Topological Characterization of Consensus in Distributed Systems

Thomas Nowak, Ulrich Schmid, Kyrill Winkler

Topological Characterization of Consensus in Distributed Systems

Dedicated to the 2018 Dijkstra Prize winners Bowen Alpern and Fred B. Schneider

THOMAS NOWAK*, Université Paris-Saclay, CNRS, ENS Paris-Saclay, France and Institut Universitaire de France, France
ULRICH SCHMID†, TU Wien, Austria
KYRILL WINKLER‡, ITK Engineering, Austria

We provide a complete characterization of both uniform and non-uniform deterministic consensus solvability in distributed systems with benign process and communication faults using point-set topology. More specifically, we non-trivially extend the approach introduced by Alpern and Schneider in 1985, by introducing novel fault-aware topologies on the space of infinite executions: the process-view topology, induced by a distance function that relies on the local view of a given process in an execution, and the minimum topology, which is induced by a distance function that focuses on the local view of the process that is the last to distinguish two executions. Consensus is solvable in a given model if and only if the sets of admissible executions leading to different decision values is disconnected in these topologies. By applying our approach to a wide range of different applications, we provide a topological explanation of a number of existing algorithms and impossibility results and develop several new ones, including a general equivalence of the strong and weak validity conditions.

CCS Concepts: • **Theory of computation** → **Distributed algorithms**.

Additional Key Words and Phrases: Topological characterization; point-set topology; consensus; distributed systems; benign faults

ACM Reference Format:
Thomas Nowak, Ulrich Schmid, and Kyrill Winkler. 2024. Topological Characterization of Consensus in Distributed Systems: Dedicated to the 2018 Dijkstra Prize winners Bowen Alpern and Fred B. Schneider. 1, 1 (October 2024), 51 pages. https://doi.org/0000001.0000001

*Thomas Nowak has been supported by the Université Paris-Saclay project DEPEC MODE and the ANR project DREAMY (ANR-21-CE48-0003).
†Ulrich Schmid has been supported by the Austrian Science Fund (FWF) under project ADynNet (P28182), RiSE/SHiNE (S11405), DMAC (P32431), and ByzDEL (P33600).
‡Kyrill Winkler has been supported by the Austrian Science Fund (FWF) under project ADynNet (P28182) and RiSE/SHiNE (S11405). When this work was initiated, Kyrill Winkler was with TU Wien.

Authors' addresses: Thomas Nowak, Université Paris-Saclay, CNRS, ENS Paris-Saclay, Gif-sur-Yvette, France, Institut Universitaire de France, Paris, France, thomas@thomasnowak.net; Ulrich Schmid, TU Wien, Vienna, Austria, s@ecs.tuwien.ac.at; Kyrill Winkler, ITK Engineering, Austria, kyrill.winkler@itk-engineering.com.

