$. For every $b \in A \setminus U(a)$, $U(b)$ is also open in $A$, and so is $V(a) = \bigcup_{b \in A \setminus U(a)} U(b)$. However, the open sets $U(a)$ and $V(a)$ satisfy $U(a) \cup V(a) = A$, hence $A$ cannot be connected. □

Corollary 10.4.8 follows immediately from Theorem 10.4.2:

**Corollary 10.4.8** (Diameter of broadcastable $PS_{z_v}$)**.** *If $PS_{z_v}$ for a $v$-valent $z_v \in PS$ is broadcastable for $p$, then $d_{\min}(PS_{z_v}) \leqslant d_{\{p\}}(PS_{z_v}) \leqslant 1/2$ since $p$'s input value $x_p(a) = v$ is the same for all $a \in PS_{z_v}$.*

We can now prove the following necessary and sufficient condition for solving consensus based on broadcastability:

**Theorem 10.4.3** (Consensus characterization via broadcastability)**.** *A message adversary allows to solve consensus if and only if it guarantees that the connected components of the set $PS$ of admissible processes-time graphs are broadcastable for some process.*

*Proof.* ($\Leftarrow$) We need to prove that if every $PS_{z_v}$ is broadcastable for some process, then consensus is solvable. First, Theorem 10.4.2 secures that $d_{\min}(PS_{z_v}) \leqslant 1/2$. We claim that this implies that $PS_{z_v} \cap PS_{z_w} = \emptyset$ for every $w \neq v \in \mathcal{V}_O$: if they intersected, $PS_{z_v} = PS_{z_w}$ as they consist of connected components. Since $d_{\min}(z_v, z_w) = 1$, this would contradict $d_{\min}(PS_{z_v}) \leqslant 1/2$, however. Since this implies that $PS(v) = \bigcup_{z_v \in PS} PS_{z_v}$ and $PS(w) = \bigcup_{z_w \in PS} PS_{z_w}$ constructed in our meta-procedure are also disjoint, the algorithm given in Theorem 10.4.4 allows to solve consensus.

($\Rightarrow$) We prove the contrapositive: If there is some $PS_{z_v}$ with a $v$-valent process-time graph $z_v$ that is not broadcastable by any process, then we show that there is some $w$-valent sequence $z_w$ with $w \neq v$ such that $PS_{z_v} \cap PS_{z_w} \neq \emptyset$, which implies $PS_{z_v} = PS_{z_w}$ and hence makes consensus impossible by Theorem 10.4.4. More specifically, we will consider a finite sequence of process-time graphs $z_v = a_0, a_1, \ldots, a_n = z_w$ that is obtained from

the process-time graph $z_v$ by changing just the input values of the processes $1, \dots, n$ from $v$ to a fixed $w \neq v$, one by one. We show inductively that $a_i \in PS_{z_v}$ for every $i$, which proves our claim since $a_n = z_w$.

The induction basis is trivial, so suppose $a_i \in PS_{z_v}$ but $a_{i+1} \notin PS_{z_v}$. Since $a_i$ and $a_{i+1}$ differ only in the input value of $i$ and since $PS_{z_v}$ is not broadcastable by any process, hence also not by $i$, there is some process-time graph $b \in PS_{z_v}$ and a process $j \neq i$ with $(i, 0, x_i) \notin V_{\{j\}}(b^t)$ for every $t \geqslant 0$. Consequently, for the process-time graph $c$, which is the same as $b$ except that $i$'s input value $x_i = w$ instead of $v$, we get $d_{\{j\}}(b, c) = 0$ and hence $d_{\min}(b, c) = 0$. Since $c$ must be in the connected component of $a_{i+1}$, as the same sequence of communication graph sequences, used to get from $a_i$ to $b$, can be used to get from $a_{i+1}$ to $c$, this implies that $c \in PS_{a_{i+1}}$; these connected components must hence be the same. Thus, $a_{i+1} \in PS_{z_v}$, which completes the induction step and completes our proof. □

### 10.4.2 Characterization in the $P$-View Topologies

It is possible to shed some additional light on the consensus characterization given in Theorem 10.4.4, by exploiting the fact that $d_P$ (unlike $d_{\min}$) is a pseudo-metric: Since most of the convenient properties of metric spaces, including sequential compactness, also hold in pseudo-metric spaces, we can further explore the border of the decision sets $PS(v)$. It will turn out in Corollary 10.4.14 that consensus is impossible if and only if certain limit points in the $P$-view topology $\mathcal{T}_P^{\mathcal{PT}^\omega}$ are admissible.

For a given consensus algorithm, we again consider the set of all admissible process-time graph sequences $PS$ resp. the corresponding set of admissible executions $\Gamma$. We endow $PS$ with the subspace topology generated by $\mathcal{PT}^\omega \cap PS$ resp. $\Gamma$ with the subspace topology[5] generated by $\mathcal{C}^\omega \cap \Gamma$, both in the $P$-view topology. Recall that $PS$ and $\Gamma$ are not closed in general, hence not compact, even though $\mathcal{PT}^\omega$ resp. $\tau(\mathcal{PT}^\omega)$ are compact.

**Definition 10.4.9** (Distance of sets)**.** *For $A, B \subseteq \mathcal{PT}^\omega$, let $d_P(A, B) = \inf\{d_P(a, b) \mid a \in A,\ b \in B\}$.*

In [Now10, Theorem 4.3], the following theorem has been proved:

**Theorem 10.4.10** (Compact set distance condition)**.** *Let $A \subseteq \mathcal{PT}^\omega$ be closed and $B \subseteq \mathcal{PT}^\omega$ be compact with respect to the $[n]$-view topology. If $A \cap B = \emptyset$, then $d_{[n]}(A, B) > 0$.*

We prove the following result, which also holds when $A$, $B$ are not closed/compact. Note that it also implies Theorem 10.4.10 as a simple corollary for any $P$-view topology: Corollary 10.4.11 shows that it also holds in the minimum topology.

[5] Whenever we state a topological property w.r.t. the subspace topology, we will refer to $\Gamma$ (resp. $PS$), otherwise to $\mathcal{C}^\omega$ (resp. $\mathcal{PT}^\omega$).

**Theorem 10.4.4** (General set distance condition)**.** *Let $P \subseteq [n]$ with $P \neq \emptyset$. Let $A, B$ be arbitrary subsets of $\mathcal{PT}^\omega$. Then, $d_P(A,B) = 0$ if and only if there are infinite sequences $(a_k) \in A^\omega$ and $(b_k) \in B^\omega$ of process-time graphs as well as $\hat{a}, \hat{b} \in \mathcal{PT}^\omega$ with $a_k \to \hat{a}$ and $b_k \to \hat{b}$ with respect to the $P$-view topology and $d_P(\hat{a}, \hat{b}) = 0$.*

*Proof.* For direction $\Leftarrow$, assume that such sequences exist. The triangle inequality provides

$$3\varepsilon \geqslant d_P(a_k, \hat{a}) + d_P(\hat{a}, \hat{b}) + d_P(\hat{b}, b_k) \geqslant d_P(a_k, b_k)$$

for $t \geqslant \max\{T_a(\varepsilon), T_b(\varepsilon)\}$, since the sequences converge and hence $d_P(a_k, \hat{a}) \leqslant \varepsilon$ for $k \geqslant T_a(\varepsilon)$ (and analogous for $d(\hat{b}, b_k)$). Hence, $\inf\{d_P(a_k, b_k) \mid k \geqslant 1\} = 0$, and so $d_P(A,B) = 0$.

