TU WIEN Informatics

Comprehensive Characterization of Consensus Solvability in Dynamic Networks with Transient Stability.

DIPLOMARBEIT

zur Erlangung des akademischen Grades

Diplom-Ingenieur

im Rahmen des Studiums

Logic and Computation

eingereicht von

Peter Pacheiner, BSc.
Matrikelnummer 01325582

an der Fakultät für Informatik

der Technischen Universität Wien

Betreuung: Univ.Prof. Dr. Ulrich Schmid

Wien, 26. September 2023

Peter Pacheiner	Ulrich Schmid

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

TU WIEN Informatics

Comprehensive Characterization of Consensus Solvability in Dynamic Networks with Transient Stability

DIPLOMA THESIS

submitted in partial fulfilment of the requirements for the degree of

Diplom-Ingenieur

in

Logic and Computation

by

Peter Pacheiner, BSc.
Registration Number 01325582

to the Faculty of Informatics

at the TU Wien

Advisor: Univ.Prof. Dr. Ulrich Schmid

Vienna, 26th September, 2023

Peter Pacheiner	Ulrich Schmid

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

Erklärung zur Verfassung der Arbeit

Peter Pacheiner, BSc.

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

Wien, 26. September 2023

Peter Pacheiner

Danksagung

Hier möchte ich meiner Verlobten Emma und meinem Professor danken, die mich beide beim Verfassen dieser Arbeit unterstützt haben und endlose Mengen an Geduld für mich und meine Schwierigkeiten hatten.

Acknowledgements

I want to thank both my fiancée Emma and my professor, the two people that supported me without fail throughout the writing of this thesis and had far more patience with me than I thought possible.

Kurzfassung

Diese Masterarbeit entwickelt eine vollständige Charakterisierung des Konsensusproblems in dynamischen Netzwerken mit kurzlebiger Stabilität. Während die existierende Literatur dieses Problem nur unter spezifischen Parametern untersucht hat, wird in dieser Arbeit ein allgemeines Setting vorgestellt, vollständig charakterisiert, und gezeigt wie es sich zu alternativen Modellen aus der existierenden Literatur verhält. Hierfür mussten sowohl existierende Beweise untersucht und überarbeitet, als auch neue Beweise formuliert werden, um bislang ungelöste Fälle korrekt zu klassifizieren.

Abstract

This thesis establishes a comprehensive characterization of consensus in dynamic networks with transient stability. In the existing literature, this problem has only been studied for specific parameters and slightly different models. In this thesis, a unified setting is proposed, completely characterized and shown how it relates to the existing alternative models in the literature. This required revisiting and adapting existing proofs, as well as developing new ones for the cases that were unsolved before.

Contents

CHAPTER 1

Introduction

This thesis deals with deterministic consensus algorithms in synchronous dynamic networks, where a potentially unknown number of processes, which never fail, communicate via unacknowledged messages over unreliable, unidirectional point-to-point links. These links are under the control of an omniscient message adversary.
Consensus, which is a pivotal service in truly distributed applications, is the problem of computing a common decision value based on local input values of all the processes. Examples witnessing the importance of such agreement algorithms can be found both on the macro and micro scale, ranging from distributed servers that need to confirm whether or not a transaction really did occur down to several sensors that are used to achieve fault tolerance in hostile environments.
An execution of a consensus algorithm in our system proceeds in a sequence of lock-step synchronous rounds, where message loss is modelled using an omniscient message adversary that determines the directed communication graph G^r for each round r. A directed edge $(p \to q)$ present in G^r means that the message sent by p in round r is successfully received by q in the same round.
While most existing work in this area allows the message adversary to choose each round r from the same set of admissible graphs, a few papers utilize the notion of an eventually stabilizing message adversary, were the set of admissible choices for G^r may change with evolving round numbers r. This allows for the modelling of unstable periods e.g. during start up or due to heavy interference. As this adds a non-trivial liveness condition to the message adversary specification, solving consensus is much more difficult in such a system.

1.1 Related Work

The study of consensus in distributed systems dates back at least to the well known paper by Fischer, Laynch and Patterson [FLP85].

While [FLP85] focused on consensus in asynchronous systems, with processes that can crash, [SW89] studied consensus in synchronous message-passing systems with transmission faults. Later [AG13] introduced the term message adversary, for synchronous system in which the communication graph for each round is chosen from a set of possible graphs. For message adversaries that choose from a fixed set of options, [CGP15] introduced the term oblivious message adversary.
Kuhn, Oshman, Lynch and Moses' research on dynamic networks with bidirectional links goes back at least to [KO11, KLO10, KOM11], which primarily aimed at simulating (bounded) faults in the network. Since then, dynamic networks have grown into their own field, showing themselves to quite effectively model wireless mobile ad hoc networks, where packets from different sources may collide or otherwise interfere destructively with one another.
Whereas solving consensus in bidirectional dynamic networks is easy, it is known since the seminal work by Santaro and Widmayer [SW89] that this is not the case in directed dynamic networks. In fact, deciding whether consensus is solvable for a given oblivious message adversary is not easy [CGP15], and practical algorithms are lacking [WSM20].
Eventually stabilizing message adversaries were first introduced in [BRS+16], which assumed communication graphs with a non-empty set of sources and long periods of stability. These message adversaries were further studied in [BRS+18] [SWS16] [WSS19], which constitute the basis for this thesis: Especially noteworthy is [WSS19], which deals with message adversaries in rooted networks, and [SWS16], which introduces the transient stability message adversary studied in this thesis (See Definition 19 for its definition and properties).

1.2 Goal of the thesis

The goal of this thesis is a complete characterization of the parameters of the transient stability message adversary MA(x, y, D) (see Definition 19) [SWS16], providing concise proofs under which parameters consensus is possible/impossible. Whereas several papers exist that characterized it partly [WSS19] [SWS16] [WSS16], they leave several instances of parameter settings unexplored (see Figure 1.1 - Here the red area is proven to be unsolvable, while for the thin green line an algorithm exists that can solve consensus. For the yellow area it is currently unknown if consensus is solvable).
This thesis completely explores these areas and provide proofs filling in the (previously unknown) yellow areas. (see Figure 4.1 - Figure 4.2). Additionally, an answer and proof for an open question in [WSS19] is also provided.

1.3 Outline

Chapter 2 provides an explanation of the model of computation we are using, as well as an explanation of the consensus problem, the properties of dynamic networks and the

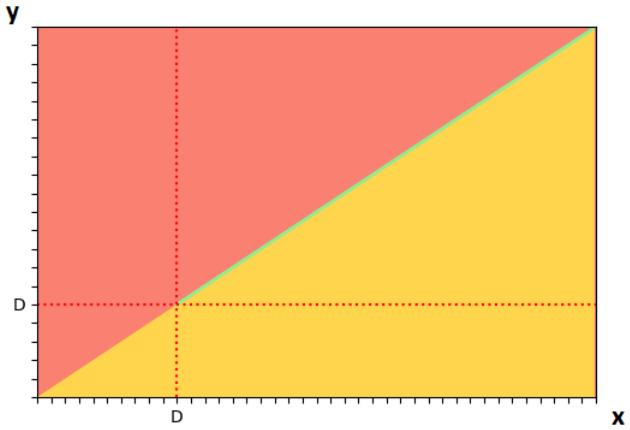


Figure 1.1: State of the Art of consensus solvability under MA(x, y, D) according to [SWS16]. While for the red/green areas it is proven that consensus is impossible/solvable, for the yellow area no such proofs existed.

different basic definitions employed throughout this thesis. Additionally, it formally (re-) introduces our setting of *transient stability* and specifies the transient stability message adversary. The following Chapter 3 then completely characterizes consensus solvability/impossibility for all parametrizations, using indistinguishable chains of executions for proving impossibility. In Chapter 4, the results are summarized, and an outlook of further possible research is provided.

CHAPTER 2

Model of Computation

2.1 Distributed Systems

The model used in this paper is mostly based on [AW04], with additional inspiration taken from [WSS19] [SWS16] and [WSS16]. As most of these definitions are assumed to be fundamental to the field of distributed computing, they are not explicitly referenced in this thesis. Still, the definitions of the message adversaries are cited to support cross-referencing.

We model our synchronous message passing system as a set Π of $|\Pi| = n > 1$ deterministic state machines called *processes*. While our processes are not aware of n (or even a bound on it), they have unique identifiers that we pick w.l.o.g. from the set $\{1, ..., n\}$. Throughout this thesis processes are labeled as either p_i, q_i, x_i, depending on which set of processes they belong to. Here, the parameter i is not to be taken as the global unique ID of the process but only of its designation within the appropriate set.
These processes operate in lock-step rounds: In each round $r \geqslant 1$, every process can send messages to all other processes followed by every process taking an immediate and instantaneous computation step, where based on its current local state and all messages it just received, a successor state and the messages to be sent in the next round are computed.
Whether or not a message reaches its intended recipient depends on the round-r communication graph $G^r = (V, E^r)$. Each vertex in V corresponds to one process from Π, and an edge in E^r, denoted as $(p \to q)$, represents the fact that the message sent from p to q is not lost. We assume that G^r always contains all self-loops $(p \to p)$, i.e., processes always receive their own messages.
Note: these self-loops are never displayed in the figures but always assumed to be present.

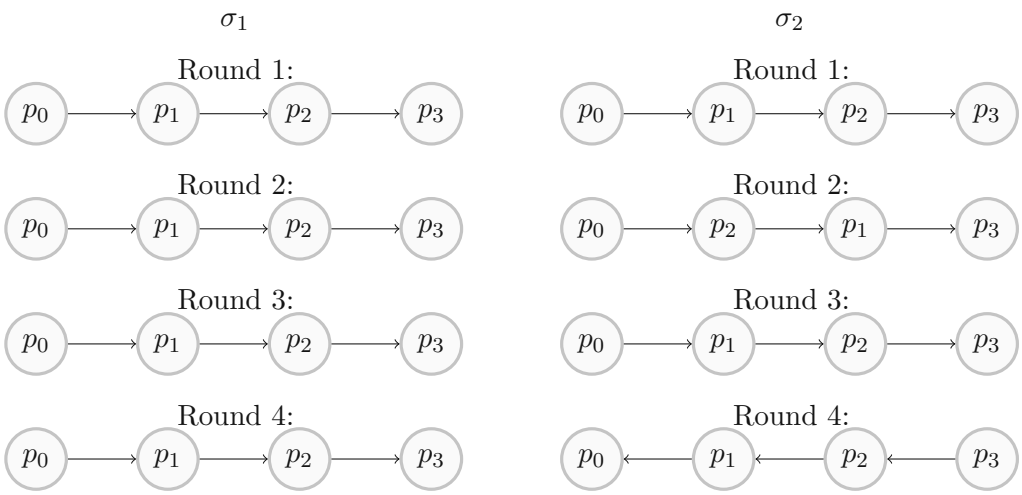


Figure 2.1: Static communication graph vs dynamic communication graph.

2.1.1 Dynamic Networks and Message Adversaries

In dynamic networks, the communication graph can be different in different rounds, resulting in a change from a static communication graph $G^r = G = (V, E)$ to a graph sequence $(G^r)_{r=1}^{\infty} = (V, E^r)$, where G^i is the communication graph for the synchronous round i. Figure 2.1 provides an example of this notion. While the graph on the left side remains static, the connections on the right-side change, leading to a reordering of the nodes in the resulting chain graphs.

To encapsulate the notion that an algorithm must work even in some worst-case scenario, we give control of this graph sequence to a **message adversary** [AG13]. Message adversaries have been recognized as a useful approach for modelling different executions of a distributed system dictated by the environment.

In [KO11], three different message adversaries are presented for dynamic networks:

- *oblivious worst-case adversary*: chooses the sequence of communication graphs at the start of the execution (Note that the meaning of oblivious is different in [CGP15], where it characterizes a message adversary that may choose any graph from a given set of graphs in every round).
- *adaptive worst-case adversary*: chooses the next communication graph based on the current state of the processes.
- *random-graph adversary*: chooses the next communication graph randomly.

For deterministic algorithms, the *oblivious worst-case adversary* and the *adaptive worst-case adversary* are equivalent.

Formally, we identify a message adversary by a set of graph sequences, which are called

admissible for this message adversary. The set of admissible graph sequences in turn is defined by the restrictions placed upon the message adversary by the environment. In order to be useful in practice, they should comprise all graph sequences that can occur in the modelled environment but should not make the problem at hand (consensus in our case) impossible to solve.

The following definitions describe general properties of graph sequences and will be used in later definitions.

2.1.2 Root Components

A root component of a graph is a strongly connected component without any incoming edges. By construction at least one root component will always exist in every graph.

Definition 1 (Root component [SWS16, Definition 1]). *A non-empty set of nodes $R \subseteq V$ is called a round r root component of $\mathcal{G}^r$, if it is the set of vertices of a strongly connected component $\mathcal{R}$ of $\mathcal{G}^r$ and $\forall p \in \mathcal{G}^r, q \in R : (p \to q) \in \mathcal{G}^r \Rightarrow p \in R$. We denote by* $\text{roots}(\mathcal{G}^r)$ *the set of all root components of $\mathcal{G}^r$, resp. the single root component of $\mathcal{G}^r$, and by $|R|$ the number of nodes in R.*

We define a graph sequence as a set of consecutive communication graphs. If a root R exists in all communication graphs of a graph sequence we refer to it as a common root of the sequence. If a graph sequence with a common root R can not be extended without R ceasing to be a common root, then R is the maximal common root of the graph sequence. It is important to note that the internal interconnectivity of a root component does not matter beyond the fact that it is strongly connected.

Definition 2 (Common root [SWS16, Definition 2]). *We say that a sequence $(\mathcal{G}^r)_{r\in I}$ has a common root R, iff there exists a root R (with possibly different interconnect topology) such that $R \in \text{roots}(\mathcal{G}^r)$ for all $r \in I$. If $I = [a, b]$ with $|I| = b - a + 1$ is an interval of consecutive rounds $a, a + 1, \ldots, b$, $(\mathcal{G}^r)_{r\in I}$ is called a consecutive graph sequence. We call R a maximal common root of a consecutive graph sequence $(\mathcal{G}^r)^b_{r=a}$, iff R is a common root of $(\mathcal{G}^r)^b_{r=a}$ but neither of $(\mathcal{G}^r)^b_{r=a-1}$ nor $(\mathcal{G}^r)^{b+1}_{r=a}$.*

In turn, we define a single-rooted sequence as a graph sequence that only contains a single root.