XXXX-XXXX/2024/10-ART $15.00
https://doi.org/0000001.0000001

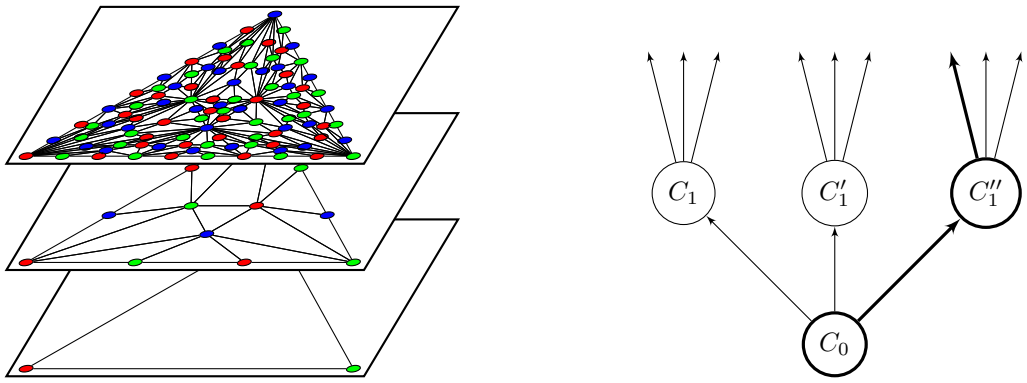

Fig. 1. Comparison of the combinatorial topology approach and the point-set topology approach: The combinatorial topology approach (left) studies sequences of increasingly refined spaces in which the objects of interest are simplices (corresponding to configurations). The point-set topology approach (right) studies a single space in which the objects of interest are executions (i.e., infinite sequences of configurations).

1 INTRODUCTION

We provide a complete characterization of the solvability of deterministic non-uniform and uniform consensus in distributed systems with benign process and/or communication failures, using point-set topology as introduced in the Dijkstra Prize-winning paper by Alpern and Schneider [4]. Our results hence precisely delimit the consensus solvability/impossibility border in very different distributed systems such as dynamic networks [27] controlled by a message adversary [2], synchronous distributed systems with processes that may crash or commit send and/or receive omission failures [40], or purely asynchronous systems with crash failures [19], for example. Whereas we will primarily focus on message-passing architectures in our examples, our topological approach also covers shared-memory systems [34].

Deterministic consensus, where every process starts with some initial input value picked from a finite set $\mathcal{V}$ and has to irrevocably compute a common output value, is arguably the most well-studied problem in distributed computing. Both impossibility results and consensus algorithm are known for virtually all distributed computing that have been proposed so far. However, they have been obtained primarily on a case-by-case basis, using classic combinatorial analysis techniques [18]. Whereas there are also some generic characterizations [31, 33], i.e., ones that can be applied to different models of computation, we are not aware of any approach that allows to precisely characterize the consensus solvability/impossibility border for arbitrary distributed systems with benign process- and communication-failures.

In this paper, we provide such a characterization based on point-set topology, as introduced by Alpern and Schneider [4]. Regarding topological methods in distributed computing, one has to distinguish point-set topology, which considers the space of infinite executions of a distributed algorithm, from combinatorial topology, which studies the topology of reachable states of prefixes of admissible executions using simplicial complexes. Fig. 1 illustrates the objects studied in combinatorial topology vs. point-set topology. As of today, combinatorial topology has been developed into a quite widely applicable tool for the analysis of distributed systems [24]. A celebrated result in this area is the *Asynchronous Computability Theorem* [25],

[21], for example, which characterizes solvable tasks in wait-free asynchronous shared memory systems with crashes.

By contrast, point-set topology has only rarely been used in distributed computing. The primary objects are the *infinite* executions of a distributed algorithm [4]. By defining a suitable metric between two infinite executions γ and δ, each considered as the corresponding infinite sequence of global states of the algorithm in the respective execution, they can be viewed as elements of a topological space. For example, according to the common-prefix metric $d_{\max}(\gamma, \delta)$, the executions γ and δ are close if the common prefix where *no* process can distinguish them is long. A celebrated general result [4] is that closed and dense sets in the resulting space precisely characterize safety and liveness properties, respectively.

Prior to our paper, however, point-set topology has only occasionally been used for establishing impossibility results. We are only aware of some early work by one of the authors of this paper on a generic topological impossibility proof for consensus in compact models [37], and a topological study of the strongly dependent decision problem [9]. Lubitch and Moran [31] introduced a construction for schedulers, which leads to limit-closed submodels[1] of classic non-closed distributed computing models (like asynchronous systems consisting of $|\Pi| = n$ processes, up to which $t < n - 1$ may crash). In a similar spirit, Kuznetsov, Rieutord, and He [28] showed, in the setting of combinatorial topology, how to reason about non-closed models by considering equivalent affine tasks that are closed. Gafni, Kuznetsov, and Manolescu [20] tried to extend the ACT to also cover some non-compact shared memory models that way. A similar purpose is served by defining layerings, as introduced by Moses and Rajsbaum [33]. Whereas such constructions of closed submodels greatly simplify impossibility proofs, they do not lead to a precise characterization of consensus solvability in non-closed models: We are not aware of any proof that there is an equivalent closed submodel for every non-closed model.