Conversely, assume $d_P(A,B) = 0$. By definition of inf, there is a pair of sequences $(e_k) \in A^\omega$ and $(f_k) \in B^\omega$ with $\inf\{d_P(e_k, f_k) \mid k \geqslant 1\} = 0$. Since $\mathcal{PT}^\omega$ is compact, there are convergent subsequences $(a_k) \subseteq (e_k)$ with $a_k \to \hat{a} \in \overline{A}$ and $(b_k) \subseteq (f_k)$ with $b_k \to \hat{b} \in \overline{B}$, respectively. They can be chosen such that, for every $\varepsilon > 0$, there is some $T(\varepsilon)$ such that $d_P(a_k, \hat{a}) \leqslant \varepsilon$ and $d_P(b_k, \hat{b}) \leqslant \varepsilon$ and also $d_P(a_k, b_k) \leqslant \varepsilon$ as $\inf\{d_P(a_k, b_k) \mid k \geqslant 1\} = 0$. In order to prove $d_P(\hat{a}, \hat{b}) = 0$, we note

$$d_P(\hat{a}, \hat{b}) \leqslant d_P(\hat{a}, a_k) + d_P(a_k, b_k) + d_P(b_k, \hat{b}) \leqslant 3\varepsilon.$$

So letting $t \to \infty$ causes $\varepsilon \to 0$ and hence indeed $d_P(\hat{a}, \hat{b}) = 0$. □

The representation $d_{\min} = \min_{p \in [n]} d_{\{p\}}$ from (10.3) allows us to extend this result from the $P$-topologies to the minimum topology:

**Corollary 10.4.11.** *Let $P \subseteq [n]$ with $P \neq \emptyset$. Let $A, B$ be arbitrary subsets of $\mathcal{PT}^\omega$. Then, $d_{\min}(A,B) = 0$ if and only if there are infinite sequences $(a_k) \in A^\omega$ and $(b_k) \in B^\omega$ of process-time graph sequences as well as $\hat{a}, \hat{b} \in \mathcal{PT}^\omega$ with $a_k \to \hat{a}$ and $b_k \to \hat{b}$ with respect to the minimum topology and $d_{\min}(\hat{a}, \hat{b}) = 0$.*

*Proof.* The proof of Theorem 10.4.4 can be carried over literally by using the fact that every convergent infinite sequence $(a^t)$ w.r.t. $d_{min}$ has a convergent infinite subsequence w.r.t. some $d_{\{p\}}$ by the pigeonhole principle. □

The above Theorem 10.4.4 allows us to distinguish 3 main cases that cause $d_P(A,B) = 0$: (i) If $\hat{a} \in A \cap B \neq \emptyset$, one can choose the sequences defined by $a_k = b_k = \hat{a} = \hat{b}$, $k \geqslant 1$. (ii) If $A \cap B = \emptyset$ and $\hat{a} = \hat{b}$, there is a "fair sequence" as the common limit. (iii) If $A \cap B = \emptyset$ and $\hat{a} \neq \hat{b}$, there is a pair of "unfair sequences" acting as limits, which have distance 0 (and are hence also common w.r.t. the pseudo-metric $d_P$). We note, however, that due to the non-uniqueness of the limits in our pseudo-metric, (iii) are actually two instances of (ii). We kept the distinction for compatibility with the existing results [FG11, Pfl18] for $n = 2$.

**Definition 10.4.12** (Fair and unfair process-time graph sequences)**.** *Consider two process-time graph sequences $r, r' \in \mathcal{PT}^\omega$ of some consensus algorithm with partitions $PS(v)$, $v \in \mathcal{V}_O$, in the $P'$-topology with $P' = \{p\}$, $p \in [n]$, or $P' = \min$:*

- *$r$ is called* fair*, if for some $v, w \neq v \in \mathcal{V}_O$ there are convergent sequences $(a_k) \in PS(v)$ and $(b_k) \in PS(w)$ with $a_k \to r$ and $b_k \to r$ with respect to $\mathcal{T}_{P'}^{\mathcal{PT}^\omega}$.*
- *$r$, $r'$ are called a pair of* unfair *sequences, if for some $v, w \neq v \in \mathcal{V}_O$ there are convergent sequences $(a_k) \in PS(v)$ with $a_k \to r$ and $(b_k) \in PS(w)$ with $b_k \to r'$ and $d_{P'}(r, r') = 0$ with respect to $\mathcal{T}_{P'}^{\mathcal{PT}^\omega}$.*

An illustration is shown in Figure 10.3.

The above findings go nicely with the alternative characterization of consensus solvability given in Corollary 10.4.14, which results from applying the following Lemma 10.4.13 from [Mun00] to Theorem 10.4.4.

**Lemma 10.4.13** (Separation lemma [Mun00, Lemma 23.12])**.** *If $Y$ is a subspace of $X$, a separation of $Y$ is a pair of disjoint nonempty sets $A$ and $B$ whose union is $Y$, neither of which contains a limit point of the other. The space $Y$ is connected if and only if there exists no separation of $Y$.*

*Proof.* The closure of a set $A$ in $Y$ is $(\overline{A} \cap Y)$, where $\overline{A}$ denotes the closure in $X$. To show that $Y$ is not connected implies a separation, assume that $A, B$ are closed and open in $Y = A \cup B$, so $A = (\overline{A} \cap Y)$. Consequently, $\overline{A} \cap B = \overline{A} \cap (Y - A) = \overline{A} \cap Y - \overline{A} \cap A = \overline{A} \cap Y - A = \emptyset$. Since $\overline{A}$ is the union of $A$ and its limit points, none of the latter is in $B$. An analogous argument shows that none of the limit points of $B$ can be in $A$.

Conversely, if $Y = A \cup B$ for disjoint non-empty sets $A$, $B$ which do not contain limit points of each other, then $\overline{A} \cap B = \emptyset$ and $a \cap \overline{B} = \emptyset$. From the equivalence above, we get $\overline{A} \cap Y = A$ and $\overline{B} \cap Y = B$, so both $A$ and $B$ are closed in $Y$ and, as each others complement, also open in $Y$ as well. □

**Corollary 10.4.14** (Separation-based characterization)**.** *Consensus is solvable with a message adversary messageadversary generating the set of admissible process-time graph sequences $PS$ if and only if there exists a partition of $PS$ into sets $PS(v), v \in \mathcal{V}_O$ such that the following holds:*

1. *No $PS(v)$ contains a limit point of any other $PS(w)$ w.r.t. the minimum topology in $\mathcal{PT}^\omega$.*
2. *Every $v$-valent admissible sequence $z_v$ satisfies $z_v \in PS(v)$.*

We hence immediately obtain:

**Corollary 10.4.15** (Fair/unfair consensus impossibility)**.** *The set of admissible process-time graphs $PS$ of a consensus algorithm $\mathcal{A}$ with partitions $PS(v)$, $v \in \mathcal{V}_O$, does not contain any fair process-time graph sequence $r$ or any pair $r, r'$ of unfair process-time graph sequences.*

## 10.5 Applications

In this section, we will apply our various topological characterizations of consensus solvability to particular classes of examples. We start in Section 10.5.1 with the broad class of message adversaries that have been shown, or can be shown, to make consensus impossible by means of bivalence proofs. We the proceed with a complete characterization of consensus solvability the class of compact message adversaries in Section 10.5.2. A characterization for the class of non-compact message adversaries will be provided in Section 10.5.3.

### 10.5.1 Bivalence-based Impossibilities

Our topological results shed some new light on the now standard technique of bivalence-based impossibility proofs introduced in the celebrated FLP paper [FLP85], which have been generalized in [MR02] and used in many different contexts: Our results reveal that the forever bivalent executions constructed inductively in bivalence proofs such as [SW89, SWK09] and [BRS+18, WSS19] are just the common limit of two infinite sequence of executions $\alpha_0, \alpha_1, \dots$ all contained in, say, $\Gamma(0)$ and $\beta_0, \beta_1, \dots$ all contained in $\Gamma(1)$ that have a common limit $\alpha_k \to \hat{\alpha}$ and $\beta_k \to \hat{\beta}$ in some $\{p\}$-view topology with $d_{\{p\}}(\hat{\alpha}, \hat{\beta}) = 0$.