Definition 3 (Single-rooted sequence [SWS16, Definition 3]). *We call a sequence $(\mathcal{G}^r)_{r\in I}$ single-rooted, or R-single-rooted, if there exists a unique root component R s.t. $\forall i, j \in I : \text{roots}(\mathcal{G}^i) = \text{roots}(\mathcal{G}^j) = R$. We call R a maximal single root of a consecutive graph sequence $(\mathcal{G}^r)_{r\in I}$ with $I = [a, b]$, iff R is a single root of $(\mathcal{G}^r)^b_{r=a}$ but neither of $(\mathcal{G}^r)^b_{r=a-1}$ nor $(\mathcal{G}^r)^{b+1}_{r=a}$.*

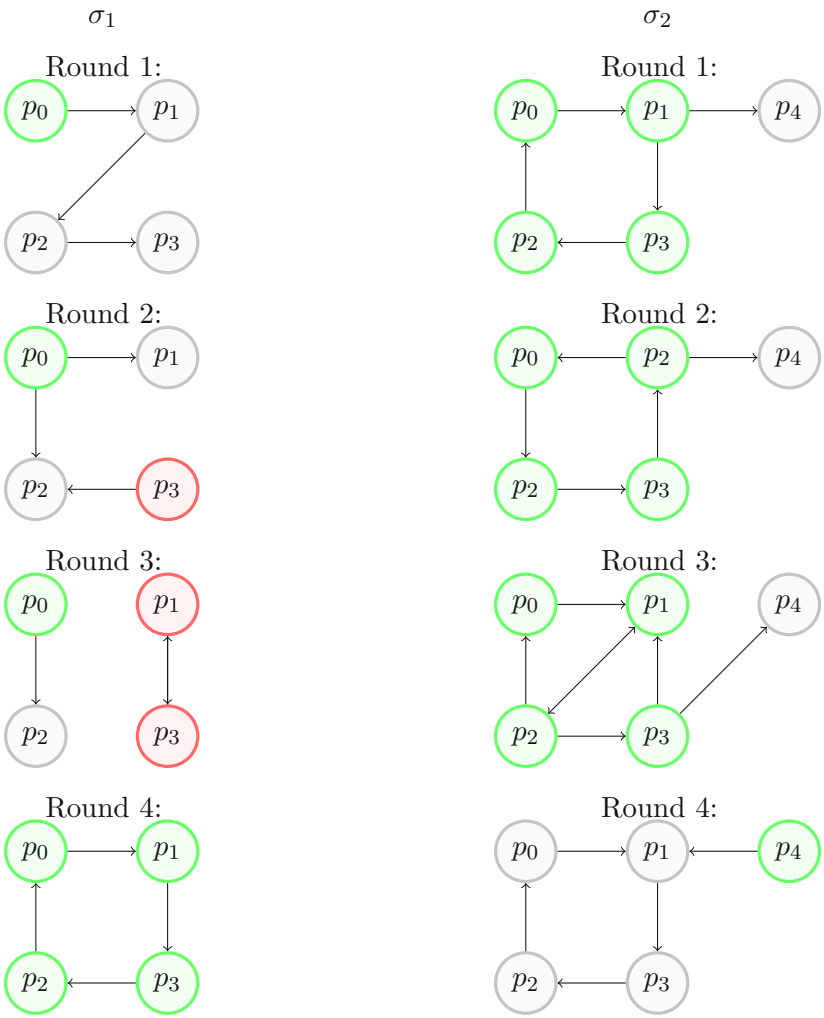


Figure 2.2: Rooted graph sequences.

This property naturally leads to the graph being at least weakly connected, since otherwise more than a single root would exist.

Corollary 4 ([SWS16, Corollary 1])**.** *For any directed graph* $\mathcal{G}^r$*,* $|\text{roots}(\mathcal{G}^r)| \geqslant 1$*, and if* $|\text{roots}(\mathcal{G}^r)| = 1$*, then* $\mathcal{G}^r$ *is weakly connected.*

Figure 2.2 shows an example of two graph sequences, with the (different) root components coloured. On the right-side the set $\{p_1, p_2, p_3, p_4\}$ is a maximal single root of the graph sequence $(\mathcal{G}^r)_{r=1}^3$ but only a single root of the graph sequence $(\mathcal{G}^r)_{r=2}^3$, while for the left sequence the set consisting of the single node p_0 is a maximal common root of the graph sequence $(\mathcal{G}^r)_{r=1}^3$ but only a maximal single root of the graph sequence $(\mathcal{G}^r)_{r=1}^1$.

2.1.3 Causal past

The definition of causal past is strongly related to the propagation of information. Given some round b, the process $p's$ causal past $\mathrm{CP}_p^b(a)$ down to round $a \geqslant 0$, are exactly the processes who were able to propagate the information they held at the end of round a to p at the end of round b. Note that the information of p held at the end of round 0 is its input value.

Definition 5 (Causal past [SWS16, Definition 4])**.** *For a given infinite sequence σ of communication graphs, we define the causal past $\mathrm{CP}_p^b(a)$ of process p from (the end of) round b down to (the end of) round a as $\mathrm{CP}_p^b(b) := \emptyset$ and for $0 \leqslant a < \ell \leqslant b$, $\mathrm{CP}_p^b(\ell - 1) := \mathrm{CP}_p^b(\ell) \cup \{q \in \Pi \mid \exists q' \in (\{p\} \cup \mathrm{CP}_p^b(\ell)) : (q \to q') \in \mathcal{G}^\ell\}$.*

Figure 2.3 shows the evolution of the causal past of different nodes over time. For the proofs presented later on, we are mostly interested in $\mathrm{CP}_p^T(0)$, so the maximum information about the systems initial configuration a single node could have received until the end of round T.

2.1.4 Dynamic Network Diameter(s)

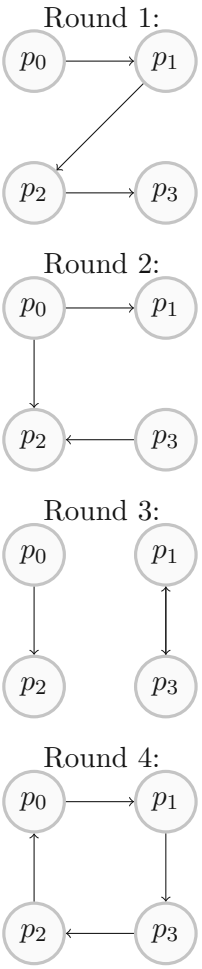
The papers [SWS16] and [WSS19] introduced a slightly varying concepts of the dynamic network diameter - a property which describes how information propagates in graph sequences containing single-rooted subsequences. While [SWS16] allows for the round r of the communication graph $\mathcal{G}^r$ with a fixed single root R to be non-consecutive, [WSS19] deals with consecutive single-rooted-communication graphs only.

In the non-consecutive case, the dynamic diameter *nc-D* tells how often a specific single root must (at most) reoccur until information from it has propagated to all other processes.

Definition 6 (Non-consecutive dynamic diameter $Nc\text{-}D(D)$ [SWS16, Definition 5])**.** *A message adversary* `MA` *guarantees a non-consecutive dynamic (network) diameter of D, if for every graph sequence $\sigma \in$ `MA` that contains a subsequence $S = (\mathcal{G}^{r_1}, \ldots, \mathcal{G}^{r_D})$ of D not necessarily consecutive R-single-rooted communication graphs, it holds that $R \subseteq \mathrm{CP}_p^{r_D}(r_1 - 1)$ for every $p \in \Pi$.*
$nc\text{-}D(D)$ is the set of all graph sequences that guarantee a non-consecutive dynamic (network) diameter of D.

In contrast, the consecutive dynamic diameter *c-D* only enforces propagation of information if the single-rooted communication graphs occur consecutively.

Definition 7 (Consecutive dynamic diameter $C\text{-}D(D)$ [WSS19, Definition 4])**.** *A message adversary* `MA` *guarantees a consecutive dynamic (network) diameter D, if for every graph*



$\mathrm{CP}^1_{p_0}(0) = \{p_0\}$
$\mathrm{CP}^1_{p_1}(0) = \{p_0, p_1\}$
$\mathrm{CP}^1_{p_2}(0) = \{p_1, p_2\}$
$\mathrm{CP}^1_{p_3}(0) = \{p_2, p_3\}$

-

$\mathrm{CP}^2_{p_0}(1) = \{p_0\}$
$\mathrm{CP}^2_{p_1}(1) = \{p_0, p_1\}$
$\mathrm{CP}^2_{p_2}(1) = \{p_0, p_2, p_3\}$
$\mathrm{CP}^2_{p_3}(1) = \{p_3\}$

$\mathrm{CP}^2_{p_0}(0) = \{p_0\}$
$\mathrm{CP}^2_{p_1}(0) = \{p_0, p_1\}$
$\mathrm{CP}^2_{p_2}(0) = \{p_0, p_1, p_2, p_3\}$
$\mathrm{CP}^2_{p_3}(0) = \{p_2, p_3\}$

$\mathrm{CP}^3_{p_0}(2) = \{p_0\}$
$\mathrm{CP}^3_{p_1}(2) = \{p_1, p_3\}$
$\mathrm{CP}^3_{p_2}(2) = \{p_0, p_2\}$
$\mathrm{CP}^3_{p_3}(2) = \{p_1, p_3\}$

$\mathrm{CP}^3_{p_0}(0) = \{p_0\}$
$\mathrm{CP}^3_{p_1}(0) = \{p_0, p_1, p_2, p_3\}$
$\mathrm{CP}^3_{p_2}(0) = \{p_0, p_1, p_2, p_3\}$
$\mathrm{CP}^3_{p_3}(0) = \{p_0, p_1, p_2, p_3\}$

$\mathrm{CP}^4_{p_0}(3) = \{p_0, p_2\}$
$\mathrm{CP}^4_{p_1}(3) = \{p_0, p_1\}$
$\mathrm{CP}^4_{p_2}(3) = \{p_2, p_3\}$
$\mathrm{CP}^4_{p_3}(3) = \{p_1, p_3\}$

$\mathrm{CP}^4_{p_0}(0) = \{p_0, p_1, p_2, p_3\}$
$\mathrm{CP}^4_{p_1}(0) = \{p_0, p_1, p_2, p_3\}$
$\mathrm{CP}^4_{p_2}(0) = \{p_0, p_1, p_2, p_3\}$
$\mathrm{CP}^4_{p_3}(0) = \{p_0, p_1, p_2, p_3\}$

Figure 2.3: Causal past of an example graph sequence.

sequence $\sigma \in \mathtt{MA}$ *that contains a subsequence* $S = (\mathcal{G}^{r_1}, \ldots, \mathcal{G}^{r_D})$ *of* D *consecutive* R*-single-rooted communication graphs, it holds that* $R \subseteq \mathrm{CP}^{r_D}_p(r_1 - 1)$ *for every* $p \in \Pi$. *c-*$D(D)$ *is the set of all graph sequences that guarantee a consecutive dynamic (network) diameter of* D.

For better understanding, Figure 2.4 is provided: It shows a static chain graph sequence with a diameter of three on the left-hand side, a graph sequence with a non-consecutive dynamic diameter *nc-*$D = 2$ in the middle, and a graph with a consecutive dynamic network diameter *c-*$D = 2$ on the right-hand side. The roots in each graph are marked by a double border for improved visibility.
The green colour shows how far the information of the input value of R has propagated at the beginning of the respective round.

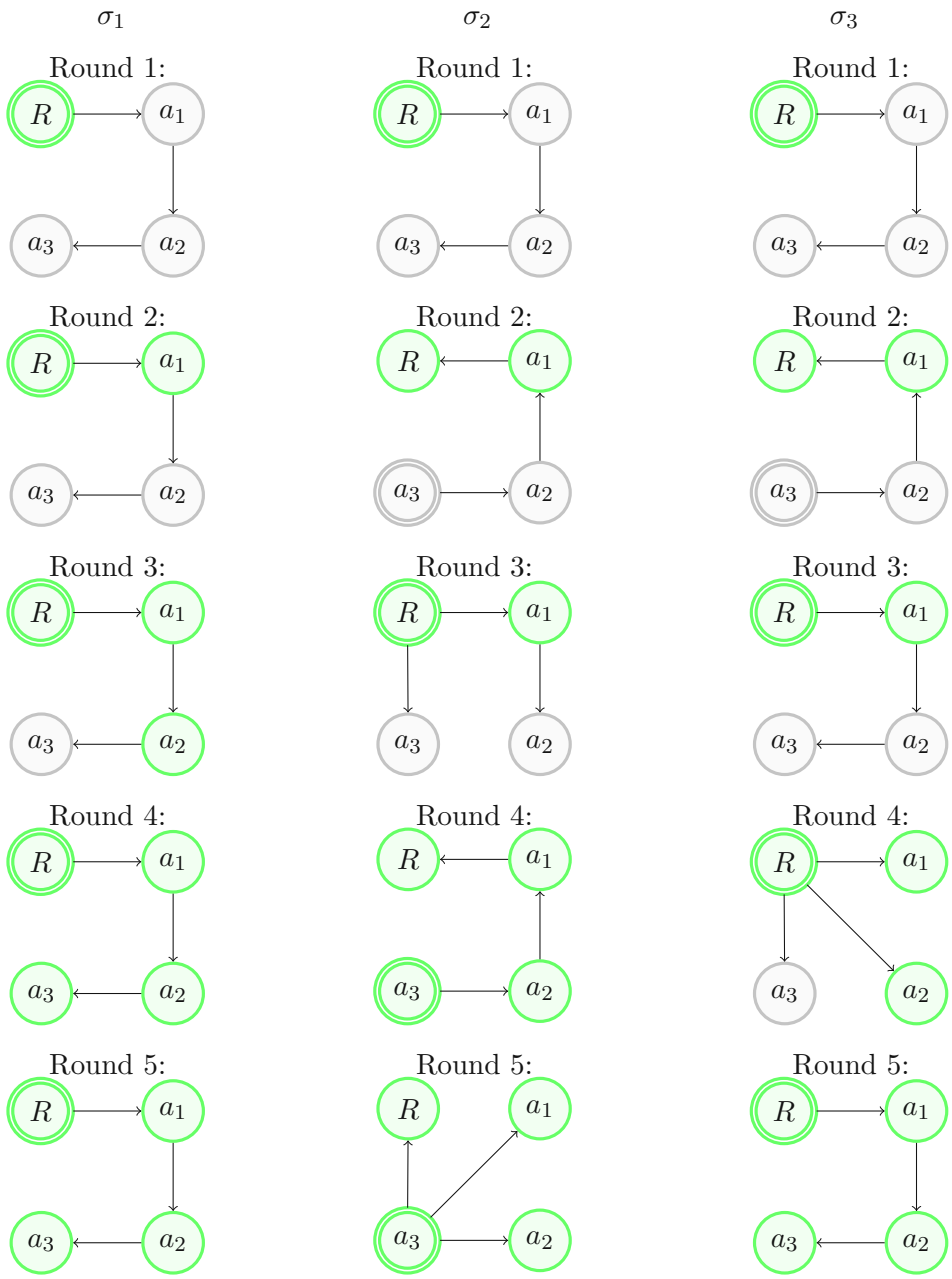


Figure 2.4: Graph sequences with static diameter $D = 3$ vs. non-consecutive diameter $nc\text{-}D = 2$ vs consecutive diameter $c\text{-}D = 2$. Depicted in green is the propagation of the input value of R over time (rounds).