Contributions. Building on our PODC'19 paper [38] devoted to consensus in dynamic networks under message adversaries [2], the present paper provides a complete topological characterization of both the non-uniform and uniform deterministic consensus solvability/impossibility border for general distributed systems with benign process and/or communication faults. To achieve this, we had to add several new topological ideas to the setting of Alpern and Schneider [4], as detailed below, which not only allowed us to deal with both closed and non-closed models, but also provided us with a topological explanation of bivalence [19] and bipotence [33] impossibility proofs. In more detail:

(i) We introduce a simple generic system model for full-information protocols that covers all distributed system models with benign faults we are aware of. We define new topologies on the execution space of general distributed algorithms in this model, which allow us to reason about sequences of local views of (correct) processes, rather than about global configuration sequences. The *p-view topology* is defined by a distance function $d_p(\gamma, \delta)$ based on the common prefix of p's local views in the executions γ and δ. The uniform and non-uniform minimum topology are induced by the last (correct) process to notice a difference between two executions. In the appendix, we introduce process-time graphs [8] as a succinct alternative to configuration sequences in executions, and show that they are equivalent w.r.t. our topological reasoning. This is accomplished by instantiating our generic system model as an "operational" system model, based on the widely applicable modeling framework introduced by Moses and Rajsbaum [33].

[1]Informally, a model is limit-closed if the limit of a sequence of growing prefixes of admissible executions is admissible. Note that the wait-free asynchronous model is limit-closed.

(ii) We show that consensus can be modeled as a continuous decision function Δ in our topologies, which maps an admissible execution to its unique decision value. This allows us to prove that consensus is solvable if and only if all the decision sets, i.e., the pre-images $\Sigma_v = \Delta^{-1}(v)$ for every decision value $v \in \mathcal{V}$, are disconnected from each other. We also provide a universal uniform and non-uniform consensus algorithm, which rely on this separation.

(iii) We provide an alternative characterization of uniform and non-uniform consensus solvability based on the broadcastability of the decision sets and their connected components. It applies for the usual situation where every vector of values from $\mathcal{V}$ is an allowed assignment of the input values of the processes (which is not the case for condition-based consensus [34], however). Interestingly, our respective results imply that solving consensus with weak validity and consensus with strong validity is equivalent in any model with benign faults. Moreover, we provide a characterization of consensus solvability based on the limits of two infinite sequences of admissible executions, taken from different decision sets. Consensus is impossible if there is just one pair of such limits with distance 0, which actually coincide with the forever bivalent/bipotent executions constructed in previous proofs [19, 33].

(iv) We apply our topological approach to different distributed computing models. This way, we provide a topological explanation of well-known classic results like bivalence proofs and consensus solvability/impossibility in synchronous systems with general omission faults. Despite the fact that consensus has been thoroughly studied in virtually any conceivable distributed computing model, we also provide some new results: We provide a new necessary and sufficient condition for solving condition-based consensus with strong validity in asynchronous shared-memory systems [34], comprehensively characterize consensus solvability in dynamic networks with both compact and non-compact message adversaries [2], and give a novel consensus algorithm that does not rely on an implementation of the Ω failure detector for systems with an eventually timely f-source [3, 26].

Paper organization. In Section 3, we define the elements of the space that will be endowed with our new topologies in Section 4. Section 5 introduces the consensus problem in topological terms and provides our abstract characterization result for uniform consensus (Theorem 5.2) and non-uniform consensus (Theorem 5.3), which also provide universal algorithms. Alternative characterizations based on limit exclusion and broadcastability are provided in Section 6 and Section 7, respectively. Our topological characterizations are complemented by Section 8, which is devoted to applications. Some conclusions in Section 9 round off our paper. In Appendix A, we introduce process-time graphs and an operationalization of our generic system model for some classic distributed computing models.