More specifically, what is common to these proofs is that one shows that, for any consensus algorithm, there is an admissible forever bivalent run. This is usually done inductively, by showing that there is a bivalent initial configuration and that, given a bivalent configuration $C^{t-1}$ at the end of round $t-1$, there is a 1-round extension leading to a bivalent configuration $C^t$ at the end of round $t$. By definition, bivalence of $C^t$ means that there are two admissible executions $\alpha_t$ with decision value 0 and $\beta_t$ with decision value 1 starting out from $C^t$, i.e., having a common prefix that leads to $C^t$. Consequently, their distance, in any $\{p\}$-view topology, satisfies $d_{\{p\}}(\alpha_t, \beta_t) < 2^{-t}$. Note that this is also true for the more general concept of a bipotent configuration $C^t$, as introduced in [MR02].

By construction, the $(t-1)$-prefix of $\alpha_t$ and $\alpha_{t-1}$ are the same, for all $t$, which implies that they converge to a limit $\hat{\alpha}$ (and analogously for $\hat{\beta}$), see Figure 10.3 for an illustration. Therefore, these executions match Definition 10.4.12, and Corollary 10.4.15 implies that the stipulated consensus algorithm cannot be correct. Concrete examples are the lossy link impossibility [SW89], i.e., the impossibility of consensus under an oblivious message adversary for $n = 2$ that may choose any graph out of the set $\{\leftarrow, \leftrightarrow, \rightarrow\}$, and the impossibility of solving consensus with vertex-stable source components with insufficient

stability interval [BRS+18, WSS19]. In the case of the oblivious lossy link message adversary using the reduced set $\{\leftarrow, \rightarrow\}$ considered in [CGP15], consensus is solvable and there is no forever bivalent run. Indeed, there exists a consensus algorithm, such that all configurations reached after the first round are already univalent.

### 10.5.2 Compact Message Adversaries

In this section, we consider message adversaries (like oblivious ones [SW89, CGP15]) that are limit-closed, in the sense that every convergent sequence of process-time graphs $a_0, a_1, \ldots$ with $a_i \in PS$ for every $i$ has a limit $\hat{a} \in PS$. An illustration is shown in Fig. 10.2, where the blue dots represent the $a_i$'s and $\times$ the limit point $\hat{a}$ at the boundary.

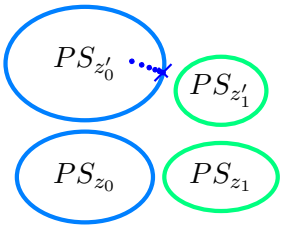


Figure 10.2: Examples of two connected components of the decision sets $PS(0) = PS_{z_0} \cup PS_{z'_0}$ and $PS(1) = PS_{z_1} \cup PS_{z'_1}$ for a compact message adversary. They are closed in $\mathcal{PT}^\omega$, hence contain all their limit points (marked by $\times$) and have a distance $> 0$ by Theorem 10.4.10.

In this case, the set of admissible process-time graph sequences $PS$ is closed and hence a compact subspace both in any $P$-view topology and in the minimum topology. Moreover, we obtain:

**Corollary 10.5.1** (Decision sets for compact MAs are compact)**.** *For every correct consensus algorithm for a compact message adversary and every $v \in \mathcal{V}_O$, $PS(v)$ is closed in $PS$ and compact, and $d_{\{p\}}(PS(v), PS(w)) > 0$ for any $v, w \neq v \in \mathcal{V}_O$ and $p \in [n]$, and hence also $d_{\min}(PS(v), PS(w)) > 0$.*

*Moreover, there are only finitely many different connected components $PS_x$, $x \in PS$, which are all compact, and for every $x, y$ with $PS_x \neq PS_y$, it holds that $d_{\{p\}}(PS_x, PS_y) > 0$ and hence also $d_{\min}(PS_x, PS_y) > 0$.*

*Proof.* Since all decision sets $PS(v)$ are closed in $PS$ by Theorem 10.4.4 and $PS$ is compact for a compact message adversary, it follows that every $PS(v)$ is also compact. From Theorem 10.4.10 it hence follows that $d_{\{p\}}(PS(v), PS(w)) > 0$. As this holds for every $p \in [n]$, we also have $d_{\min}(PS(v), PS(w)) > 0$.

Since every connected component $PS_x$ of $PS$ that contains $x$ is closed in $PS$, as the closure of a connected subspace is also connected by [Mun00, Lemma 23.4] and a connected component is maximal, the same arguments as above also apply to $PS_x$. To show that

there are only finitely many different $PS_{z_v}$ for $v$-valent sequences $z_v \in PS$, observe that $PS(v) = \bigcup_{z_v \in PS} PS_{z_v}$ is an open covering of $PS(v)$. Since the latter is compact, there is a finite sub-covering $PS(v) = PS_{z_v^1} \cup \cdots \cup PS_{z_v^m}$, and all other $PS_{z_v}$ for a $v$-valent $z_v$ must be equal to one of those, as connected components are either disjoint or identical. □

We now make the abstract characterization of Theorem 10.4.4 and our meta-procedure more operational, by introducing the $\varepsilon$-approximation of the connected component $PS_z$ that contains a process-time graph $z \in PS$, typically for some $\varepsilon = 2^{-t}$, $t \geqslant 0$. It is constructed iteratively, using finitely many iterations (since the number of different possible $t$-prefixes satisfies $|\mathcal{PT}^t| < \infty$) of the following algorithm:

**Definition 10.5.2** ($\varepsilon$-approximations)**.** *Let $z \in PS$ be an admissible process-time graph. In the minimum topology, we iteratively define $PS_z^\varepsilon$, for $\varepsilon > 0$, as follows: $PS_z^\varepsilon[0] = \{z\}$; for $\ell > 0$, $PS_z^\varepsilon[\ell] = \bigcup_{a \in PS_z^\varepsilon[\ell-1]} (B_\varepsilon(a) \cap PS)$; and $PS_z^\varepsilon = PS_z^\varepsilon[m]$ where $m < \infty$ is such that $PS_z^\varepsilon[m] = PS_z^\varepsilon[m+1]$. For $v \in \mathcal{V}_O$, the $\varepsilon$-approximation $PS^\varepsilon(v)$ is defined as $PS^\varepsilon(v) = \bigcup_{z_v \in PS} PS_{z_v}^\varepsilon$, where every $z_v$ denotes a $v$-valent process-time graph.*

**Lemma 10.5.3** (Properties of $\varepsilon$-approximation)**.** *For every $\varepsilon > 0$, every $v, w \in \mathcal{V}_O$, every $z \in PS$, $v$-valent $z_v$, and every $w$-valent $z_w$, the $\varepsilon$-approximations have the following properties:*

*(i) For a compact message adversary, there are only finitely many different $PS_z^\varepsilon$, $z \in PS$.*

*(ii) For every $0 < \varepsilon' \leqslant \varepsilon$, it holds that $PS_{z_v}^{\varepsilon'} \subseteq PS_{z_v}^\varepsilon$.*

*(iii) $PS_{z_v}^\varepsilon \cap PS_{z_w}^\varepsilon \neq \emptyset$ implies $PS_{z_v}^\varepsilon = PS_{z_w}^\varepsilon$.*