2.2 Consensus Problem

In the consensus problem, all processes p_i have an input or initial value x_i of some kind (where $x_i \neq \perp$) and a decision value y_i, which once assigned cannot be changed and is initially set to $\perp$. An algorithm solving the consensus problem, in our (fault-free) setting, must satisfy the following three properties:

- **termination**: every process will assign a value to y_i eventually.
- **agreement**: no two processes p_i, p_j differ on the non-$\perp$ value assigned to y_i, y_j.
- **validity**: if all processes p_i start with the same value $x_i = x$, then $y_i = x$ for all processes.

In the case of binary consensus (which we are using for our proofs), where every input value x_i can only either be 0 or 1, our validity condition is equivalent to strong validity, which requires that the decision value is the input value of some process.

Note that the combination of **validity** and **agreement** lends itself to indistinguishability proofs. For example, it is enough to show that a processor only ever learns about one input value x, to show that this processor (by **validity**) and all other processors (by **agreement**) have to decide on x.
This is also where the previous notion of the causal past comes into play: the input values x_i process p_j learns about until round T are exactly the input values of the processes in $\mathrm{CP}_p^T(0)$.

2.3 Indistinguishability Proofs

A configuration C^r is the set of all local states of all processes $p_i \in |\Pi|$ at the end of round r. C^0 is the initial configuration the beginning of an execution α. An execution α in our model is an alternating sequence of configurations C^r and communications graphs $\mathcal{G}^r$, starting with the initial configuration C^0. An execution is called admissible under a message adversary iff the sequence of communication graphs used in it is admissible under the message adversary.

For **deterministic** algorithms, it holds that the initial configuration C^0, together with the sequence of communication graphs $\sigma \in (G^r)_{r\geqslant 1}$, fully defines all configurations in the execution α, i.e., $C^0 + (G^r)_{r\geqslant 1} \Rightarrow (C^r)_{r\geqslant 1}$. In this thesis we use α_i to denote an execution, C_i^0 to denote the corresponding initial configuration and σ_i to denote the corresponding graph sequence $(\alpha_i = (C_i^0; \sigma_i))$.

The set of decision values that can be reached from a particular configuration is called its *valence*. It is equivalent to the set of all values that are decided upon by a processor in some configuration that is reachable from C in an admissible execution that includes C. By the **termination** condition, this set cannot be empty. C is *univalent* if this set contains only one value; it is *0-valent* if this value is 0 and *1-valent* if this value is 1. If the set contains both values, then C is *bivalent* [AW04].

In our model we extend this notion of valence to executions. An execution is *1-valent* iff all processes in it will decide on the value 1. If consensus is solvable under a message adversary, it implies that all executions admissable under the message adversary are univalent.

Two graph sequences σ_1, σ_2 are called indistinguishable for some process p_i, denoted as $\sigma_1 \overset{p_i}{\sim} \sigma_2$, if the the causal past of p_i is the same in both at least until some round T when p_i decides on a value.

To prove the non-existence of an algorithm that solves consensus under a message adversary, it is enough to show that two executions α_1, α_2 exist such that:

- their graph sequences σ_1, σ_2 are both admissible under the message adversary
- their graph sequences σ_1, σ_2 are indistinguishable for some process p_i ($\sigma_1 \overset{p_i}{\sim} \sigma_2$)
- all processes $p \in \mathrm{CP}_{p_i}^{T}(0)$ have the same input values in both executions α_1 and α_2
- p_i decides on 1 in α_1 and on 0 in α_2

Figure 2.5 is provided as an example of such a proof. The left execution α_1 is *1-valent* (by **validity**, as the input values of all processes are assumed to be 1), while the right execution α_3 is *0-valent* (again by **validity**, as the input values of all processes are assumed to be 0). Assuming that p_0 decides (on 1) in α_1 in round T (whenever that might be), it will (by **determinism**) also decide on 1 in α_2 in round T (as $\sigma_1 \overset{p_0}{\sim} \sigma_2$). From this follows (by **agreement**) that in α_2 all processes will decide 1 as well. However the same argument can be used for some round T' and the process q_0 in α_3, which causes all processes in α_2 to decide on 0. Therefore, by **validity**, **determinism** and **agreement** all processes in α_2 would need to decide on both 0 and 1 simultaneously. This contradiction proves that either one of these executions is inadmissible under the choosen message adversary, or that the assumption that there is an algorithm that always solves consensus under the choosen message adversary is wrong.

In Figure 2.5, as well as in all subsequent figures, we use the following notation:

- Each vertex (representing a single process) is coloured corresponding to its input value (with input value of 0 resulting in the colour red).
- Root components are marked by having a double border.
- All processes not displayed are assumed to be directly connected to all roots and to have no outgoing edges.

2.4 Message Adversaries

In this section, we first discuss the message adversaries introduced in the papers [WSS19] [SWS16] [WSS16], how they differ and relate to each other. Recall that message adversaries are defined as a set of possible graph sequences they can generate. Therefore, the mathematical notion of subset relations can be used to compare the power of two

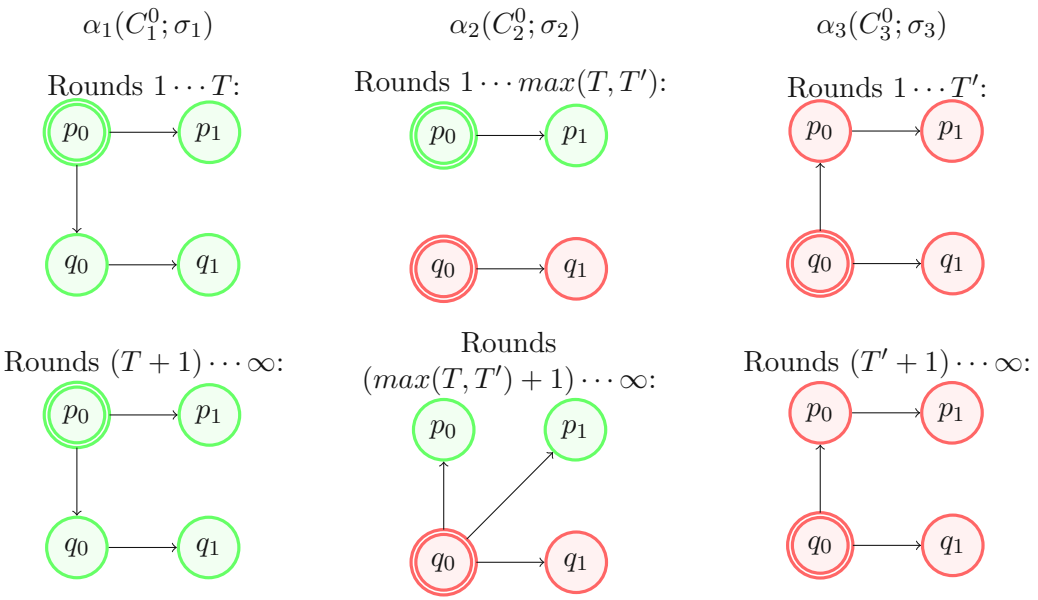


Figure 2.5: Example of how a $\Diamond$`FAES-STABILITY` adversary can prevent consensus, by having multiple roots for an arbitrary amount of time without any guarantee of ever changing. In the initial configuration C_1^0 resp. C_3^0, all processes start with the input value 1 resp. 0. In C_2^0 process p_0 resp. q_0 have input value 1 resp. 0. Since $\sigma_1 \overset{p_0}{\sim} \sigma_2$, process p_0 must decide on 1 in σ_2. Similarly, as $\sigma_2 \overset{q_0}{\sim} \sigma_3$, process q_0 must decide on 0 in σ_2, which violates **agreement**.

adversaries.

Definition 8. *A message adversary MA_j is as strong or stronger than another MA_i, denoted $MA_i \subseteq MA_j$, iff all graph sequences σ that are in MA_j are also in MA_i.*

Since sets can be incomparable, message adversaries can also be incomparable. The message adversaries discussed in this section are constructed by properties they have to guarantee, which in turn are sets of allowed graph sequences that are combined via set intersections.

2.4.1 FAES-Message Adversary

The FAES-(forever-after-eventually-single)-adversary [SWS16] is a message adversary that models an unbounded (but finite) initial period of "chaotic" behavior, where the communication graphs can be arbitrary. After some (unknown) round r_{stab}, the message adversary "stabilizes" and guarantees a permanent ("stable") single-rooted graph sequence.

This kind of of definition is well-known from eventual-type models in distributed computing, like some partially synchronous systems [DLS88].

The property ◊FAES-STABILITY is the liveness property of the FEAS-adversary. Liveness properties encapsulate the notion that "something good" will eventually happen. In this case, it is the property that after some unknown round r_{stab} the message adversary has to allow a root component to "stabilize" and eventually even become the single root component of the respective graph.

Definition 9 (◊FAES-STABILITY [SWS16, Definition 6])**.** *We say that* $(\mathcal{G}^r)_{r=1}^{\infty}$ *has a (unique)* FAES-common root R *("forever after, eventually single") starting at round* $r_{stab} \geqslant 1$*, iff* R *is (i) a maximal common root of* $(\mathcal{G}^r)_{r=r_{stab}}^{\infty}$ *and (ii) a maximal single root of* $(\mathcal{G}^r)_{r=r_{sr}}^{\infty}$*, for some round* $r_{sr} \geqslant r_{stab}$*.* ◊FAES-STABILITY *contains those communication graph sequences* $(\mathcal{G}^r)_{r=1}^{\infty}$ *that have a* FAES*-common root* R*.*

Since the ◊FAES-STABILITY property is quite weak and would allow a message adversary to deceive the processes by presenting multiple roots for an arbitrary time (an example of this shown in σ_2 in the rounds $1...T$ in Figure 2.5), a safety property is added. Safety properties encapsulate the notion that "nothing bad" ever happens. FAES-STICKY is the property that a root that is stable for some number of rounds is the FAES-common root of the graph sequence.

Definition 10 (FAES-STICKY(y) [SWS16, Definition 7])**.** FAES-STICKY(y) *contains those communication graph sequences* $\sigma = (\mathcal{G}^r)_{r=1}^{\infty}$*, where every root* R *that is common for* $> y$ *consecutive rounds in* σ *is the* FAES*-common root* R *in* σ*.*

Additionally, the FEAS-adversary also needs to guarantee some non-consecutive dynamic diameter D.

Definition 11 (◊FAES-STABLE [SWS16, Definition 8])**.** *The message adversary* ◊FAES-STABLE(D) = FAES-STICKY(D)+◊FAES-STABILITY+nc-$D(D)$ *contains the graph sequences* FAES-STICKY$(D) \cap$ ◊FAES-STABILITY $\cap$ nc-$D(D)$*.*

Notes:

- While [SWS16] originally only introduced the ◊FAES-STABLE(D)-adversary, one can derive a generalized version from their definitions: ◊FAES-STABLE(y, D) = FAES-STICKY(y) + ◊FAES-STABILITY + nc-$D(D)$.
- Note that [SWS16] implicitly also considered a variant of the ◊FAES-STABLE(D)-adversary where the non-consecutive dynamic diameter nc-$D(D)$ has been replaced by the (weaker) consecutive dynamic diameter c-$D(D)$.

2.4.2 ECS-Message Adversary

The ECS-(embedded-consecutive-single)-adversary [SWS16] is a relaxation of the FAES-adversary. Instead of modelling a permanent stabilization, it models a stable period that is long enough to allow the dissemination of information throughout the system, followed by a slightly less chaotic phase. It is strictly stronger than the FAES-adversary.

First, we define an ECS(x)-common root as a common root for some graph sequence, which is also single for at least x consecutive rounds during this sequence.

Definition 12 ($\texttt{ECS}(x+1)-$*commonroot* [SWS16, Definition 9])**.** *We say that a graph sequence $(G^r)_{r=\alpha}^{\alpha+d}$ has an ECS($x+1$)-common root ("embedded x+1-consecutive single common root") R, if (i) $(G^r)_{r=\alpha}^{\alpha+d}$ has a common root R and (ii) $(G^r)_{r=\alpha'}^{\alpha'+x} \subseteq (G^r)_{r=\alpha}^{\alpha+d}$ has a single root R.*

The liveness property of the ECS-adversary defines that the message adversary must allow for the existence of a sufficiently long ECS-common root, which later reappears often enough to propagate the information that it was a ECS-common root to all processes.

Definition 13 ($\Diamond\texttt{ECS-STABILITY}(x)$ [SWS16, Definition 10])**.** *Every communication graph sequence $\sigma \in \Diamond\texttt{ECS-STABILITY}(x)$ contains a subsequence $(\mathcal{G}^r)_{r=\alpha}^{\alpha+d}$, which has a $\texttt{ECS}(x+1)$-common root R; let $r_{stab} = \alpha$ be its starting round and $r_{sr} = \alpha'$ be the time when it becomes single. Furthermore, there are at least D, not necessarily consecutive, R-single rooted round graphs $\mathcal{G}^{r_1}, \ldots, \mathcal{G}^{r_D}$ with $r_{sr} + x < r_1 < \cdots < r_D$ in σ.*

Since once again the message adversary could trick processes into believing they have seen an ECS-common root, the safety property $\texttt{ECS-STICKY}(y)$ states that the first root that was common for at least $y+1$ rounds will be our $\texttt{ECS}(y+1)$-common root. This allows the processes to safely decide on whichever value the root was proposing, once they detect that is was the earliest common root for $x+1$ rounds.

Definition 14 ($\texttt{ECS-STICKY}(y)$ [SWS16, Definition 11])**.** *For every $\sigma \in \texttt{ECS-STICKY}(y)$, it holds that the earliest subsequence in σ with a maximal common root R in at least $y+1$ consecutive rounds actually has a $\texttt{ECS}(y+1)$-common root.*

In contrast to the FAES-adversary, this construction is tied to a non-consecutive dynamic diameter *nc-D*, as Definition 13 is explicitly structured around this property. A swap to the consecutive dynamic diameter would in turn necessitate a reformulation of $\Diamond\texttt{ECS-STABILITY}(x)$, which would not necessarily lead to message adversary that could be classified as strictly weaker or stronger than the original message adversary $\Diamond\texttt{ECS-STABLE}(D)$.

Definition 15 ($\Diamond\texttt{ECS-STABLE}(D)$ [SWS16, Definition 12])**.** *The strong stabilizing message adversary $\Diamond\texttt{ECS-STABLE}(D) = \texttt{ECS-STICKY}(D) + \Diamond\texttt{ECS-STABILITY}(D) + \textit{nc-D}(D)$ contains the graph sequences $\texttt{ECS-STICKY}(D) \cap \Diamond\texttt{ECS-STABILITY}(D) \cap \textit{nc-D}(D)$.*

2.4.3 Rooted-Eventually-Stable message adversary

The RES(rooted-eventually-stable)-adversary [WSS19] is a message adversary that, like the ECS-adversary, models a chaotic phase that precedes a brief period of stabilization. Instead of allowing nearly arbitrary chaos at the beginning and restricted chaos after the stability period, however, the RES-adversary simply demands that all communication graphs contain a single root in each round.