2 RELATED WORK

Besides the few point-set topology papers [4, 9, 37] and the closed model constructions [20, 28, 31, 33] already mentioned in Section 1, there is an abundant literature on consensus algorithms and impossibility proofs.

Regarding combinatorial topology, it is worth mentioning that our study of the indistinguishability relation of prefixes of executions is closely connected to connectivity properties of the r-round protocol complex. However, in non-limit-closed models, we need to go beyond a uniformly bounded prefix length. This is in sharp contrast to the models usually considered in combinatorial topology [6, 11], which are limit-closed (typically, wait-free asynchronous).

A celebrated paper on the impossibility of consensus in asynchronous systems with crash failures is by Fischer, Lynch, and Paterson [19], who also introduced the bivalence proof technique. This impossibility can be avoided by means of unreliable failure detectors [12] or condition-based approaches restricting the allowed inputs [34]. Consensus in synchronous systems with Byzantine-faulty processes has been introduced by Lamport, Shostak, and Pease [30]. The seminal works by Dolev, Dwork, and Stockmeyer [15] and Dwork, Lynch, and Stochmeyer [16] 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. Consensus in systems with weak timely links and crash failures was considered [3, 26]. Algorithms for consensus in systems with general omission process faults were provided by Perry and Toueg [40].

Perhaps one of the earliest characterizations of consensus solvability in synchronous distributed systems prone to communication errors is the seminal work by Santoro and Widmayer [44], where it was shown that consensus is impossible if up to $n-1$ messages may be lost in each round. This classic result was refined by several authors [13, 45] and, more recently, by Coulouma, Godard, and Peters [14], where a property of an equivalence relation on the sets of communication graphs was found that captures exactly the source of consensus impossibility. Following Afek and Gafni [2], such distributed systems are nowadays known as dynamic networks, where the per-round directed communication graphs are controlled by a message adversary. Whereas the paper by Coulouma, Godard, and Peters [14] and follow-up work [46] studied oblivious message adversaries, where the communication graphs are picked arbitrarily from a set of candidate graphs, more recent papers [10, 49] studied eventually stabilizing message adversaries, which guarantee that some rounds with "good" communication graphs will eventually be generated. Note that oblivious message adversaries are limit-closed, which is not the case for message adversaries like the eventually stabilizing ones. Raynal and Stainer explored the relation between message adversaries and failure detectors [42].

The first characterization of consensus solvability under general message adversaries was provided by Fevat and Godard [17], albeit only for systems that consist of two processes. A bivalence argument was used there to show that certain communication patterns, namely, a "fair" or a special pair of "unfair" communication patterns (see Definition 6.6 for more information), must be excluded by the message adversary for consensus to become solvable.

3 GENERIC SYSTEM MODEL

We consider distributed message passing or shared memory systems made up of a set of n deterministic processes Π with unique identifiers, taken from $[n] = \{1, \ldots, n\}$ for simplicity. We denote individual processes by letters p, q, etc.

For our characterization of consensus solvability, we restrict our attention to *full-information executions*, in which processes continuously relay all the information they gathered to all other processes, and eventually apply some local decision function. The exchanged information includes the process's initial value, but also, more importantly, a record of all events (message receptions, shared memory readings, object invocations, ...) witnessed by the process. As such, our general system model is hence applicable whenever no constraints are placed on the size of the local memory and the size of values to be communicated (e.g., message/shared-register size). In particular, it is applicable to classical synchronous and asynchronous message-passing and shared-memory models, with benign process and

communication faults. In Appendix A, we will also provide a topologically equivalent "operationalization" of our generic system model built on top of process-time graphs [8], based the modeling framework introduced by Moses and Rajsbaum [33].