*(iv) $PS_z \subseteq PS_z^\varepsilon$.*

*Proof.* Properties (ii)–(iv) hold for arbitrary message adversaries: To prove (ii), it suffices to mention $B_{\varepsilon'}(a) \subseteq B_\varepsilon(a)$. As for (iii), if $a \in PS_x^\varepsilon \cap PS_y^\varepsilon \neq 0$, the iterative construction of $PS_x^\varepsilon$ would reach $a$, which would cause it to also include the whole $PS_y^\varepsilon$, as the latter also reaches $a$. If (iv) would not hold, $PS_z$ could be separated into disjoint open sets, which contradicts connectivity. Finally, (i) holds for compact message adversaries, since Corollary 10.5.1 implies that there are only finitely many different connected components $PS_z$, which carry over to $PS_z^\varepsilon$ by (iii) and (iv). □

We now show that $PS_x^\varepsilon$ and $PS_y^\varepsilon$ for sequences $x$ and $y$ with $PS_x \neq PS_y$ have a distance $> 0$, provided $\varepsilon$ is sufficiently small:

**Lemma 10.5.4** (Separation of $\varepsilon$-approximations for compact MAs)**.** *For a compact message adversary that allows to solve consensus, let $x \in PS$ and $y \in PS$ be such that $PS_x \neq PS_y$. Then, there is some $\varepsilon > 0$ such that, for any $0 < \varepsilon' \leqslant \varepsilon$, it holds that $d_{\min}(PS_x^{\varepsilon'}, PS_y^{\varepsilon'}) > 0$.*

*Proof.* According to Corollary 10.5.1, the components $PS_x$ and $PS_y$ are compact. Theorem 10.4.10 reveals that we have $d_{\min}(PS_x, PS_y) = d > 0$. By Lemma 10.5.3.(iv), for every $\varepsilon > 0$, $PS_x \subseteq PS_x^\varepsilon$ and $PS_y \subseteq PS_y^\varepsilon$. Therefore, setting $\varepsilon < d/2$ secures $d_{\min}(PS_x^\varepsilon, PS_y^\varepsilon) > 0$. □

We immediately get the following corollary, which allows us to reformulate Theorem 10.4.3 as given in Theorem 10.5.1.

**Corollary 10.5.5** (Matching $\varepsilon$-approximation)**.** *For a compact message adversary, if $\varepsilon > 0$ is chosen in accordance with Lemma 10.5.4, then $PS_z^\varepsilon = PS_z$ for every $z \in PS$.*

**Theorem 10.5.1** (Consensus characterization for compact MAs)**.** *A compact message adversary allows to solve consensus if and only if there is some $\varepsilon > 0$ such that every $v$-valent $PS_{z_v}^\varepsilon$, $v \in \mathcal{V}_O$, is broadcastable for some process.*

*Proof.* Our theorem follows from Theorem 10.4.3 in conjunction with Corollary 10.5.5. □

Note carefully that if consensus is solvable, then, for every $0 < \varepsilon' \leqslant \varepsilon$, the universal algorithm from Theorem 10.4.4 with $PS(v) = PS^{\varepsilon'}(v) \cup PS \setminus \bigcup_{w \neq v \in \mathcal{V}_O} PS^{\varepsilon'}(w)$ for some arbitrary value $v \in \mathcal{V}_O$, and $PS(w) = PS^{\varepsilon'}(w)$ for the remaining $w \in \mathcal{V}_O$, can be used.

We conclude this section with noting that checking the broadcastability of $PS_{z_v}^\varepsilon$ can be done by checking broadcastability only in *finite* prefixes. More specifically, like the decision function $\Delta$ of consensus, the function $T(a)$ that gives the round by which every process in $a \in PS$ has $(p, 0, x_p(a))$ of the broadcaster $p$ in its view is locally constant for a sufficiently small neighborhood, namely, $B_{2^{-T(a)}}(a)$, and hence continuous. Since $PS_{z_v} = PS_{z_v}^\varepsilon$ is compact, $T(a)$ is in fact uniformly continuous and hence attains its maximum $\hat{T}$ in $PS_{z_v}^\varepsilon$. It hence suffices to check broadcastability in the $t$-prefixes of $PS_{z_v}^\varepsilon$ for $t = \max\{\lfloor \log_2(1/\varepsilon) \rfloor, \hat{T}\}$ in Theorem 10.5.1.

### 10.5.3 Non-compact Message Adversaries

In this section, we finally consider message adversaries that are not limit-closed, like the ones of [FG11, Pfl18, WSS19]. Unfortunately, we cannot use the $\varepsilon$-approximations according to Definition 10.5.2 for non-compact message adversaries. Even if $\varepsilon$ is made arbitrarily small, Lemma 10.5.4 does not hold. An illustration is shown in Fig. 10.3. Hence, adding a ball $B_\varepsilon(a)$ in the iterative construction of $PS_z^\varepsilon$, where $d_{\min}(a, r) < \varepsilon$ for some forbidden limit sequence $r$, inevitably lets the construction grow into some $PS_{z'}^\varepsilon$, where $z'$ has a different valence than $z$. Whereas this could be avoided by adapting $\varepsilon$ when coming close to $r$, the resulting approximation does not provide any advantage over directly using the connected components.

Actually, this is in accordance with the existing solutions for non-compact message adversaries we are aware of, albeit they typically use their complement, namely, excluded sequences. For example, the binary consensus algorithms for $n = 2$ given in [FG11, Pfl18]

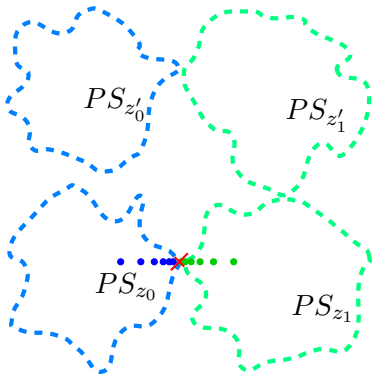


Figure 10.3: Examples of two connected components of the decision sets $PS(0) = PS_{z_0} \cup PS_{z'_0}$ and $PS(1) = PS_{z_1} \cup PS_{z'_1}$ for a non-compact message adversary. They are not closed in $\mathcal{PT}^\omega$ and may have distance 0; common limit points (like for $PS_{z_0}$ and $PS_{z_1}$, marked by $\times$) must hence be excluded by Corollary 10.4.14.

assume that the algorithm knows a fair or a pair of unfair sequences a priori, which effectively partition the sequence space into two connected components.[6] The $(D+1)$-VSRC message adversary of [WSS19] even excludes all sequences without a root component that is vertex-stable for at least $D+1$ rounds, where $D$ is the dynamic diameter that ensures broadcastability by all root members, which renders the remaining connected components broadcastable.

We restate the following necessary and sufficient condition for solving consensus with a non-compact message adversary from Theorem 10.4.3:

**Theorem 10.5.6** (Consensus characterization for non-compact MAs)**.** *A non-compact message adversary allows to solve consensus if and only if every $v$-valent $PS_{z_v}$, $v \in \mathcal{V}_O$, is broadcastable for some process.*

We will finally prove that the set of to be excluded limit sequences for any decision set $PS(v)$ is compact for a message adversary that allows to solve consensus:

**Lemma 10.5.7** (Compactness of excluded sequences)**.** *Let $PS(v)$, $v \in \mathcal{V}_O$, be any decision set of a correct consensus algorithm for an arbitrary message adversary, $\overline{PS}(v)$ be its closure in $\mathcal{PT}^\omega$ and $Int(PS)(v)$ its interior. Then, $\hat{PS}(v) = \overline{PS}(v) - Int(PS)(v)$, which is the set of to be excluded limit points, is compact.*

*Proof.* The closure $\overline{PS}(v)$ is closed by definition. Since the complement of the interior $\mathrm{Int}(PS)(v)^C$ is also closed by definition, it follows that $\hat{PS}(v) = \overline{PS}(v) - \mathrm{Int}(PS)(v) = \overline{PS}(v) \cap \mathrm{Int}(PS)(v)^C$ is also closed. As a closed subset of the compact set $\overline{PS}(v)$, $\hat{PS}(v)$ is hence compact. □

[6] Note that there are uncountably many choices for separating $PS(0)$ and $PS(1)$ here.