The liveness property of the RES-adversary is that it must allow for an R-single-rooted subsequence of sufficient length.

Definition 16 ($\Diamond\texttt{GOOD}(x)$ [WSS19, Definition 5])**.** $\Diamond\texttt{GOOD}(x)$ *is the set of all infinite communication graph sequences* σ *such that there exists a set* $R \subseteq \Pi$ *and an* R*-single-rooted* $\sigma' \subseteq \sigma$ *with* $|\sigma'| \geqslant x$*.*

The safety property is, as stated earlier, simply the fact that all graphs in our graph sequences must be single-rooted. This prevents some tactics where the message adversary produces disjoint subgraphs, as shown in σ_2 in Figure 2.5.

Definition 17 (ROOTED [WSS19, Definition 6])**.** ROOTED *is the set of all infinite sequences* σ *of* rooted *communication graphs.*

Together with the guarantee of a consecutive dynamic diameter of D, these two properties form the message adversary $\Diamond\texttt{STABLE}_D(x)$.

Definition 18 ($\Diamond\texttt{STABLE}_D(x)$ [WSS19, Definition 7])**.** *The message adversary* $\Diamond\texttt{STABLE}_D(x) = \texttt{ROOTED} + \Diamond\texttt{GOOD}(x) + c\text{-}D(D)$ *contains the graph sequences* $\texttt{ROOTED} \cap \Diamond\texttt{GOOD}(x) \cap c\text{-}D(D)$*.*

2.4.4 Transient Stability message adversary

The Transient Stability message adversary is a generalized version of the ECS-message adversary [SWS16, Chapter 6]. Note carefully that, while in the original paper x and y were used for $\Diamond\texttt{ECS-STABILITY}(y)$ and $\texttt{ECS-STICKY}(x)$, they are swapped here for consistency!
Additionally in this more generalized form we now use the consecutive dynamic diameter $c\text{-}D(D)$, in Corollary 29 we proof that this is a weaker bound on the message adversary.

Definition 19 (Transient stability message adversary MA(x, y, D) [SWS16, Chapter 6])**.** *The transient stability message adversary* $MA(x, y, D) = \Diamond\texttt{ECS-STABILITY}(x) + \texttt{ECS-STICKY}(y) + c\text{-}D(D)$ *contains all graph sequences in* $\Diamond\texttt{ECS-STABILITY}(x) \cap \texttt{ECS-STICKY}(y)$ *that guarantees a consecutive dynamic diameter of* D*.*

2.4.5 Comparison and Overview

As discussed earlier, the set of all message adversaries $\{\texttt{MA}, \subseteq\}$ is a (reflexive) partial order. Up to this point, we introduced six properties and three message adversaries derived from them. Figure 2.6 is a Hasse diagram detailing how the properties relate to one another, Figure 2.7 focuses on the message adversaries and their relationship. The dotted lines in Figure 2.6 are used to display the relationship between increasing parameter values of properties. An example of this is the relationship between the consecutive and non-consecutive dynamic diameters on the left hand side. Here one can infer:

- $\forall D : D > 2 : c\text{-}D(D-1) \subset c\text{-}D(D)$
- $\forall D : D > 2 : nc\text{-}D(D-1) \subset nc\text{-}D(D)$
- $\forall D : D > 2 : nc\text{-}D(D) \subset c\text{-}D(D)$

We introduce the omniscient, omnipotent message adversary $\mathbf{\Omega}$. This adversary has no limitation on what it can and cannot do and is used as a maximal element in our partial order $\{\texttt{MA}, \subseteq\}$. We also introduce the weakest possible message adversary $\mathbf{\varnothing}$ as the minimal element, which is unable to drop any messages - resulting in a fully connected communication graph in every round.

We will now provide the proofs of the relations depicted in Figure 2.6 and Figure 2.7.

Property relation proofs

Lemma 20. $nc\text{-}D(1) = c\text{-}D(1)$.

Proof. This follows from their definitions - a graph sequence consisting of a single round is both consecutive and non-consecutive. □

Lemma 21. $c\text{-}D(1) \subseteq \texttt{ROOTED}$.

Proof. By contradiction. Assume that there is a graph, which has two root components, but guarantees a diameter of 1. This would imply that every process p_i in both root components has an incoming edge from each process p_j in the other root component. This is a contradiction, since a root is defined as a set of processes without external incoming edges. □

Lemma 22. $\texttt{ROOTED} \not\subseteq c\text{-}D(1)$.

Proof. By example. All graphs in Figure 2.1 are rooted but do not guarantee a consecutive dynamic diameter of 1. □

The above lemmas immediately imply the following corollary.

Corollary 23. $nc\text{-}D(1) = c\text{-}D(1) \subset \texttt{ROOTED}$.

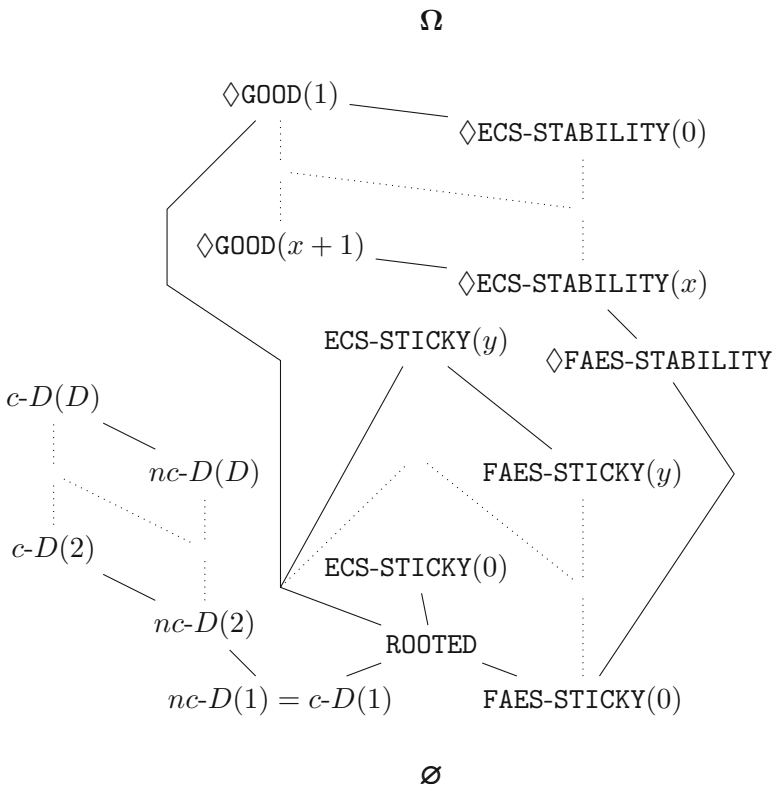


Figure 2.6: The partially ordered set of properties. Weaker properties (stronger message adversaries) are higher-up in this Hasse-diagram.
The blank spaces reached by dotted lines depict the intermediary parameterizations of properties. E.g. the spot between ECS-STICKY(0) and ECS-STICKY(y), symbolises all parameterizations ECS-STICKY($z : 0 < z < y$)

Lemma 24. $\forall D \geqslant 1 : nc\text{-}D(D) \subseteq nc\text{-}D(D+1) \wedge c\text{-}D(D) \subseteq c\text{-}D(D+1)$.

Proof. Ths follows from the definition of the causal past, since $\forall i \in \mathrm{CP}_p^{r_d}(a) \Rightarrow i \in \mathrm{CP}_p^{r_{d+1}}(a)$ for $1 \leqslant d \leqslant D$, recall Definition 5. □

Lemma 25. $\forall D \geqslant 1 : nc\text{-}D(D+1) \nsubseteq nc\text{-}D(D) \wedge c\text{-}D(D+1) \nsubseteq c\text{-}D(D)$.

Proof. By example. In Figure 2.8, a graph sequence is shown that is admissible under the $nc\text{-}D(D+1)$ and $c\text{-}D(D+1)$ adversaries but not under the $nc\text{-}D(D)$ and $c\text{-}D(D)$ adversaries. □

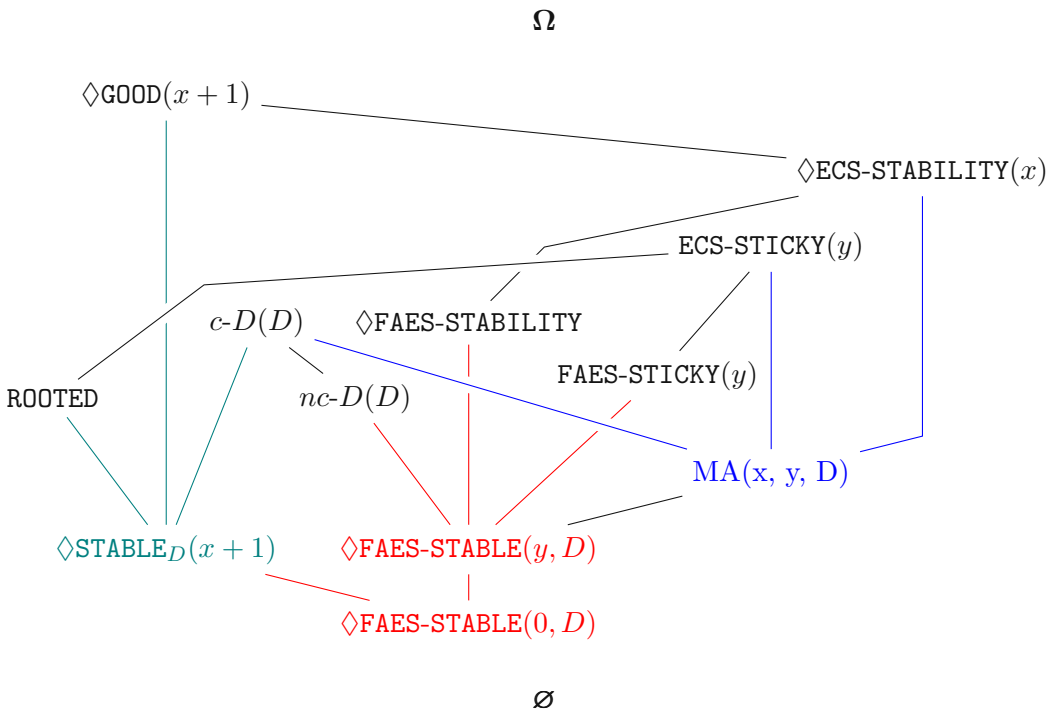


Figure 2.7: The partially ordered set of message adversaries.

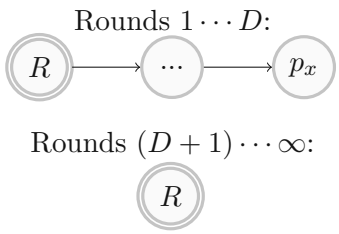


Figure 2.8: Example of a graph sequence that is admissible under the $nc\text{-}D(D+1)$ and $c\text{-}D(D+1)$ adversaries but not under the $nc\text{-}D(D)$ and $c\text{-}D(D)$ adversaries. Note that the ... denotes a chain of D processes.

Corollary 26. $\forall D \geqslant 1 : nc\text{-}D(D) \subset nc\text{-}D(D+1) \wedge c\text{-}D(D) \subset c\text{-}D(D+1)$.

Lemma 27. $\forall x > 1 : nc\text{-}D(D) \subseteq c\text{-}D(D)$.

Proof. This follows directly from the definition. □

Lemma 28. $\forall x > 1 : c\text{-}D(D) \nsubseteq nc\text{-}D(D)$.

Proof. By example. In Figure 2.9, a graph sequence is shown that is admissible under the $c\text{-}D(D)$-adversary but not under the $nc\text{-}D(D)$-adversary. □

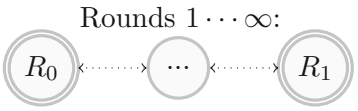


Figure 2.9: Example of a graph sequence that is admissible under the c-$D(D)$-adversary, but would not be admissible under the nc-$D(D)$-adversary. Note that the ... denotes a chain of D processes and the dotted lines denote communication paths that change direction each round. For example, in the first round R_0 is the only root of the graph and all communication goes from left to right and vice versa in the second round.

Corollary 29. $\forall x > 1 : nc\text{-}D(x) \subset c\text{-}D(x)$.

Lemma 30. $\forall y \geqslant 0 : \texttt{FAES-STICKY}(y) \subseteq \texttt{FAES-STICKY}(y+1)$.

Proof. Ths follows from the fact that if a root R is common for y rounds, it is the FAES-common root and therefore will also be common for $y+1$ rounds. □

Lemma 31. $\forall y \geqslant 0 : \texttt{FAES-STICKY}(y+1) \nsubseteq \texttt{FAES-STICKY}(y)$.

Proof. By example. In Figure 2.10, a graph sequence is shown that is admissible under the $\texttt{FAES-STICKY}(y+1)$-adversary but not under the $\texttt{FAES-STICKY}(y)$-adversary. □

Figure 2.10: Example of a graph sequence that is admissible under the $\texttt{FAES-STICKY}(y+1)$-adversary but not under the $\texttt{FAES-STICKY}(y)$-adversary.

Corollary 32. $\forall y \geqslant 0 : \texttt{FAES-STICKY}(y) \subset \texttt{FAES-STICKY}(y+1)$.

Lemma 33. $\texttt{FAES-STICKY}(0) \subseteq \texttt{ROOTED}$.

Proof. By contradiction. Assume that there is a graph sequence in $\texttt{FAES-STICKY}(0)$ that has two root components in some round r. This would imply that both root components will eventually become single and never disappear - which is a contradiction. □

Lemma 34. $\texttt{FAES-STICKY}(0) \not\supseteq \texttt{ROOTED}$.

Proof. By example. σ_2 in Figure 2.4 is in ROOTED but not in $\texttt{FAES-STICKY}(0)$. □

Corollary 35. $\texttt{FAES-STICKY}(0) \subset \texttt{ROOTED}$.

Theorem 36. $\forall y, y' : y \neq y' \wedge y \geqslant 0 \wedge y' \geqslant 0 \Rightarrow \texttt{ECS-STICKY}(y) \nsubseteq \texttt{ECS-STICKY}(y') \wedge \texttt{ECS-STICKY}(y') \nsubseteq \texttt{ECS-STICKY}(y)$.

Proof. Figure 2.11 shows the construction of a graph sequence that is not admissible under the ECS-STICKY(y)-adversary but admissible under the ECS-STICKY(y')-adversary for $y \neq y'$. □

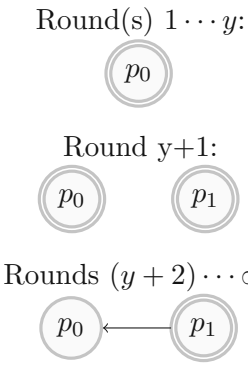


Figure 2.11: Example of a graph sequence that is not admissible under the ECS-STICKY(y)-adversary but would be admissible under any other ECS-STICKY($y' : y' \neq y$)-adversary. For the special case of $y = 0$, rounds $1 \cdots 0$ must be dropped.