Formally, a (full-information) execution is a sequence of (full-information) configurations. For every process $p \in \Pi$, there is an equivalence relation $\sim_p$ on the set $\mathcal{C}$ of configurations—the p-indistinguishability relation—indicating whether process p can locally distinguish two configurations, i.e., if it has the same *view* $V_p(C) = V_p(D)$ in C and D. In this case we write $C \sim_p D$. Note that two configurations that are indistinguishable for all processes need not be equal. In fact, configurations usually include some state of the communication media that is not accessible to any process.

In addition to the indistinguishability relations, we assume the existence of a function $Ob : \mathcal{C} \to 2^\Pi$ that specifies the set of *obedient* processes in a given configuration. Obedient processes must follow the algorithm and satisfy the (consensus) specification; usually, $Ob(C)$ is the set of non-faulty processes. Again, this information is usually not accessible to the processes. We make the restriction that disobedient processes cannot recover and become obedient again, i.e., that $Ob(C) \supseteq Ob(C')$ if C' is reachable from C. We extend the obedience function to the set $\Sigma \subseteq \mathcal{C}^\omega$ of *admissible executions* in a given model by setting $Ob : \Sigma \to 2^\Pi$, $Ob(\gamma) = \bigcap_{t \geq 0} Ob(C^t)$ where $\gamma = (C^t)_{t \geq 0}$. Here, $t \in \mathbb{N}_0 = \mathbb{N} \cup \{0\}$ denotes a notion of *global* time that is not accessible to the processes. Consequently, a process is obedient in an execution if it is obedient in all of its configurations. We further make the restriction that there is at least one obedient process in every execution, i.e., that $Ob(\gamma) \neq \emptyset$ for all $\gamma \in \Sigma$. Moreover, we assume that $Ob(C) = \Pi$ for every initial configuration, in order to make input value assignments (see below) well-defined for all processes.

We also assume that every process has the possibility to weakly count the steps it has taken. Formally, we assume the existence of weak clock functions $\chi_p : \mathcal{C} \to \mathbb{N}_0$ such that for every execution $\delta = (D^t)_{t \geq 0} \in \Sigma$ and every configuration $C \in \mathcal{C}$, the relation $C \sim_p D^t$ implies $t \geq \chi_p(C)$. Additionally, we assume that $\chi_p(D^t) \to \infty$ as $t \to \infty$ for every execution $\delta \in \Sigma$ and every obedient process $p \in Ob(\delta)$. χ_p hence ensures that a configuration D^t where p has some specific view $V_p(D^t) = V_p(C)$ cannot occur before time $t = \chi_p(C)$ in any execution δ. Our weak clock functions hence allow to model lockstep synchronous rounds by choosing $\chi(D^t) = t$ for any execution $\delta = (D^t)_{t \geq 0} \in \Sigma$, but are also suitable for modeling non-lockstep, even asynchronous, executions (see Appendix A.2).

For the discussion of decision problems, we need to introduce the notion of input values, which will also be called initial values in the sequel. Since we limit ourselves to the consensus problem, we need not distinguish between the sets of input values and output values. We thus just assume the existence of a finite set $\mathcal{V}$ of potential input values, and require that the potential output values are also in $\mathcal{V}$. Furthermore, the initial configuration $I = I(\gamma)$ of any execution γ is assumed to contain an input value $I_p \in \mathcal{V}$ for every process $p \in \Pi$. This information is locally accessible to the processes, i.e., each process can access its own initial value (and those it has heard from). We assume that there is a unique initial configuration for every input-value assignment of the processes.