Chapter 11

# Concluding Remarks

This thesis investigated various aspects of solving deterministic consensus in dynamic networks controlled by message adversaries. It turned out that a fundamental property of a communication pattern of a message adversary is its dynamic radius, which is the number of rounds needed to achieve a non-empty kernel, defined as the set of broadcasters that manage to reach all processes. As a crucial prerequisite for every consensus algorithm, broadcastability indeed plays, either directly or indirectly, an important role in almost all of our results. It also has a close relation to the rootedness of the communication graphs, which we assumed throughout most of the thesis, although it is possible to solve consensus also when this condition is dropped.

We complete the thesis by presenting some insights and conclusions that we draw from the individual chapters.

In **Chapter 4**, we studied nonsplit message adversaries, which are a convenient abstraction that arises naturally when considering information dissemination in a variety of message adversaries. Since classic information dissemination problems are trivially impossible in these nonsplit message adversaries, it made sense to study the more relaxed dynamic radius here. As we showed in Theorem 4.0.4, this is an important characteristic with respect to the impossibility of exact consensus. For our main technical contribution, we proved a new upper bound in Theorem 4.0.2, which shows that the dynamic radius of nonsplit message adversaries is in $O(\log \log n)$. This is an exponential improvement of the best previously known upper bound of $O(\log n)$.

We also showed an upper bound of 2 asynchronous rounds for the dynamic radius in the asynchronous message passing model with crash failures. Thus, in this important class of nonsplit message adversaries, information dissemination is considerably faster than what is currently known for the general case.

While this is another hint at the usefulness of the nonsplit abstraction for message adversaries, the tightness of this bound remains an open question. For message adversaries, whose communication graphs are rooted, the upper bound of the dynamic radius of

$O(n \log\log n)$ from Theorem 4.0.3 is still not matched by the best lower bound of $\Omega(n)$ from the literature.

**Chapter 5** presented our combinatorial characterization of the solvability of consensus for closed message adversaries, which highlights the importance of a non-empty kernel of a communication pattern. Our result has the immediate following intuitive interpretation: There exists a round, such that in this round it is common knowledge that some nontrivial set occurs in the Kernel. This is plausible since the equivalence class of to the transitive indistinguishability relation, respectively its kernel, can always be computed. Compared to the work on oblivious message adversaries by Coulouma, Godard, and Peters, our characterization is a simple statement on the sequences of the message adversary itself, rather than an involved $\beta$-relation, defined on the set of allowed graphs $\mathbf{D}$. Operationally, given a message adversary $\mathsf{MA}$, it is straightforward to check whether consensus is solvable under $\mathsf{MA}$ by means of our condition.

Moving on from closed message adversaries, in **Chapter 6** we provided tight upper and lower bounds for the solvability of consensus under message adversaries that guarantee a vertex-stable root component (which consists of the same set of processes, with possibly changing interconnection topology) only eventually and only for a short period of time: We showed that consensus is solvable if and only if each graph has exactly one root component and, eventually, there is a period of at least $D+1$ consecutive rounds (with $D \leqslant n-1$ denoting the dynamic diameter, i.e., the number of rounds required by root members for broadcasting) where the root component remains the same. We also provided a matching consensus Algorithm 6.2, along with its correctness proof.

One aspect that therefore might warrant further study, is the dependence of Algorithm 6.2 on the knowledge of the system size. As we know, counting the number of processes is hopeless under a message adversary in general (there can always be an arbitrarily long outgoing chain from a process), yet knowledge of $n$ alone is insufficient to solve consensus, as we have seen in Theorem 6.3.6. Still, Algorithm 6.2 shows that exact knowledge of $n$ is not necessary under the message adversary $\Diamond\mathsf{STABLE}_{n,D}$, and an estimate $N \geqslant n$ suffices. It remains an open question, how far this relaxation can be taken and whether agreement on $N$ is in fact necessary or whether there is an algorithm that can even cope with each process $p$ holding a different estimate $N_p \geqslant n$.

Regarding solutions to uniform consensus, where the number of processes is unknown, Theorem 6.3.1 showed that uniform consensus is impossible if the duration of the stability period is $< 2D$. For a stability period with duration $> 2D$, Chapter 7 presents uniform consensus algorithms that work even under assumptions on the communication graphs that are more relaxed than to assume that all of them are rooted. This means that, even though almost the entire range of durations of the stability window w.r.t. the solvability of uniform consensus is now explored, the question of whether a stability duration of exactly $2D$ rounds is sufficient, remains open.

In **Chapter 7**, we investigated how consensus solvability is affected, if the assumption that all communication graphs are rooted is dropped. This lead to different generalizations,

the simplest of which assumes that a constant fraction of the communication graphs is rooted. In this case, we could essentially apply all of the previously established techniques.

A more refined generalization essentially asserts a single root component, only for a certain number of (partly consecutive) rounds ($\Diamond\mathsf{STABLE}_{n,D}(z_1, z_2, D)$). We established that no deterministic algorithm can terminate earlier than $2D + 1$ rounds after stabilization in some execution under this message adversary. and provided a matching algorithm, along with its correctness proof.

We then proceeded to investigate $k$-set agreement, where we showed that even extensive partitioning into less than $k$ components sometimes does not allow to solve $k$-set agreement. For these results, we used also a generic $k$-set agreement impossibility theorem that formalizes partitioning arguments.

In **Chapter 8**, we proceeded to look out for a simpler consensus algorithm. In particular, we found a new and simple algorithmic technique for solving consensus under our previously introduced eventually stabilizing message adversary. Aside from hopefully contributing to making this model more accessible, we showed that our new method provides an asymptotically tight solution. The source of simplicity in our algorithm primarily comes from dividing the task of solving consensus into two separate tasks, which has already been used implicitly in Chapter 5: One centralized task that analyzes the execution prefixes of our message adversary, and one distributed task, which exploits this analysis to decide despite of local uncertainty. The central toolbox we used here are statements about rooted message adversaries, which leads us to conjecture that our techniques should be widely applicable for this important class of message adversaries.

In our analysis, we showed that the predicate on $\Delta(.)$ given in Assumption 1 is sufficient for solving consensus under some message adversary. It is unclear, however, whether it is also necessary. Weakening this predicate might hence be an important step towards a combinatorial characterization of consensus solvability for general message adversaries.

With respect to the relation of message adversaries and failure detectors, we found in **Chapter 9** that, although the principal ability to simulate $(\Sigma, \Omega)$ atop of the simulated system $\mathcal{SMP}_n[adv : \mathsf{STAR}]$ sounds promising for developing consensus algorithms for message adversaries at first sight, it is actually not very useful in practice: After all, it hinges on the availability of a consensus algorithm for the bottom-level message adversary $\mathsf{MA}'$! Consequently, this simulation does not open up a viable alternative to the development of consensus algorithms tailored to specific message adversaries like the ones introduced in the preceding chapters.

Overall, it turned out that strongest message adversaries according to Definition 9.4.4 do not have much discriminating power, as essentially all message adversaries known to us that allow to solve consensus are strongest according to Corollary 9.4.6. Finding a better definition of a strongest message adversary might hence be a topic for future research. Note, however, that naive alternative definitions like one that (i) admits a solution algorithm and (ii) is maximal w.r.t. its set of admissible graph sequences it may generate do not easily work out: Given that the latter set is usually uncountable,

as admissible graph sequences are infinite, it is not clear whether (ii) is well-defined in general.