Lemma 37. $\forall y$: ROOTED $\subseteq$ ECS-STICKY(y)

Proof. By definition. As in a rooted graph sequence all common roots are necessarily common-single roots. □

Lemma 38. $\forall y$: ECS-STICKY(y) $\not\subseteq$ ROOTED.

Proof. By example. The graph sequence in Figure 2.12 is admissible under any ECS-STICKY(y)-adversary but not under ROOTED. □

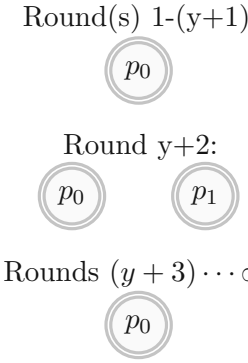


Figure 2.12: Example of a graph sequence that is admissible under any ECS-STICKY(y)-adversary but is not admissible under the ROOTED-adversary.

Corollary 39. $\forall y : \texttt{ROOTED} \subset \texttt{ECS-STICKY}(y)$

Theorem 40. $\forall y \geqslant 0 : \texttt{FAES-STICKY}(y) \subset \texttt{ECS-STICKY}(y)$

Proof. This follows from the fact that both $\texttt{ECS-STICKY}(y)$ and $\texttt{FAES-STICKY}(y)$ react to the first root component that reoccurs in more than y consecutive rounds and that this root component being a FAES-root component necessarily implies that it is also an ECS$(y+1)$-common root component. □

Theorem 41. $\forall x \geqslant 0 : \Diamond\texttt{ECS-STABILITY}(x+1) \subset \Diamond\texttt{ECS-STABILITY}(x)$

Proof. This follows directly from the fact that any adversary that guarantees $\Diamond\texttt{ECS-STABILITY}(x+1)$ by definition also guarantees $\Diamond\texttt{ECS-STABILITY}(x)$ but not vice versa. □

Theorem 42. $\forall x \geqslant 0 : \Diamond\texttt{FAES-STABILITY} \subset \Diamond\texttt{ECS-STABILITY}(x)$.

Proof. This follows from the fact that any FAES-root is forever single by definition. □

Theorem 43. $\forall x \geqslant 1 : \Diamond\texttt{GOOD}(x+1) \subset \Diamond\texttt{GOOD}(x)$

Proof. This follows directly from the fact that any adversary that guarantees $\Diamond\texttt{GOOD}(x+1)$ by definition also guarantees $\Diamond\texttt{GOOD}(x)$ but not vice versa. □

Lemma 44. $\forall x \geqslant 0 : \Diamond\texttt{ECS-STABILITY}(x) \subseteq \Diamond\texttt{GOOD}(x+1)$.

Proof. This follows from the fact that any ECS$(x+1)$-root, is by definition single for at least $x+1$ rounds. □

Lemma 45. $\forall x \geqslant 0 : \Diamond\texttt{GOOD}(x+1) \not\subseteq \Diamond\texttt{ECS-STABILITY}(x)$.

Proof. By example. In Figure 2.13, a graph sequence is shown that is admissible under the $\Diamond\texttt{GOOD}(x+1)$-adversary but not under the $\Diamond\texttt{ECS-STABILITY}(x)$-adversary, since the reappearance requirement of $\Diamond\texttt{ECS-STABILITY}(x)$ is not guaranteed for any values of D. □

Figure 2.13: Example of a graph sequence that is admissible under the $\Diamond\texttt{GOOD}(x+1)$-adversary but not under the $\Diamond\texttt{ECS-STABILITY}(x)$-adversary.

Corollary 46. $\forall x \geqslant 0 : \Diamond\texttt{ECS-STABILITY}(x) \subset \Diamond\texttt{GOOD}(x+1)$

And finally, we show the relation between some liveness and safety properties for small values:

Theorem 47. $\texttt{ROOTED} \subset \Diamond\texttt{GOOD}(1)$.

Proof. Since in a rooted execution all roots are necessarily single, rooted automatically guarantees $\Diamond\texttt{GOOD}(1)$. □

Theorem 48. $\texttt{FAES-STICKY}(0) \subset \Diamond\texttt{FAES-STABILITY}$.

Proof. Since $\texttt{FAES-STICKY}(0)$ guarantees that the first root will be the FAES-common root, it automatically guarantees the existence of a FAES-common root. □

Message adversary relation proofs

The results in this subsection justify the message adversary relations depicted in Figure 2.7.

Lemma 49. $\forall D \geqslant 1, x \geqslant 0, y \geqslant 0 : \Diamond\texttt{FAES-STABLE}(y, D) \subseteq MA(x, y, D)$.

Proof. This follows directly from Theorem 40 and Theorem 42. □

Lemma 50. $\forall D \geqslant 1, x \geqslant 0, y \geqslant 0 : MA(x, y, D) \not\subseteq \Diamond\texttt{FAES-STABLE}(y, D)$.

Proof. By example. The graph sequence in Figure 2.14 is admissible under $MA(x, y, D)$, but not under $\Diamond\texttt{FAES-STABLE}(y, D)$. □

Figure 2.14: Example of a graph sequence that is admissible both under the $\Diamond\texttt{STABLE}_D(x)$-adversary and the $MA(x, y, D)$-adversary but not under the $\Diamond\texttt{FAES-STABLE}(y, D)$-adversary.

Corollary 51. $\forall D \geqslant 1, x \geqslant 0, y \geqslant 0 : \Diamond\texttt{FAES-STABLE}(y, D) \subset MA(x, y, D)$.

Theorem 52. $\forall D \geqslant 1, x \geqslant 0, y > 0 : \Diamond\texttt{FAES-STABLE}(y, D) \not\subseteq \Diamond\texttt{STABLE}_D(x)$.

Proof. By example. The graph sequence in Figure 2.12 is admissible under $\Diamond\texttt{FAES-STABLE}(y, D)$ for values of $y > 0$ but not under $\Diamond\texttt{STABLE}_D(x)$. □

Lemma 53. $\forall D \geqslant 1, x \geqslant 0 : \Diamond\texttt{FAES-STABLE}(0, D) \subseteq \Diamond\texttt{STABLE}_D(x)$.

Proof. This follows directly from Lemma 27, Corollary 35, Theorem 42 and Corollary 46. □

Lemma 54. $\forall D \geqslant 1, x \geqslant 0, y \geqslant 0 : \Diamond\texttt{STABLE}_D(x) \nsubseteq \Diamond\texttt{FAES-STABLE}(y, D)$.

Proof. By example. The graph sequence in Figure 2.14 is admissible under $\Diamond\texttt{STABLE}_D(x)$, but not under $\Diamond\texttt{FAES-STABLE}(y, D)$. □

Corollary 55. $\forall D \geqslant 1, x \geqslant 0 : \Diamond\texttt{FAES-STABLE}(0, D) \subset \Diamond\texttt{STABLE}_D(x)$.

Consensus solvability for message adversaries

For the message adversary $\Diamond\texttt{STABLE}_D(x : x < 2D)$, [WSS19, Theorem 2] shows that consensus is unsolvable, while [SWS16, Theorem 3] shows that consensus is solvable for the message adversary $\Diamond\texttt{STABLE}_D(x : x > 2D)$. The open question of $\Diamond\texttt{STABLE}_D(2D)$ will be adressed later in Section 3.4. The status of consensus solvability for $\Diamond\texttt{STABLE}_D$, both before and after this thesis, is illustrated in Figure 4.2.

For the message adversary $\Diamond\texttt{FAES-STABLE}(D, D)$, [SWS16, Sec. 5] provides an algorithm that solves consensus. Although not explicitly stated in the paper, the algorithm also solves consensus for the generalization $\Diamond\texttt{FAES-STABLE}(y : y \neq D, D)$:

Theorem 56. *The algorithm depicted in [SWS16, Fig. 3] solves consensus for* $\Diamond\texttt{FAES-STABLE}(y, D) : y < D$.

Proof. This was indirectly shown by Corollary 32, since an algorithm that solves consensus under some message adversary necessarily solves it under a weaker message adversary as well. □

Theorem 57. *A modified version of the algorithm depicted in [SWS16, Fig. 3] solves consensus for* $\Diamond\texttt{FAES-STABLE}(y, D) : y > D$.

Proof. This was indirectly shown by Corollary 26, since an algorithm that solves consensus under some message adversary necessarily solves it under a weaker message adversary as well.

For $y > D$, the algorithm [SWS16, Fig. 3] requires the small update of using $max(y, D)$ instead of D to determine the waiting periods. This is possible, since no claim is made on it being an optimal algorithm with respect to termination time. Additionally, since all proofs used to show correctness of the algorithm [SWS16, Lemmas 2-7] only use the notion of a consecutive single common root, one could even replace the non-consecutive dynamic diameter with the weaker notion of a consecutive one in the definition of the

FAES-message adversary.
(This would render the resulting slightly weaker message adversary fully characterized, but also incomparable with respect to the $\Diamond$ECS-STABLE(D) message adversary). □

While the $\Diamond$ECS-STABLE(D) message adversary is already fully characterized [SWS16, Theorem 3], the generalized message adversary MA(x, y, D) is not. The open range of parameters are adressed in Section 3.1 3.2 and 3.3. The status of consensus solvability for MA(x, y, D), both before and after this thesis, is illustrated in Figure 4.1.

CHAPTER 3

Characterization of MA(x, y, D)

In this chapter, we characterize the transient stability message adversary MA(x, y, D) using the following three proofs:

- Consensus impossibility for MA(x, y, D): $y > x$ (Section 3.1)
- Consensus impossibility for MA(x, y, D): $y < D$ (Section 3.2)
- Consensus solvability for MA(x, y, D): $x \geqslant y \geqslant D$ (Section 3.3)

Additionally, a proof for the impossibility of consensus for $\Diamond\texttt{STABLE}_D(2D)$ is given in Section 3.4, thereby also completing the characterization of that setting.

Theorem 58 (Solvability MA(x, y, D) [SWS16, Theorem 3])**.** *Solving Consensus is impossible under message adversary MA(x, y, D):* $y > x$.

Theorem 59 (Impossibility MA(x, y, D) [SWS16, Theorem 4])**.** *Solving Consensus is possible under message adversary MA(x, y, D):* $x = y = D$.

Figure 3.1 summarizes what was stated about MA(x, y, D) in [SWS16]. In it, the areas coloured red represent the ability of the message adversary to prevent consensus and the thin green line represents the message adversary $\Diamond\texttt{ECS-STABLE}(D)$ for which consensus can be solved. The currently unknown combinations of parameters (yellow) will be explored as part of this thesis.

The proofs presented in this section are all indistinguishability proofs as discussed in Section 2.3.

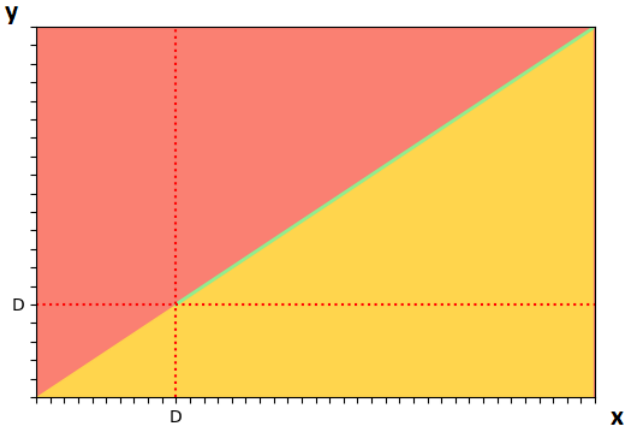


Figure 3.1: Characterization of consensus solvability under MA(x, y, D) according to [SWS16].

3.1 Impossibility under MA(x, y, D): $y > x$

The general idea of this proof (for Theorem 58) is that once $y > x$, the message adversary can choose executions that never contain any graph sequence with a maximal common root R in more than x consecutive rounds. This prevents the processes from safely deciding on a suspected single-root, as the message adversary could hide a disconnected subgraph, in which all processes started with a different input value. An example of this is shown in the execution α_2 in Figure 3.3.

Proof. By contradiction. We assume the existence of some algorithm that solves consensus under MA(x, y, D): $y > x$.
Three executions α_1, α_2 and α_3 will be provided, which fulfil the criteria for admissibility under the message adversary MA(x, y, D) and whose graph sequences are indistinguishable from one another for at least one process ($\sigma_1 \overset{r}{\sim} \sigma_2 \overset{p}{\sim} \sigma_3$). In α_1 resp. α_3, all processes start with input value 1 resp. 0.
By **agreement**, all processes need to decide on the same values in each execution. Additionally, by **validity**, all processes in α_1 need to decide on 1 and all processes in α_3 need to decide on 0. But since the three graph sequences σ_1, σ_2 and σ_3 are indistinguishable for at least one process, all processes have to decide on the same values in all executions, which provides the required contradiction.
We will now provide the executions α_1, α_2 and α_3, using our graphical notation, which we recall briefly.

Notation:

- Shown are executions (represented by their graph sequence and their input values) over time and discussed below the admissibility of their communication graphs.
- Each vertex (representing a single process) is coloured corresponding to its input value (green 1, red 0).
- Root components consist of a single process and are marked by having a double border.
- A dotted line between two vertices describes a directed connection whose direction flips every round (see e.g. rounds (x+1)-∞ in Figure 3.2).

Figure 3.2: Execution α_1 for the proof of Theorem 1.

Graph sequence σ_1 (of α_1 in Figure 3.2):

- c-D(D): As all processes are either directly connected to a root (or are one), σ_1 even guarantees a consecutive dynamic diameter $D = 1$.
- `ECS-STICKY`(y): Since no root is common for more than $x + 1$ consecutive rounds (and by assumption $y > x$), σ_1 guarantees this property.
- $\Diamond$`ECS-STABILITY`(x): Since the node r is a single root for $x + 1$ consecutive rounds (in round $x + 1$ r is still the root and only after this it alternates with r') and will infinitely often reappear as a single root, σ_1 guarantees this property.

Therefore, σ_1 is admissible under MA(x, y, D) and so all processes in α_1 will (by **termination**) decide on some value by some round T.

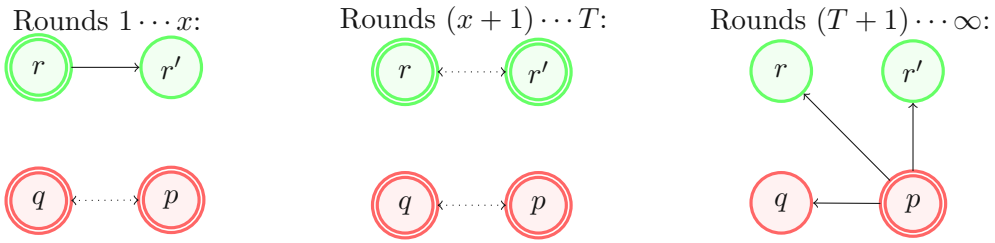


Figure 3.3: Execution α_2 for the proof of Theorem 1.