A *decision algorithm* is a collection of functions $\Delta_p : \mathcal{C} \to \mathcal{V} \cup \{\perp\}$ such that $\Delta_p(C) = \Delta_p(D)$ if $C \sim_p D$ and $\Delta_p(C') = \Delta_p(C)$ if C' is reachable from C and $\Delta_p(C) \neq \perp$, where $\perp \notin \mathcal{V}$ represents the fact that p has not decided yet. That is, decisions depend on local information only and are irrevocable. Every process p thus has at most one decision value in an execution. We can extend the decision function to executions by setting $\Delta_p : \Sigma \to \mathcal{V} \cup \{\perp\}$,

$\Delta_p(\gamma) = \lim_{t\to\infty} \Delta_p(C^t)$ where $\gamma = (C^t)_{t\geq 0}$. We say that p has decided value $v \neq \perp$ in configuration C or execution γ if $\Delta_p(C) = v$ or $\Delta_p(\gamma) = v$, respectively.

We will consider both non-uniform and uniform consensus with either weak or strong validity as our decision tasks, which are defined as follows:

Definition 3.1 (Non-uniform and uniform consensus). A *non-uniform consensus* algorithm $\mathcal{A}$ is a decision algorithm that ensures the following properties in all of its admissible executions:

(T) Eventually, every obedient process must irrevocably decide. (Termination)
(A) If two obedient processes have decided, then their decision values are equal. (Agreement)
(V) If the initial values of processes are all equal to v, then v is the only possible decision value. (Validity)

In a *strong consensus* algorithm $\mathcal{A}$, weak validity (V) is replaced by

(SV) The decision value must be the input value of some process. (Strong Validity)

A *uniform consensus* algorithm $\mathcal{A}$ must ensure (T), (V) or (SV), and

(UA) If two processes have decided, then their decision values are equal. (Uniform Agreement)

Note that we will primarily focus on consensus with weak validity, which is the usual meaning of the term consensus unless otherwise noted.

By Termination, Agreement, and the fact that every execution has at least one obedient process, for every consensus algorithm, we can define the consensus decision function $\Delta : \Sigma \to \mathcal{V}$ by setting $\Delta(\gamma) = \Delta_p(\gamma)$ where p is any process that is obedient in execution γ, i.e., $p \in Ob(\gamma)$. Recall that the initial value of process p in the execution γ is denoted $I_p(\gamma)$ or just I_p if γ is clear from the context.

To illustrate[2] the difference between uniform and non-uniform consensus, as well as to motivate the two topologies serving to characterize their solvability, consider the example of two synchronous non-communicating processes. The set of processes is $\Pi = \{1, 2\}$ and the set of possible values is $\mathcal{V} = \{0, 1\}$. Processes proceed in lock-step synchronous rounds, but do not communicate. Thus, the only information a process has access to is its own initial value and the current time. The set of executions Σ and the obedience function Ob are defined such that one of the processes eventually becomes disobedient in every execution, but not both processes. In this model, it is trivial to solve non-uniform consensus by immediately deciding on one's own initial value, but uniform consensus is impossible.

4 TOPOLOGICAL STRUCTURE OF FULL-INFORMATION EXECUTIONS

In this section, we will endow the various sets introduced in Section 3 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 [36].

A topology on 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 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.

[2]We chose this simplistic illustrating example in order not to obfuscate the essentials. See Section 8 for more realistic examples.

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. Space X is called compact if every family of open sets that covers X contains a finite sub-family that covers X.

If X is a nonempty set, then we call any function $d : X \times X \to \mathbb{R}_+$ a *distance function* on X. Define $\mathcal{T}_d \subseteq 2^X$ by setting $U \in \mathcal{T}_d$ if and only if for all $x \in U$ there exists some $\varepsilon > 0$ such that $B_\varepsilon(x) = \{y \in X \mid d(x, y) < \varepsilon\} \subseteq U$.