Finally, in **Chapter 10**, we provided a complete topological characterization of the solvability of consensus in directed dynamic networks of arbitrary size controlled by a general message adversary using point-set topology: We found that consensus can only be solved when the space of admissible process-time graph sequences can be partitioned into non-empty sets that are both closed and open in specific topologies. This requires exclusion of certain fair and unfair limit sequences, which limit broadcastability and happen to coincide with the forever bivalent executions constructed in bivalence and bipotence proofs.

Our topological framework should be generalizable other distributed computing models and, most importantly, to other decision problems. Another very interesting area of future research is to study the homology of non-compact message adversaries, i.e., the topological structure of the space of admissible executions using algebraic topology.

# List of Figures

# List of Algorithms

# Bibliography


[ADGFT01] Marcos Kawazoe Aguilera, Carole Delporte-Gallet, Hugues Fauconnier, and Sam Toueg. Stable leader election. In *DISC '01: Proceedings of the 15th International Conference on Distributed Computing*, pages 108–122. Springer-Verlag, 2001.

[ADGFT03] Marcos K. Aguilera, Carole Delporte-Gallet, Hugues Fauconnier, and Sam Toueg. On implementing Omega with weak reliability and synchrony assumptions. In *Proceeding of the 22nd Annual ACM Symposium on Principles of Distributed Computing (PODC'03)*, pages 306–314, New York, NY, USA, 2003. ACM Press.

[ADGFT04] Marcos Kawazoe Aguilera, Carole Delporte-Gallet, Hugues Fauconnier, and Sam Toueg. Communication-efficient leader election and consensus with limited link synchrony. In *Proceedings of the 23th ACM Symposium on Principles of Distributed Computing (PODC'04)*, pages 328–337, St. John's, Newfoundland, Canada, 2004. ACM Press.

[AFM+04] Emmanuelle Anceaume, Antonio Fernández, Achour Mostéfaoui, Gil Neiger, and Michel Raynal. A necessary and sufficient condition for transforming limited accuracy failure detectors. *J. Comput. Syst. Sci.*, 68(1):123–133, 2004.

[AG13] Yehuda Afek and Eli Gafni. Asynchrony from synchrony. In Davide Frey, Michel Raynal, Saswati Sarkar, RudrapatnaK. Shyamasundar, and Prasun Sinha, editors, *Distributed Computing and Networking*, volume 7730 of *Lecture Notes in Computer Science*, pages 225–239. Springer Berlin Heidelberg, 2013.

[AS85] Bowen Alpern and Fred B. Schneider. Defining liveness. *Information Processing Letters*, 21(4):181–185, 1985.

[AW04] Hagit Attiya and Jennifer Welch. *Distributed Computing*. John Wiley & Sons, 2nd edition, 2004.

[BHDPW07] Martin Biely, Martin Hutle, Lucia Draque Penso, and Josef Widder. Relating stabilizing timing assumptions to stabilizing failure detectors regarding solvability and efficiency. In *9th International Symposium on Stabilization, Safety, and Security of Distributed Systems*, volume 4838 of *Lecture Notes in Computer Science*, pages 4–20, Paris, November 2007. Springer Verlag.

[BMZ90] Ofer Biran, Shlomo Moran, and Shmuel Zaks. A combinatorial characterization of the distributed 1-solvable tasks. *Journal of Algorithms*, 11(3):420 – 440, 1990.

[BR11] François Bonnet and Michel Raynal. On the road to the weakest failure detector for k-set agreement in message-passing systems. *Theoretical Computer Science*, 412(33):4273 – 4284, 2011.

[BR19] Martin Biely and Peter Robinson. On the hardness of the strongly dependent decision problem. In *Proceedings of the 20th International Conference on Distributed Computing and Networking, ICDCN 2019, Bangalore, India, January 04-07, 2019*, pages 120–123, 2019.

[BRS11] Martin Biely, Peter Robinson, and Ulrich Schmid. Easy impossibility proofs for k-set agreement in message passing systems. In *Proceedings 15th International Conference on Principles of Distributed Systems (OPODIS'11), Springer LNCS 7109*, pages 299–312, 2011.

[BRS12] Martin Biely, Peter Robinson, and Ulrich Schmid. Agreement in directed dynamic networks. In *Proceedings 19th International Colloquium on Structural Information and Communication Complexity (SIROCCO'12)*, LNCS 7355, pages 73–84. Springer-Verlag, 2012.

[BRS+15] Martin Biely, Peter Robinson, Ulrich Schmid, Manfred Schwarz, and Kyrill Winkler. Gracefully degrading consensus and $k$-set agreement in directed dynamic networks. In *Revised selected papers Third International Conference on Networked Systems (NETYS'15)*, Springer LNCS 9466, pages 109–124, Agadir, Morocco, 2015. Springer International Publishing.

[BRS+18] Martin Biely, Peter Robinson, Ulrich Schmid, Manfred Schwarz, and Kyrill Winkler. Gracefully degrading consensus and k-set agreement in directed dynamic networks. *Theoretical Computer Science*, 726:41–77, 2018.

[BSW11] Martin Biely, Ulrich Schmid, and Bettina Weiss. Synchronous consensus under hybrid process and link failures. *Theoretical Computer Science*, 412(40):5602 – 5630, 2011. http://dx.doi.org/10.1016/j.tcs.2010.09.032.

[BZM14] Ido Ben-Zvi and Yoram Moses. Beyond Lamport's happened-before: On time bounds and the ordering of events in distributed systems. *J. ACM*, 61(2):13:1–13:26, April 2014.

[CBFN15] Bernadette Charron-Bost, Matthias Függer, and Thomas Nowak. Approximate consensus in highly dynamic networks: The role of averaging algorithms. In Magnús M. Halldòrsson, Kazuo Iwama, Naoki Kobayashi, and Bettina Speckmann, editors, *Automata, Languages, and Programming*, volume 9135 of *Lecture Notes in Computer Science*, pages 528–539. Springer Berlin Heidelberg, 2015.

[CBFN16] Bernadette Charron-Bost, Matthias Függer, and Thomas Nowak. Fast, Robust, Quantizable Approximate Consensus. In Ioannis Chatzigiannakis, Michael Mitzenmacher, Yuval Rabani, and Davide Sangiorgi, editors, *43rd International Colloquium on Automata, Languages, and Programming (ICALP 2016)*, Rome, Italy, July 2016. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.

[CBHW10] Bernadette Charron-Bost, Martin Hutle, and Josef Widder. In search of lost time. *Information Processing Letters*, 110(21):928–933, October 2010.

[CBS09] Bernadette Charron-Bost and André Schiper. The Heard-Of model: computing in distributed systems with benign faults. *Distributed Computing*, 22(1):49–71, April 2009.

[CFP+19] Armando Castañeda, Pierre Fraigniaud, Ami Paz, Sergio Rajsbaum, Matthieu Roy, and Corentin Travers. A topological perspective on distributed network algorithms. In Keren Censor-Hillel and Michele Flammini, editors, *Structural Information and Communication Complexity*, pages 3–18, Cham, 2019. Springer International Publishing.

[CFQS11] Arnaud Casteigts, Paola Flocchini, Walter Quattrociocchi, and Nicola Santoro. Time-varying graphs and dynamic networks. In Hannes Frey, Xu Li, and Stefan Rührup, editors, *ADHOC-NOW*, volume 6811 of *Lecture Notes in Computer Science*, pages 346–359. Springer, 2011.