Graph sequence σ_2 (of α_2 in Figure 3.3):

- c-D(D): As only p is ever a single root and in all rounds it appears as such, all other processes are either directly connected it, σ_2 even guarantees a consecutive dynamic diameter $D = 1$.
- ECS-STICKY(y): Since the earliest root that is common for more than y consecutive rounds is p, which is even a FAES-common-root, σ_2 guarantees this property.
- $\Diamond$ECS-STABILITY(x): Since the node p is a single root from round $T + 1$ onwards, σ_2 guarantees this property.

Therefore, σ_2 is admissible under MA(x, y, D) and so all processes in α_2 will (by **termination**) decide on some value by some round T'.

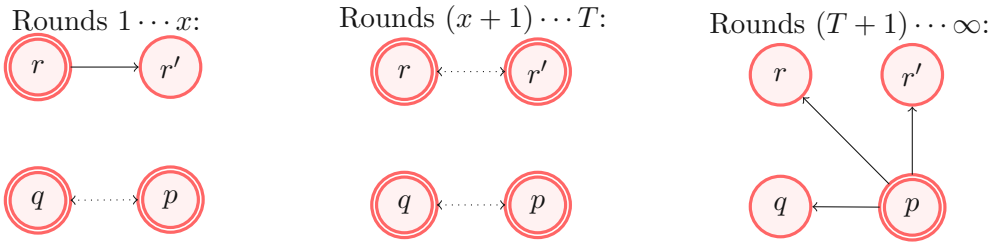


Figure 3.4: Execution α_3 for the proof of Theorem 1.

Graph sequence σ_3 (of α_3 in Figure 3.3):
Since σ_3 is identical to σ_2, with the only difference being their initial configurations, the properties c-D(D), ECS-STICKY(y) and $\Diamond$ECS-STABILITY(x) are guaranteed to hold for σ_3, hence σ_3 is admissible under MA(x, y, D) and so all processes in α_3 will (by **termination**) decide on some value by some round T''. (Note that we may not assume that T'=T'').

As for $\sigma_1 \overset{r}{\sim} \sigma_2$, we note that r has the same input value in both executions and obviously receives the same messages in round $1, ..., T$ in Figure 3.2 and Figure 3.3.
As for $\sigma_2 \overset{p}{\sim} \sigma_3$, we note that p has the same input value in both executions and obviously receives the same messages in all round $1, ..., \infty$ in Figure 3.3 and Figure 3.4.

With this all necessary conditions from Section 2.3 are met, concluding this proof. □

3.2 Impossibility under MA(x, y, D): $y < D$

The following proof of Theorem 61 shows that, if $y < D$, two seperate clusters of processes can exist at the same time, each one thinking that it is alone and thereby choosing to decide on inconsistent values.

Theorem 60. *Solving Consensus is trivial under message adversary MA(x, y, D): $y < D$, for $D = 1$ and $x \geqslant y$.*

Proof. This case is trivial, since $D = 1$, $y = 0$ and $x \geqslant 0$ necessitate the following: as $y = 0$, we know that the first round must necessarily be R-single-rooted. As $D = 1$ we also know that all processes must be directly connected to R. So, at the end of round 1 all processes have learned of the input value of all processes in the root component R. Once the root component reappears (necessitated by Definition 13) all processes will learn which processes belonged to R. After that any deterministic function, which does not violate **validity**, over the input values of the root component solves consensus. □

Theorem 61. *Solving Consensus is impossible under message adversary MA(x, y, D): $y < D$, for $D \geqslant 2$.*

Proof. By contradiction. We assume the existence of some algorithm that solves consensus under MA(x, y, D): $y < D$.
Two executions α_1, α_2 will be provided, which fulfil the criteria for admissibility under the message adversary MA(x, y, D) and whose graph sequences are indistinguishable from one another for at least one process ($\sigma_1 \overset{r}{\sim} \sigma_2$).
We show that all processes in α_1 will decide on 1 and all processes in α_2 will decide on 0. But as the graph sequences σ_1 and σ_2 are indistinguishable for at least one process, and all processes in its causal past have the same input values, all processes have to decide on the same value in both executions, which provides the required contradiction.

Notes:

- For our proofs, it is enough that the indistinguishability holds until the process r decides on a value, as this is enough to prevent consensus.
- As we want to prove that consensus is impossible to achieve for any value of $y < D$, and Theorem 36 shows that `ECS-STICKY`(y_1) and `ECS-STICKY`(y_2) are not comparable for different values of y_1, and y_2, we use graph sequences that guarantee any given `ECS-STICKY`($y : y < D$).

We will now provide the executions α_1 and α_2, using our graphical notation:

- Shown are executions (represented by their graph sequence and their input values) over time and discussed below the admissibility of their communication graphs.
- Each vertex (representing a single process) is coloured corresponding to its input value (green 1, red 0).
- The ... vertex is an exception to this, and denotes $D - 1$ processes, that were not used in previous round graphs, linked in a chain.
- Root components consist of a single process and are marked by having a double border.
- All processes not displayed are assumed to be directly connected to all roots and to have no outgoing edges.

- T is a parameter used for the construction of the communication graphs. It is either the round by which the process r will decide on a value in σ_1, or $D+2$ whichever is larger (The value of $D+2$ is arbitrarily set as our graph constructions assume $T > D+1$).

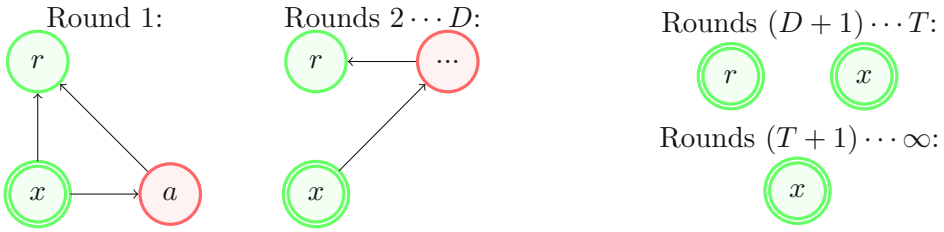


Figure 3.5: Execution α_1 for the proof of Theorem 2.

Graph sequence σ_1 (of α_1 in Figure 3.5):

- c-D(D): As process x is a single root in the first D rounds, and all other processes are directly connected to it in the first round c-D(D) holds for the first D rounds. For the rounds $k \in \{D+1, ..., T\}$, no single root exists. And for all rounds afterwards all processes are directly connected to x. So c-D(D) holds for all rounds.
- `ECS-STICKY`($y : y < D$): Since the graph sequence is x-single-rooted for the first D rounds, σ_1 guarantees these properties.
- $\Diamond$`ECS-STABILITY`(x): Since the node x is a single root from round $T+1$ onwards, σ_1 guarantees this property.

Therefore, σ_1 is admissible under MA(x, y, D).
As the only state process x ever receives is his own, it has to decide on the value 1. By **agreement** this necessitates all processes to decide on 1 in α_1.

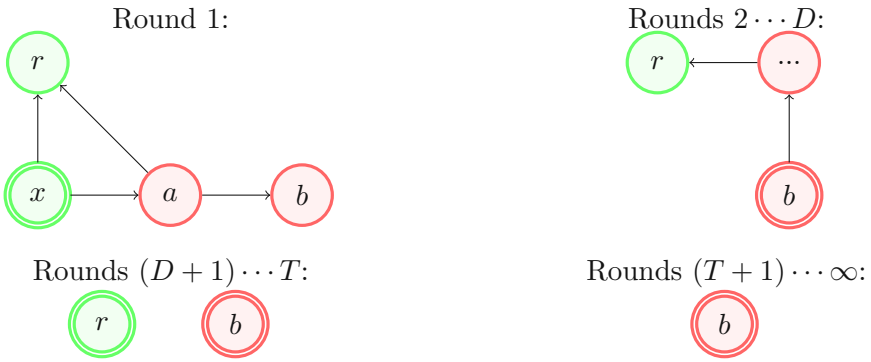


Figure 3.6: Execution α_2 for the proof of Theorem 2.

Graph sequence σ_2 (of α_2 in Figure 3.6):

- c-D(D): Since b is the only single root that exists for more than one round, and $D \geqslant 2$, we can ignore round 1. From round $D+1$ on, all processes are directly connected to b so c-D(D) holds trivially. In rounds $2, ..., D$ every process must have learned from b by round $D+1$ at latest, so c-D(D) also holds since $D \geqslant 2$.
- ECS-STICKY(0): Since the graph sequence is x-single-rooted for the first round, σ_2 guarantees this property.
- ECS-STICKY($y : 0 < y < D$): Since from round 2 on the graph has a FAES-root in b, σ_2 guarantees these properties.
- $\Diamond$ECS-STABILITY(x): Since the graph sequence contains a FAES-root, σ_2 guarantees this property.

Therefore, σ_2 is admissible under MA(x, y, D).
As the only input values that process b ever receives in σ_2 are the input value of a and b which are both 0, it has to decide on 0. By **agreement** this necessitates all processes to decide on 0 in α_2.

Since $\sigma_1 \overset{r}{\sim} \sigma_2$, however, the decision value in α_1 and α_2 should be the same.

□

3.3 Solvability under MA(x, y, D): $x \geqslant y \geqslant D$

This proof consists of three parts. The first part, solvability under MA(x, y, D): $x = y = D$ is proven in [SWS16, Theorem 3] and here assumed as given. The second part is a proof that solvability under MA(x, y, D) implies the solvability under MA(x', y, D) $\forall x' > x$. The third and last part is a proof that solvability under MA(x, y, D) implies the solvability under MA(x, y, D') $\forall D' < D$.

Theorem 62. $\forall x' > x$: *The solvability of MA(x, y, D) implies the solvability of MA(x', y, D).*

Proof. Since message adversaries are defined by the set of graph sequences they can choose from, it is enough to show that that MA(x', y, D) $\subseteq$ MA(x, y, D) if $x' > x$. MA(x, y, D) is defined as: ECS-STICKY(y) $\cap$ $\Diamond$ECS-STABILITY(x) guaranteeing a non-consecutive dynamic diameter of D. Therefore, it is enough to show that $\Diamond$ECS-STABILITY(x') $\subseteq$ $\Diamond$ECS-STABILITY(x) if $x' > x$, which is guaranteed by Theorem 41. □

Theorem 63. $\forall D' < D$: *The solvability of MA(x, y, D) implies the solvability of MA(x, y, D').*

Proof. Since message adversaries are defined by the set of graph sequences they can choose from, it is enough to show that that MA(x, y, D') $\subseteq$ MA(x, y, D) if $D' < D$. MA(x, y, D) is defined as: ECS-STICKY(y) $\cap$ $\Diamond$ECS-STABILITY(x) guaranteeing a non-consecutive

dynamic diameter of D. Therefore, it is enough to show that $nc\text{-}D(D') \subseteq nc\text{-}D(D)$ if $D' < D$, which is guaranteed by Corollary 26. □

Corollary 64. *Consensus is solvable under the message adversary MA(x, y, D): $x \geqslant y \geqslant D$.*

With Corollary 64, the message adversary MA(x, y, D) is completely characterized. The result is depicted in Figure 4.2.

3.4 Impossibility under $\Diamond\texttt{STABLE}_D(2D)$

In this subsection, we will provide an impossibility proof for consensus under the $\Diamond\texttt{STABLE}_D(x)$ message adversary [WSS19]. This answers the up to now unsolved question if $\Diamond\texttt{STABLE}_D(2D)$ is solvable, thereby also completing the characterization of this message adversary.

Theorem 65. *Solving Consensus is trivial under the message adversary $\Diamond\texttt{STABLE}_D(2D)$ for $D = 1$.*

Proof. As the graph sequences are always rooted, and $D = 1$ forces every process to be directly connected to every process in each root component, therefore the following algorithm suffices:

- Round 1:
  If a process p_i only receives messages containing $1s$ or $0s$ and this matches its input value, it decides on it.
- Round 2:
  If a process p_i only receives messages from processes that learned of both $1s$ and $0s$, decide on 0.
  Otherwise, there exists a process p_j that only received messages containing its own input value in the first round. So p_i decides on that value. (As this necessitates that the root in round 1 only contained one input value, there can't be another node p_k with a different input value that received only that value in the first round).

□

Theorem 66. *Solving Consensus is impossible under the message adversary $\Diamond\texttt{STABLE}_D(2D)$ for $D \geqslant 2$.*

The general idea of our proof is that we can construct two graph sequences (σ_3 and σ_4), such that some process x knows that it is possible that another process p has already decided on a value but does not know on which one. This either prevents x from deciding at all (violating **termination**), or forces it to "guess" a value (violating **agreement** in

some executions).

Proof. By contradiction. We assume the existence of some algorithm that solves consensus under $\Diamond\texttt{STABLE}_D(2D)$.
Six executions (α_1, α_2, α_3, α_4, α'_2, α'_1) will be provided, which fulfil the criteria for admissibility under the message adversary $\Diamond\texttt{STABLE}_D(2D)$, and whose graph sequences are indistinguishable from one another for at least one process ($\sigma_1 \overset{q_3}{\sim} \sigma_2 \overset{q_3}{\sim} \sigma_3 \overset{x}{\sim} \sigma_4 \overset{p_3}{\sim} \sigma'_2 \overset{p_3}{\sim} \sigma'_1$).
In σ_1 resp. σ'_1, all processes start with input value 1 resp. 0.
By **agreement**, all processes need to decide on the same values in each execution. Additionally, by **validity**, all processes in α_1 need to decide on 1, and all processes in α'_1 need to decide on 0. But since the six graph sequences σ_1, σ_2, σ_3, σ_4, σ'_2 and σ'_1 are indistinguishable for at least one process, all processes have to decide on the same values in all executions, which provides the required contradiction.

Notation:

- Shown are executions (represented by their graph sequence and their input values) over time and discussed below the admissibility of their communication graphs
- Each vertex (representing a single process) is coloured corresponding to its input value (green 1, red 0).
- The ... and $\vdots$ vertices are exceptions to this: the ... vertex, resp. the $\vdots$ vertex, denote D, resp. T, processes, that have not been used in the graphs of earlier rounds, which are linked in a chain.
- Root components are marked by having a double border.
- A dotted line between two vertices describes a directed connection whose direction flips every round (see e.g. rounds $(2D+1)\cdots T$ in Figure 3.9).
- All processes not displayed are assumed to be directly connected to all roots and to have no outgoing edges.

Note:

- The number T is defined as $max(T', T'')$, where T' (resp. T'') is the arbitrary number of rounds the process q_3 (resp. p_3) needs to decide on a value in α_1 (resp. α'_1).
- The executions α'_1 and α'_2 are mirrors of the graph sequences as α_1 and α_2, the processes are renamed, reordered and have a input value of 0. They are depicted in Figure 3.8 and Figure 3.10.