Many topological spaces are defined by metrics, i.e., symmetric, positive definite distance functions for which the triangle inequality $d(x, y) \leq d(x, z) + d(z, y)$ holds for any $x, y, z \in X$. For a distance function to define a (potentially non-metrizable) topology though, no additional assumptions are necessary:

LEMMA 4.1. *If d is a distance function on X, then $\mathcal{T}_d$ is a topology on X.*

PROOF. Firstly, we show that $\mathcal{T}_d$ is closed under unions. So let $\mathcal{U} \subseteq \mathcal{T}_d$. We will show that $\bigcup \mathcal{U} \in \mathcal{T}_d$. 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 $U \in \mathcal{T}_d$, there exists some $\varepsilon > 0$ such that

$$B_\varepsilon(x) \subseteq U \subseteq \bigcup \mathcal{U} \ ,$$

which shows that $\bigcup \mathcal{U} \in \mathcal{T}_d$.

Secondly, we show that $\mathcal{T}_d$ is closed under finite intersections. Let $U_1, U_2, \ldots, U_k \in \mathcal{T}_d$. We will show that $\bigcap_{\ell=1}^k U_\ell \in \mathcal{T}_d$. Let $x \in \bigcap_{\ell=1}^k U_\ell$. Then, by definition of the set intersection, $x \in U_\ell$ for all $1 \leq \ell \leq k$. Because all U_ℓ are in $\mathcal{T}_d$, there exist $\varepsilon_1, \varepsilon_2, \ldots, \varepsilon_k > 0$ such that $B_{\varepsilon_\ell}(x) \subseteq U_\ell$ for all $1 \leq \ell \leq k$. 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 $\gamma \leq \delta$, we also have

$$B_\varepsilon(x) \subseteq B_{\varepsilon_\ell}(x) \subseteq U_\ell$$

for all $1 \leq \ell \leq k$. But this shows that $B_\varepsilon(x) \subseteq \bigcap_{\ell=1}^k U_\ell$, which means that $\bigcap_{\ell=1}^k U_\ell \in \mathcal{T}_d$.

Since it is easy to check that $\emptyset, X \in \mathcal{T}_d$ as well, $\mathcal{T}_d$ is indeed a topology. □

We will henceforth refer to $\mathcal{T}_d$ as the topology induced by d.

An execution is a sequence of configurations, i.e., an element of the product space $\mathcal{C}^\omega$. Since our primary object of study are executions, we will endow this space with a topology as follows: The product topology, which is a distinguished topology on any product space $\Pi_{\iota \in I} X_\iota$ of topological spaces, is defined as the coarsest topology such that all projection maps $\pi_i : \Pi_{\iota \in I} X_\iota \to X_i$ (where π_i extracts the i-th element of the sequence) are continuous. Recall that a topology $\mathcal{T}'$ is coarser than a topology $\mathcal{T}$ for the same space if every open set $U \in \mathcal{T}'$ is also open in $\mathcal{T}$.

It turns out that the product topology on the space $\mathcal{C}^\omega$ is induced by a distance function, whose form is known in a special case that covers our needs:

LEMMA 4.2. *Let d be a distance function on X that only takes the values 0 or 1. Then the product topology $\mathcal{T}^\omega$ of X^ω, where every copy of X is endowed with the topology induced by d, is induced by the distance function*

$$X^\omega \times X^\omega \to \mathbb{R} \quad , \quad (\gamma, \delta) \mapsto 2^{-\inf\{t \geq 0 \mid d(C^t, D^t) > 0\}}$$

where $\gamma = (C^t)_{t \geq 0}$ and $\delta = (D^t)_{t \geq 0}$.