[CFQS12] Arnaud Casteigts, Paola Flocchini, Walter Quattrociocchi, and Nicola Santoro. Time-varying graphs and dynamic networks. *IJPEDS*, 27(5):387–408, 2012.

[CG13] Étienne Coulouma and Emmanuel Godard. A characterization of dynamic networks where consensus is solvable. In *Proceedings Structural Information and Communication Complexity - 20th International Colloquium (SIROCCO'13), Springer LNCS 8179*, pages 24–35, 2013.

[CGP15] Étienne Coulouma, Emmanuel Godard, and Joseph G. Peters. A characterization of oblivious message adversaries for which consensus is solvable. *Theor. Comput. Sci.*, 584:80–90, 2015.

[CT96] Tushar Deepak Chandra and Sam Toueg. Unreliable failure detectors for reliable distributed systems. *Journal of the ACM*, 43(2):225–267, March 1996.

[DDS87] Danny Dolev, Cynthia Dwork, and Larry Stockmeyer. On the minimal synchronism needed for distributed consensus. *Journal of the ACM*, 34(1):77–97, January 1987.

[DGFG+04] Carole Delporte-Gallet, Hugues Fauconnier, Rachid Guerraoui, Vassos Hadzilacos, Petr Kouznetsov, and Sam Toueg. The weakest failure detectors to solve certain fundamental problems in distributed computing. In *Proceedings of the 23rd ACM Symposium on Principles of Distributed Computing (PODC'04)*, pages 338–346. ACM Press, 2004.

[Die06] Reinhard Diestel. *Graph Theory (3rd ed.)*. Springer, 2006.

[DLS88] Cynthia Dwork, Nancy Lynch, and Larry Stockmeyer. Consensus in the presence of partial synchrony. *Journal of the ACM*, 35(2):288–323, April 1988.

[ERS66] Paul Erdös, Alfréd Rényi, and Vera T. Sós. On a problem of graph theory. *Studia Sci. Math. Hungar.*, 1:215–235, 1966.

[FG11] Tristan Fevat and Emmanuel Godard. Minimal obstructions for the coordinated attack problem and beyond. In *25th IEEE International Symposium on Parallel and Distributed Processing, IPDPS 2011, Anchorage, Alaska, USA, 16-20 May, 2011 - Conference Proceedings*, pages 1001–1011, 2011.

[FH07] Jasmin Fisher and Thomas A Henzinger. Executable cell biology. *Nature Biotechnology*, 25:1239–1249, 2007.

[FK13] Ofer Feinerman and Amos Korman. Theoretical distributed computing meets biology: A review. In Chittaranjan Hota and Pradip K. Srimani, editors, *Distributed Computing and Internet Technology*, pages 1–18, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.

[FL94] Pierre Fraigniaud and Emmanuel Lazard. Methods and problems of communication in usual networks. *Discrete Applied Mathematics*, 53(1):79 – 133, 1994.

[FLP85] Michael J. Fischer, Nancy A. Lynch, and M. S. Paterson. Impossibility of distributed consensus with one faulty process. *Journal of the ACM*, 32(2):374–382, April 1985.

[FMHV03] R. Fagin, Y. Moses, J.Y. Halpern, and M.Y. Vardi. *Reasoning About Knowledge*. MIT Press, 2003.

[FNW19] Matthias Függer, Thomas Nowak, and Kyrill Winkler. On the radius of nonsplit graphs and information dissemination in dynamic networks. *CoRR*, abs/1901.06824, 2019.

[FR10] A. Fernández and M. Raynal. From an asynchronous intermittent rotating star to an eventual leader. *IEEE Trans. Parallel Distrib. Syst.*, 21(9):1290–1303, 2010.

[Fre14] Ronald C. Freiwald. *An Introduction to Set Theory and Topology*. Washington University in St. Louis, 2014.

[Gaf98] Eli Gafni. Round-by-round fault detectors (extended abstract): unifying synchrony and asynchrony. In *Proceedings of the Seventeenth Annual ACM Symposium on Principles of Distributed Computing*, pages 143–152, Puerto Vallarta, Mexico, 1998. ACM Press.

[GKFS10] Alois Goiser, Samar Khattab, Gerhard Fassl, and Ulrich Schmid. A new robust interference reduction scheme for low complexity direct-sequence spread-spectrum receivers: Performance. In *Proceedings 3rd International IEEE Conference on Communication Theory, Reliability, and Quality of Service (CTRQ'10)*, pages 15–21, Athens, Greece, June 2010.

[HHL88] Sandra M. Hedetniemi, Stephen T. Hedetniemi, and Arthur L. Liestman. A survey of gossiping and broadcasting in communication networks. *Networks*, 18(4):319–349, 1988.

[HKMP96] Juraj Hromkovič, Ralf Klasing, Burkhard Monien, and Regine Peine. Dissemination of information in interconnection networks (broadcasting & gossiping). In Ding-Zhu Du and D. Frank Hsu, editors, *Combinatorial Network Theory*, pages 125–212. Springer US, Boston, MA, 1996.

[HKR13] Maurice Herlihy, Dmitry N. Kozlov, and Sergio Rajsbaum. *Distributed Computing Through Combinatorial Topology*. Morgan Kaufmann, 2013.

[HLPR13] H. A. Harutyunyan, A. L. Liestman, J. Peters, and D. Richards. Broadcasting and gossiping). In J. Gross and P. Zhang, editors, *The Handbook of Graph Theory*, pages 1477–1494. Chapman and Hall/CRC, 2013.

[HMSZ09] Martin Hutle, Dahlia Malkhi, Ulrich Schmid, and Lidong Zhou. Chasing the weakest system model for implementing omega and consensus. *IEEE Transactions on Dependable and Secure Computing*, 6(4):269–281, 2009.

[HS99] Maurice Herlihy and Nir Shavit. The topological structure of asynchronous computability. *J. ACM*, 46(6):858–923, November 1999.

[IEE07] Ieee 802.11 standard: Wireless lan medium access control (mac) and physical layer (phy) specifications, June 2007. IEEE Computer Society LAN MAN Standards Committee.

[JT08] Prasad Jayanti and Sam Toueg. Every problem has a weakest failure detector. In *Proceedings of the twenty-seventh ACM symposium on Principles of distributed computing (PODC '08)*, pages 75–84, New York, NY, USA, 2008. ACM.

[KLO10] Fabian Kuhn, Nancy A. Lynch, and Rotem Oshman. Distributed computation in dynamic networks. In *STOC*, pages 513–522, 2010.

[KM07] Wolfgang Kiess and Martin Mauve. A survey on real-world implementations of mobile ad-hoc networks. *Ad Hoc Netw.*, 5(3):324–339, April 2007.

[KOM11] Fabian Kuhn, Rotem Oshman, and Yoram Moses. Coordinated consensus in dynamic networks. In *Proceedings of the 30th annual ACM SIGACT-SIGOPS symposium on Principles of distributed computing*, PODC '11. ACM, 2011.

[KRH18] Petr Kuznetsov, Thibault Rieutord, and Yuan He. An asynchronous computability theorem for fair adversaries. In *Proceedings of the 2018 ACM Symposium on Principles of Distributed Computing, PODC 2018, Egham, United Kingdom, July 23-27, 2018*, pages 387–396, 2018.

[KS06] Idit Keidar and Alex Shraer. Timeliness, failure detectors, and consensus performance. In *Proceedings of the twenty-fifth annual ACM SIGACT-SIGOPS symposium on Principles of Distributed Computing (PODC'06)*, pages 169–178, New York, NY, USA, 2006. ACM Press.

[Lam98] Leslie Lamport. The part-time parliament. *ACM Transactions on Computer Systems*, 16(2):133–169, May 1998.