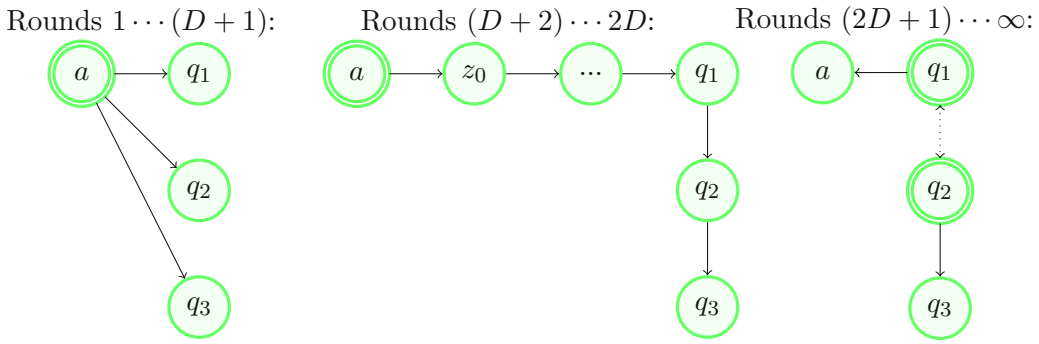


Figure 3.7: Execution α_1 for the proof of Theorem 4. For all rounds $r \geqslant 2D+1$ it holds that q_1 is the root in every odd round, and q_2 in every even round.

Graph sequence σ_1 (of α_1 in Figure 3.7):

- `ROOTED`: As every graph in the sequence is `ROOTED`, σ_1 guarantees this property.
- $\Diamond$`GOOD`$(2D)$: As a is a single root for $2D$ consecutive rounds, σ_1 guarantees this property.
- c-$D(D)$: For the first $D+1$, rounds all processes are directly connected to a, so c-D(D) holds for rounds $1, ..., (D+1)$. For rounds $k \in \{D+2, ...2D\}$ the root a is not consecutive for D rounds, so c-D(D) holds vacuously for all these rounds as well. For rounds $2D+1, ..., \infty$ no root is consecutive for $D \geqslant 2$ rounds, so c-D(D) holds vacuously as well.

Therefore, σ_1 (resp. σ_1') is admissible under $\Diamond$`STABLE`$_D(2D)$, so all processes will (by **termination**) decide on some value by some round T'(resp. T'').

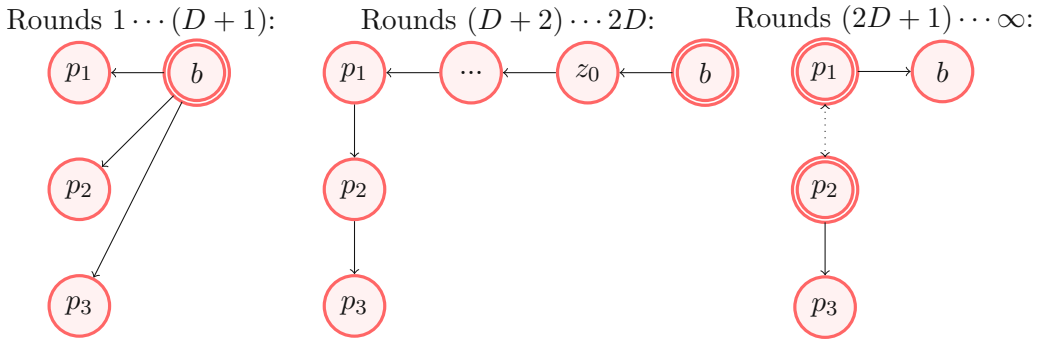


Figure 3.8: Execution α_1' for the proof of Theorem 4. For all rounds $r \geqslant 2D+1$ it holds that p_1 is the root in every odd round, and p_2 in every even round.

Graph sequence σ_1' (of α_1' in Figure 3.8): As this graph sequence is a mirror of σ_1, it has the same properties as σ_1.

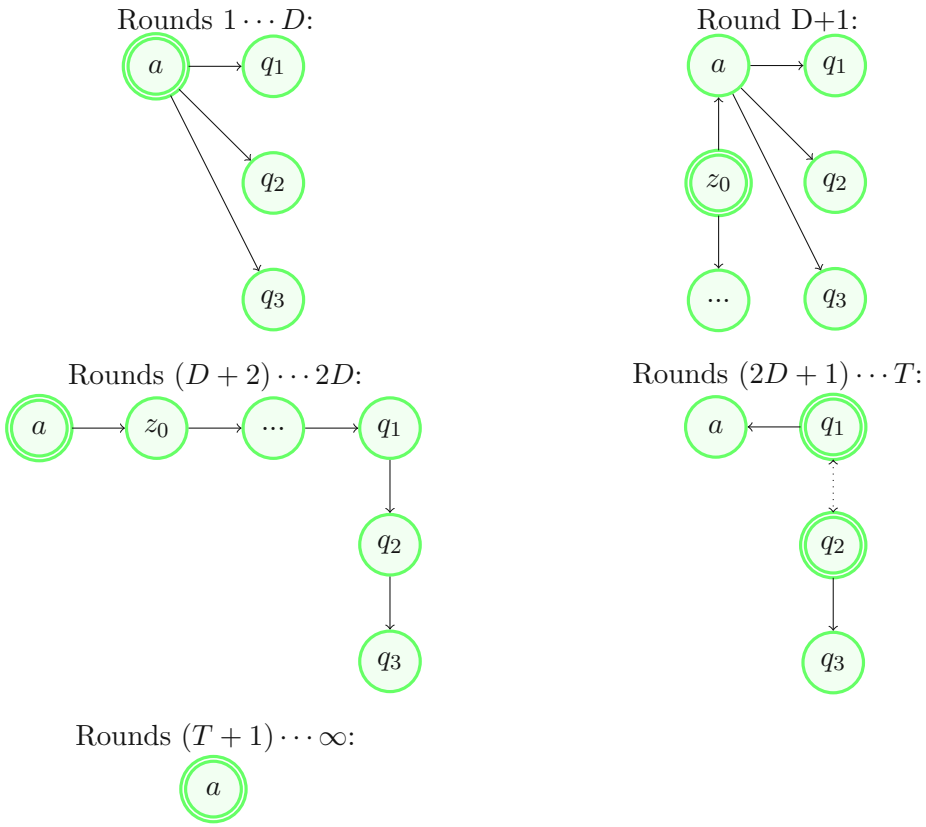


Figure 3.9: Execution α_2 for the proof of Theorem 4.

Graph sequence σ_2 (of α_2 in Figure 3.9):

- ROOTED: As every graph in the sequence is ROOTED, σ_2 guarantees this property.
- $\Diamond$GOOD$(2D)$: As a is a single root for infinitely many consecutive rounds, σ_2 guarantees this property.
- c-$D(D)$: For the first D rounds, as well as for rounds $T+1, ..., \infty$, all processes are directly connected to a, so c-D(D) holds for rounds $1, ..., D$ and rounds $T+1, ..., \infty$. For round $D+1$ as well as rounds $D+2, ..., 2D$ and rounds $2D+1, ..., T$, the root is not consecutive for at least $D \geqslant 2$ rounds, so c-D(D) holds vacuously.

Therefore, σ_2 (resp. σ_2') is admissible under $\Diamond$STABLE$_D(2D)$, so all processes will (by **termination**) decide on some value by some round.

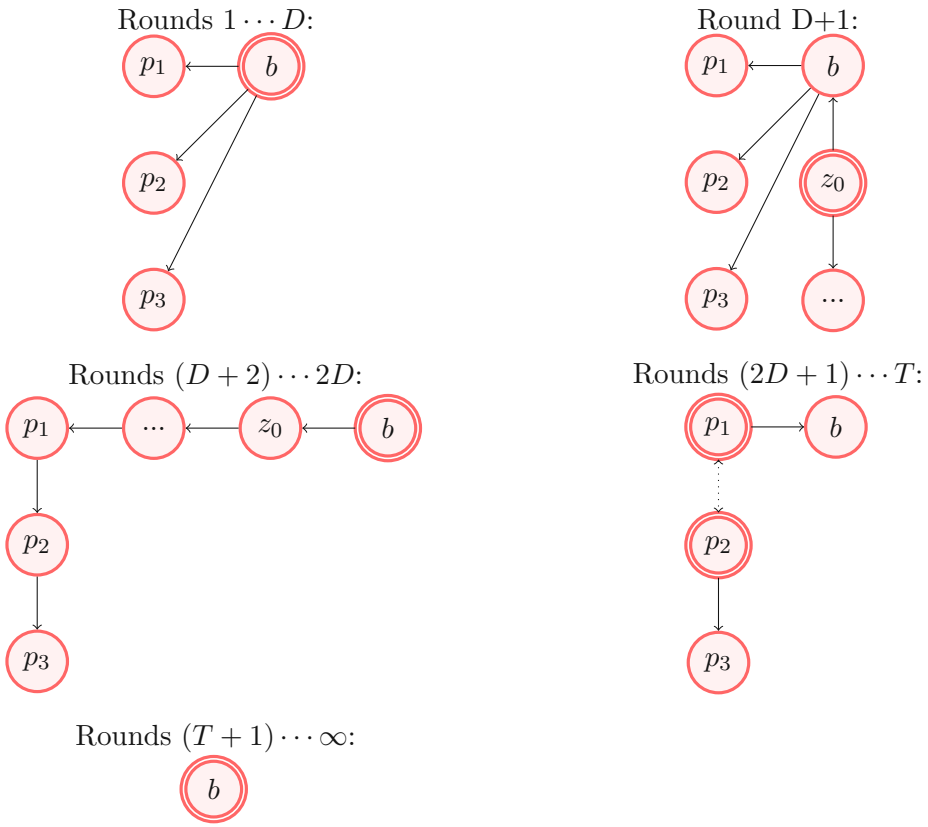


Figure 3.10: Execution α_2' for the proof of Theorem 4.

Graph sequence σ_2' (of α_2' in Figure 3.10): As this graph sequence is a mirror of σ_2, it has the same properties as σ_2.

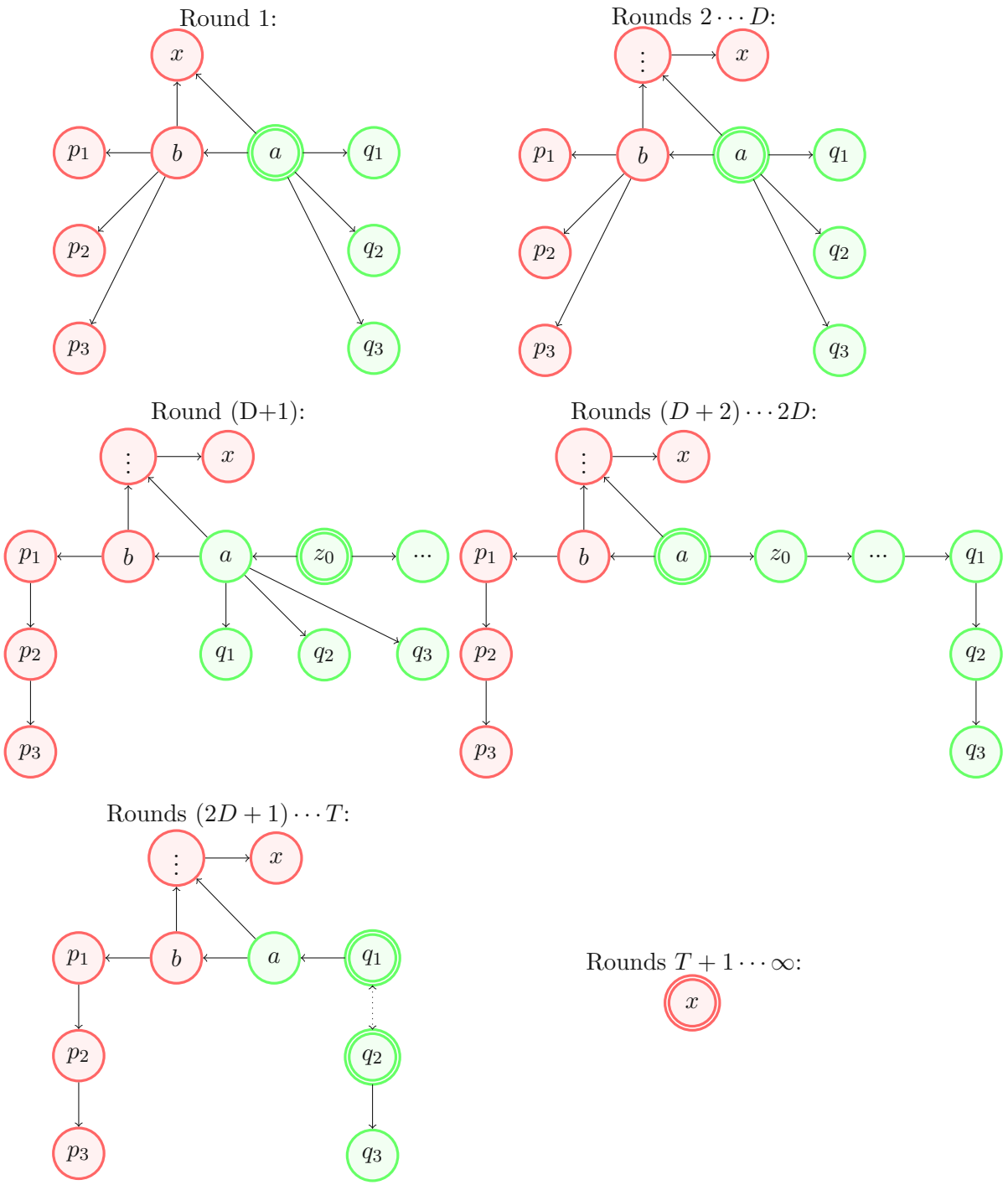
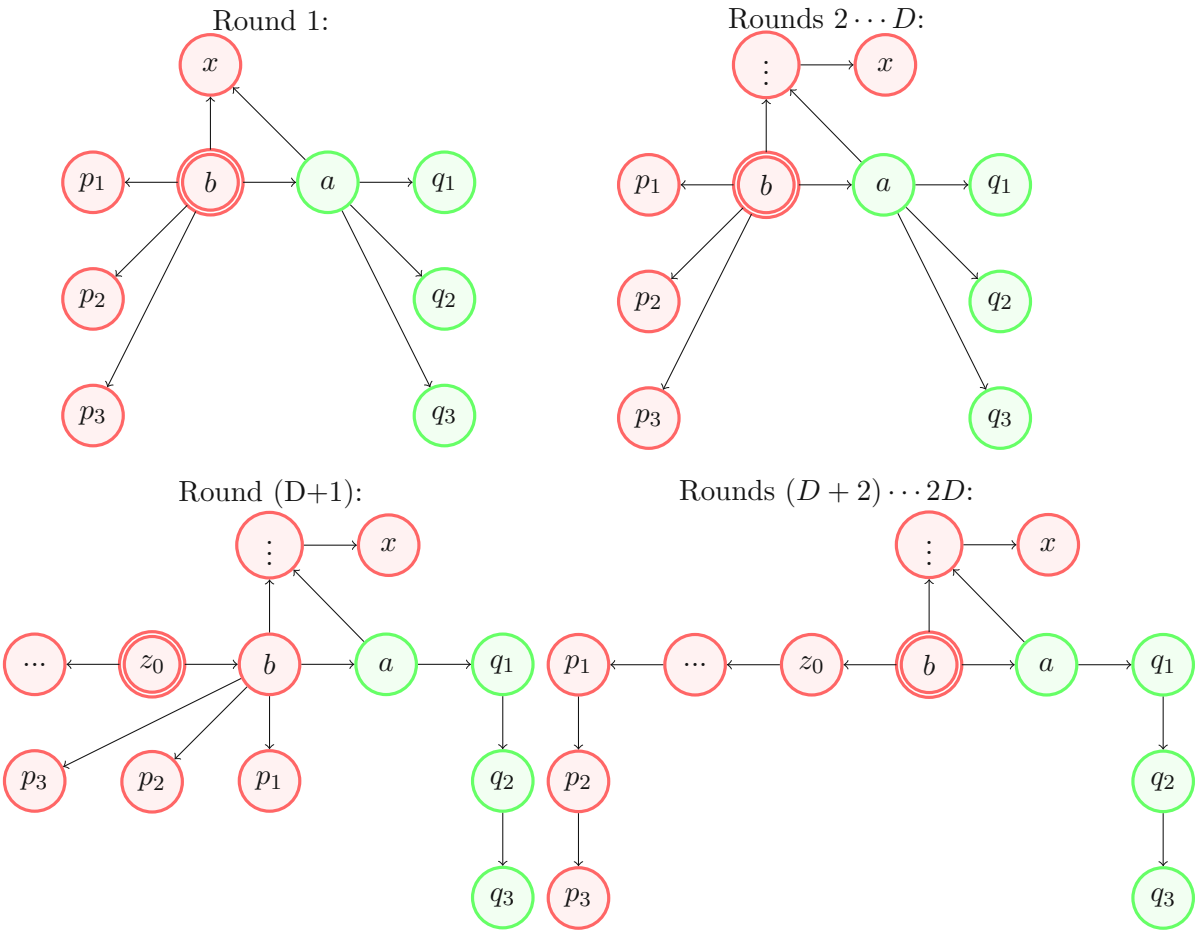


Figure 3.11: Execution α_3 for the proof of Theorem 4.

Graph sequence σ_3 (of α_3 in Figure 3.11):

- ROOTED: As every graph in the sequence is ROOTED, σ_3 guarantees this property.
- $\Diamond$GOOD$(2D)$: As x is a single root for infinitely many consecutive rounds, σ_3 guarantees this property.
- c-$D(D)$: As a is the single root for the first D rounds, and all processes have learned of its input value by round 3, c-D(D) holds for rounds $1, ..., D$. For round $D+1$ as well as rounds $D+2, ..., 2D$ and rounds $2D+1, ..., T$, the root is not consecutive for at least $D \geqslant 2$ rounds, so c-D(D) holds vacuously. For all rounds $T+1, ...\infty$ all processes are directly connected to x, so c-D(D) holds for those rounds as well.

Therefore, σ_3 is admissible under $\Diamond\texttt{STABLE}_D(2D)$, so all processes will (by **termination**) decide on some value by some round.



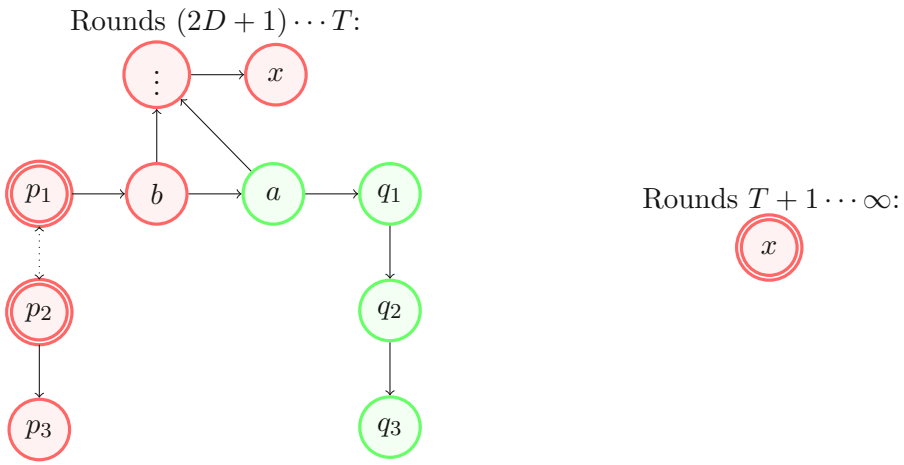


Figure 3.12: Execution α_4 for the proof of Theorem 4.

Graph sequence σ_4 (of α_4 Figure 3.12):

- ROOTED: As every graph in the sequence is ROOTED, σ_4 guarantees this property.
- ◇GOOD(2D): As x is a single root for infinitely many consecutive rounds, σ_4 guarantees this property.
- c-D(D): As b is the single root for the first D rounds, and all processes have learned of its input value by round 3, c-D(D) holds for rounds $1, ..., D$. For round $D+1$ as well as rounds $D+2, ..., 2D$ and rounds $2D+1, ..., T$, the root is not consecutive for at least $D \geqslant 2$ rounds, so c-D(D) holds vacuously. For all rounds $T+1, ...\infty$ all processes are directly connected to x, so c-D(D) holds for those rounds as well.

Therefore, σ_4 is admissible under ◇STABLE$_D$(2D), so all processes will (by **termination**) decide on some value until some arbitrary round.

Now all that is left to show is that $\sigma_1 \overset{q_3}{\sim} \sigma_2 \overset{q_3}{\sim} \sigma_3 \overset{x}{\sim} \sigma_4 \overset{p_3}{\sim} \sigma_2' \overset{p_3}{\sim} \sigma_1'$.
$\sigma_1 \overset{q_3}{\sim} \sigma_2$ is due to the construction of the graph sequence - the chain of D distinct processes prevents any information, of whether a or z_0 was the root in round $D+2$, from ever reaching the processes q_1, q_2, and q_3.
$\sigma_2 \overset{q_3}{\sim} \sigma_3$ is trivial, as the only difference in these executions until round T (when q_3 must have decided) is the depiction of additional processes that have no outgoing edges to any of the processes depicted in σ_2.
$\sigma_4 \overset{p_3}{\sim} \sigma_2' \overset{p_3}{\sim} \sigma_1'$ follows from the same arguments.
$\sigma_3 \overset{x}{\sim} \sigma_4$ due to the construction of the graph sequence: the chain of T distinct processes prevents any information would allow to distinguish be between σ_3 and σ_4 from reaching ever x.

□

CHAPTER 4

Conclusion

In this thesis, we completely characterized the transient stability message adversary MA(x, y, D), a generalized form of an eventual message adversary established in [SWS16]. The result is shown in Figure 4.1. Our characterization relies on three proofs (Theorem 58, Theorem 61 and Corollary 64). Whereas Theorem 58 was already proven in [SWS16] and Corollary 64 was, while not formally proven, mentioned there, Theorem 61 is an original contribution of this thesis. Our results reveal that the algorithm provided in [SWS16] is optimal, in the sense that it solves consensus for all non-trivial parameterizations of this message adversary.
In addition, this thesis also completes the characterization of the $\Diamond\texttt{STABLE}_D(x)$ message adversary from [WSS19], which is shown in Figure 4.2: We provided Theorem 66, which reveals that consensus is impossible if $x = 2D : D \geqslant 2$, and Theorem 65, which solves the trivial case of $x = 2D : D = 1$.

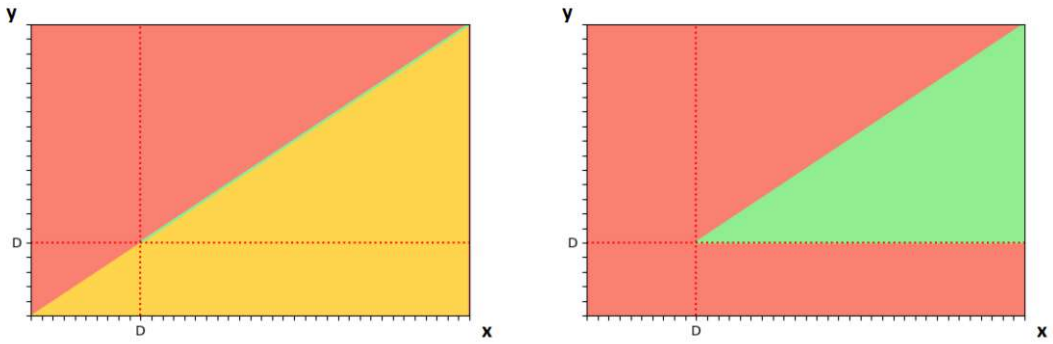


Figure 4.1: Complete Characterization MA(x, y, D).

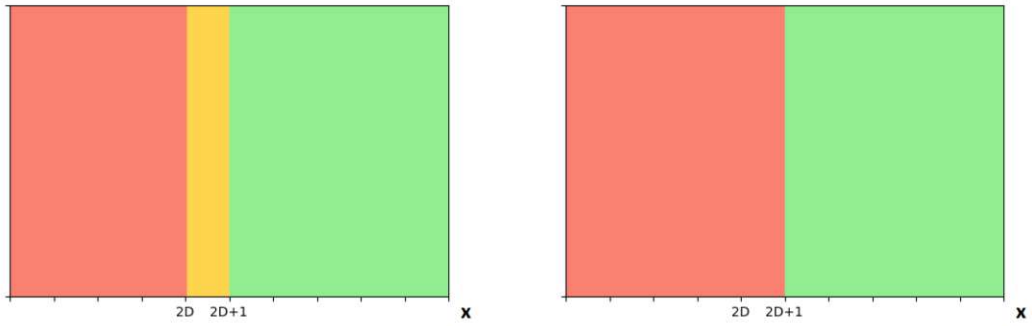


Figure 4.2: Complete Characterization $\Diamond \texttt{STABLE}_D(x)$.

Glossary

$\Diamond$FAES-STABLE(y, D)	The generalized forever-after-eventually-single message adversary. $\Diamond$FAES-STABLE(y, D) = FAES-STICKY(y)+$\Diamond$FAES-STABILITY+ nc-$D(D)$.	Def. 11	Page 15
FAES-STICKY(y)	Every root R is common for $> y$ consecutive rounds is the FAES-common root R.	Def. 10	Page 15
$\Diamond$FAES-STABILITY	A FAES-common root R exists in the graph sequence.	Def. 16	Page 15
$MA(x, y, D)$	The transient stability message adversary - a generalized form of the $\Diamond$ECS-STABLE(D) message adversary. MA(x, y, D) = $\Diamond$ECS-STABILITY(x) + ECS-STICKY(y) + c-$D(D)$.	Def. 19	Page 17
$\Diamond$ECS-STABLE(D)	The embedded-consecutive-single message adversary. $\Diamond$ECS-STABLE(D) = ECS-STICKY(D) + $\Diamond$ECS-STABILITY(D) + nc-$D(D)$.	Def. 15	Page 16
ECS-STICKY(y)	The earliest subsequence in σ with a maximal common root R in at least $y+1$ consecutive rounds actually has a ECS$(y+1)$-common root.	Def. 14	Page 16
$\Diamond$ECS-STABILITY(x)	A ECS$(x+1)$-common root R exists in the graph sequence and the graph sequence is afterwards R-single rooted atleast D-times.	Def. 13	Page 16
ECS$(x+1)$-common root	A graph sequence has an ECS$(x+1)$-common root if a subsequence is R-single-rooted $x+1$ consecutive rounds.	Def. 12	Page 16
$\Diamond$STABLE$_D(x)$	The rooted-eventually-stable message adversary. $\Diamond$STABLE$_D(x)$ = ROOTED + $\Diamond$GOOD(x) + c-$D(D)$.	Def. 18	Page 17
ROOTED	All graph sequences are *rooted*.	Def. 17	Page 17
$\Diamond$GOOD(x)	The graph sequence must be R-single-rooted for atleast x consecutive rounds.	Def. 16	Page 17

nc-D(D)	If the graph is R-single-rooted for D not necessarily consecutive rounds (starting in round r_1 and ending in round r_D), then after round r_D all processes have knowledge of $R's$ state at the end of round $r_1 - 1$.	Def. 6	Page 9
c-D(D)	If the graph is R-single-rooted for D consecutive rounds (starting in round r_1 and ending in round r_D), then after round r_D all processes have knowledge of $R's$ state at the end of round $r_1 - 1$.	Def. 7	Page 9

Bibliography

[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.

[AW04] Hagit Attiya and Jennifer Welch. *Distributed computing : fundamentals, simulations, and advanced topics*. Wiley series on parallel and distributed computing. Wiley, Hoboken, NJ, 2nd ed.. edition, 2004.

[BRS+16] Martin Biely, Peter Robinson, Ulrich Schmid, Manfred Schwarz, and Kyrill Winkler. Gracefully degrading consensus and k-set agreement in directed dynamic networks. Research Report TUW-258404, Technische Universität Wien, Institut für Technische Informatik, Treitlstr. 1-3/182-2, 1040 Vienna, Austria, 2016. http://publik.tuwien.ac.at/files/publik_258404.pdf (submitted to TCS).

[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.

[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.

[DLS88] Cynthia Dwork, Nancy Lynch, and Larry Stockmeyer. Consensus in the presence of partial synchrony. *J. ACM*, 35(2):288–323, apr 1988.

[FLP85] Michael J. Fischer, Nancy A. Lynch, and Michael S. Paterson. Impossibility of distributed consensus with one faulty process. *J. ACM*, 32(2):374–382, April 1985.

[KLO10] Fabian Kuhn, Nancy A. Lynch, and Rotem Oshman. Distributed computation in dynamic networks. In *STOC*, pages 513–522, 2010.

[KO11] Fabian Kuhn and Rotem Oshman. Dynamic networks: Models and algorithms. *SIGACT News*, 42(1):82–96, March 2011.

[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.

[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.

[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, New York, NY, USA, 2016. Association for Computing Machinery.

[WSM20] Kyrill Winkler, Ulrich Schmid, and Yoram Moses. A Characterization of Consensus Solvability for Closed Message Adversaries. In Pascal Felber, Roy Friedman, Seth Gilbert, and Avery Miller, editors, *23rd International Conference on Principles of Distributed Systems (OPODIS 2019)*, volume 153 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 17:1–17:16, Dagstuhl, Germany, 2020. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.

[WSS16] Kyrill Winkler, Manfred Schwarz, and Ulrich Schmid. Consensus in directed dynamic networks with short-lived stability. *CoRR*, abs/1602.05852v1, 2016.

[WSS19] Kyrill Winkler, Manfred Schwarz, and Ulrich Schmid. Consensus in rooted dynamic networks with short-lived stability. *Distributed Computing*, 32(5):443–458, Oct 2019.