[LHSP11] Franck Legendre, Theus Hossmann, Felix Sutton, and Bernhard Plattner. 30 years of wireless ad hoc networking research: What about humanitarian and disaster relief solutions? What are we still missing? In *International Conference on Wireless Technologies for Humanitarian Relief (ACWR 11)*, Amrita, India, 2011. IEEE.

[LLT13] Wendell A. Lim, Connie M. Lee, and Chao Tang. Design principles of regulatory networks: Searching for the molecular algorithms of the cell. *Molecular Cell*, 49(2):202 – 212, 2013.

[LM95] Ronit Lubitch and Shlomo Moran. Closed schedulers: A novel technique for analyzing asynchronous protocols. *Distrib. Comput.*, 8(4):203–210, June 1995.

[Lyn96] Nancy Lynch. *Distributed Algorithms*. Morgan Kaufman, 1996.

[MKSL04] Miklós Maróti, Branislav Kusy, Gyula Simon, and Ákos Lédeczi. The flooding time synchronization protocol. In *Proceedings of the 2nd International Conference on Embedded Networ ked Sensor Systems*, SenSys '04, pages 39–49, New York, NY, USA, 2004. ACM.

[MOZ05] Dahlia Malkhi, Florin Oprea, and Lidong Zhou. Ω meets paxos: Leader election and stability without eventual timely links. In *Proceedings of the 19th Symposium on Distributed Computing (DISC'05)*, volume 3724 of *LNCS*, pages 199–213, Cracow, Poland, 2005. Springer Verlag.

[MR99] Achour Mostéfaoui and Michel Raynal. Solving consensus using Chandra-Toueg's unreliable failure detectors: A general quorum-based approach. In P. Jayanti, editor, *Distributed Computing: 13th International Symposium (DISC'99)*, volume 1693 of *Lecture Notes in Computer Science*, pages 49–63, Bratislava, Slovak Republic, September 1999. Springer-Verlag GmbH.

[MR02] Yoram Moses and Sergio Rajsbaum. A layered analysis of consensus. *SIAM J. Comput.*, 31(4):989–1021, 2002.

[Mun00] James Munkres. *Topology (2nd Edition)*. Pearson, 2000.

[NBJ14] Saket Navlakha and Ziv Bar-Joseph. Distributed information processing in biological and computational systems. *Commun. ACM*, 58(1):94–102, December 2014.

[Now10] Thomas Nowak. Topology in distributed computing. Master thesis, Embedded Computing Systems Group, Technische Universität Wien, March 2010.

[NSW19] Thomas Nowak, Ulrich Schmid, and Kyrill Winkler. Topological characterization of consensus under general message adversaries. In *Proceedings of the 2019 ACM Symposium on Principles of Distributed Computing*, PODC '19, pages 218–227, New York, NY, USA, 2019. ACM.

[Pfl18] Daniel Pfleger. Knowledge and communication complexity. Master's thesis, Technische Universität Wien, Institut für Technische Informatik, Treitlstr. 3/2/182-2, 1040 Vienna, Austria, 2018.

[PS16] Daniel Pfleger and Ulrich Schmid. A framework for connectivity monitoring in wireless sensor networks. In *Proceedings 10th International Conference on Sensor Technlogies and Applications (SENSORCOMM'16)*, pages 40–48. IARIA, 2016. `https://www.thinkmind.org/download.php?articleid=sensorcomm_2016_3_10_10013`.

[PSL80] Marshall Pease, Robert Shostak, and Leslie Lamport. Reaching agreement in the presence of faults. *Journal of the ACM*, 27(2):228–234, April 1980.

[RRT08] Sergio Rajsbaum, Michel Raynal, and Corentin Travers. An impossibility about failure detectors in the iterated immediate snapshot model. *Inf. Process. Lett.*, 108(3):160–164, October 2008.

[RS13] Michel Raynal and Julien Stainer. Synchrony weakened by message adversaries vs asynchrony restricted by failure detectors. In *Proceedings ACM Symposium on Principles of Distributed Computing (PODC'13)*, pages 166–175, 2013.

[SBB12] Udo Schilcher, Christian Bettstetter, and Günther Brandner. Temporal correlation of interference in wireless networks with rayleigh block fading. *IEEE Transactions on Mobile Computing*, 11(12):2109–2120, 2012.

[Sch90] Fred B. Schneider. Implementing fault-tolerant services using the state machine approach: a tutorial. *ACM Comput. Surv.*, 22(4):299–319, 1990.

[Sch18] Manfred Schwarz. *Agreement algorithms in directed dynamic networks.* PhD thesis, TU Wien, Vienna, Austria, 2018. `http://katalog.ub.tuwien.ac.at/AC15078077`.

[SCS04] Jang-Ping Sheu, Chih-Min Chao, and Ching-Wen Sun. A clock synchronization algorithm for multi-hop wireless ad hoc networks. In *24th International Conference on Distributed Computing Systems, 2004. Proceedings.*, pages 574–581, 2004.

[SSW18] Ulrich Schmid, Manfred Schwarz, and Kyrill Winkler. On the strongest message adversary for consensus in directed dynamic networks. In Zvi Lotker and Boaz Patt-Shamir, editors, *Structural Information and Communication Complexity*, pages 102–120, Cham, 2018. Springer International Publishing.

[SW89] Nicola Santoro and Peter Widmayer. Time is not a healer. In *Proc. 6th Annual Symposium on Theor. Aspects of Computer Science (STACS'89)*, LNCS 349, pages 304–313, Paderborn, Germany, February 1989. Springer-Verlag.

[SWK09] Ulrich Schmid, Bettina Weiss, and Idit Keidar. Impossibility results and lower bounds for consensus under link failures. *SIAM Journal on Computing*, 38(5):1912–1951, 2009.

[SWS+14] Manfred Schwarz, Kyrill Winkler, Ulrich Schmid, Martin Biely, and Peter Robinson. Brief announcement: Gracefully degrading consensus and $k$-set agreement under dynamic link failures. In *Proceedings of the 33th ACM SIGACT-SIGOPS Symposium on Principles of Distributed Computing*, PODC '14, pages 341–343, New York, NY, USA, 2014. ACM.

[SWS16] Manfred Schwarz, Kyrill Winkler, and Ulrich Schmid. Fast consensus under eventually stabilizing message adversaries. In *Proceedings of the*

*17th International Conference on Distributed Computing and Networking*, ICDCN '16, pages 7:1–7:10, New York, NY, USA, 2016. ACM.

[TC84] R. Turpin and A. B. Coan. Extending binary Byzantine agreement to multivalued Byzantine agreement. *Information Processing Letters*, 18(2):73–6, February 1984.

[WJCD00] C. Ware, J. Judge, J. Chicharo, and E. Dutkiewicz. Unfairness and capture behaviour in 802.11 adhoc networks. In *2000 IEEE International Conference on Communications. ICC 2000. Global Convergence Through Communications.*, 2000.

[WS19] Kyrill Winkler and Ulrich Schmid. An overview of recent results for consensus in directed dynamic networks. *Bulletin of the European Association for Theoretical Computer Science EATCS*, (128):41–72, June 2019.

[WSS19] Kyrill Winkler, Manfred Schwarz, and Ulrich Schmid. Consensus in rooted dynamic networks with short-lived stability. *Distributed Computing*, Feb 2019.

[XFJ03] B. BUI XUAN, A. FERREIRA, and A. JARRY. Computing shortest, fastest, and foremost journeys in dynamic networks. *International Journal of Foundations of Computer Science*, 14(02):267–285, 2003.

[ZSS19] Martin Zeiner, Manfred Schwarz, and Ulrich Schmid. On linear-time data dissemination in dynamic rooted trees. *Discrete Applied Mathematics*, 255:307 – 319, 2019.