Proof. We first show that all projections $\pi^t : X^\omega \to X$ are continuous when endowing X^ω with the product topology $\mathcal{T}^\omega$: Let $U \subseteq X$ be open and $C \in U$, i.e., $d(C, D) = 0$ implies $D \in U$. Let $\gamma = (C^t)_{t\geq 0} \in (\pi^t)^{-1}[U]$ and set $\varepsilon = 2^{-t}$. Then,

$$\begin{aligned} B_\varepsilon(\gamma) &= \left\{\delta = (D^t)_{t\geq 0} \in X^\omega \mid \forall 0 \leq s \leq t\colon d(C^s, D^s) = 0\right\} \\ &\subseteq \left\{\delta = (D^t)_{t\geq 0} \in X^\omega \mid d(C^t, D^t) = 0\right\} \\ &= (\pi^t)^{-1}\left[\{D \in X \mid d(C^t, D) = 0\}\right] \subseteq (\pi^t)^{-1}[U], \end{aligned}$$

where the last inclusion follows from the openness of U. Since $(\pi^t)^{-1}[U]$ is hence open in $\mathcal{T}^\omega$, the continuity of π^t follows.

Let now $\mathcal{T}_0$ be an arbitrary topology on X^ω for which all projections π^t are continuous. We will show that $\mathcal{T}^\omega \subseteq \mathcal{T}_0$, which reveals that $\mathcal{T}^\omega$ is the coarsest topology with continuous projections, i.e., the product topology of X^ω where every copy of X is endowed by $\mathcal{T}_d$. This will establish our lemma.

So let $E \in \mathcal{T}^\omega$ and take any $\gamma = (C^t)_{t\geq 0} \in E$. There exists some $\varepsilon > 0$ such that $B_\varepsilon(\gamma) \subseteq E$. Choose $t \in \mathbb{N}_0$ such that $2^{-t} \leq \varepsilon$, and set

$$\begin{aligned} F &= \Big(\prod_{s=0}^{t} B_1(C^s)\Big) \times X^\omega = \bigcap_{s=0}^{t} (\pi^s)^{-1}\left[B_1(C^s)\right] \\ &\subseteq \left\{\delta = (D^t)_{t\geq 0} \in X^\omega \mid \forall 0 \leq s \leq t\colon d(C^s, D^s) = 0\right\} = B_\varepsilon(\gamma)\ . \end{aligned}$$

Then, F is open with respect to $\mathcal{T}_0$ as a finite intersection of open sets: After all, every $(\pi^s)^{-1}\left[B_1(C^s)\right]$ is open by the continuity of the projection π^s. But since $F \subseteq B_\varepsilon(\gamma) \subseteq E$, this shows that E contains a $\mathcal{T}_0$-open neighborhood for each of its points, i.e., $E \in \mathcal{T}_0$. □

4.1 Process-view distance function for executions

In previous work on point-set topology in distributed computing [4, 37], the set of configurations $\mathcal{C}$ of some fixed algorithm $\mathcal{A}$ was endowed with the discrete topology, where every subset $U \subseteq \mathcal{C}$ is open. The discrete topology is induced by the discrete metric $d_{\max}(C, D) = 1$ if $C \neq D$ and 0 otherwise (for configurations $C, D \in \mathcal{C}$). Moreover, $\mathcal{C}^\omega$ was endowed with the corresponding product topology, which is induced by the *common-prefix metric*

$$d_{\max} : \Sigma \times \Sigma \to \mathbb{R}_+ \quad , \quad d_{\max}(\gamma, \delta) = 2^{-\inf\{t\geq 0 \mid C^t \neq D^t\}}\ ,$$

where $\gamma = (C^t)_{t\geq 0}$ and $\delta = (D^t)_{t\geq 0}$, according to Lemma 4.2. Informally, $d_{\max}(\gamma, \delta)$ decreases with the length of the common prefix where *no* process can distinguish γ and δ.

By contrast, we define the *p-view distance function* d_p on the set $\mathcal{C}$ of configurations for every process $p \in \Pi$ by

$$d_p(C, D) = \begin{cases} 0 & \text{if } C \sim_p D \text{ and } p \in Ob(C) \cap Ob(D), \text{ or } C = D \\ 1 & \text{otherwise}\ . \end{cases}$$

Extending this distance function from configurations to executions, we define the *p-view distance function* by

$$d_p : \Sigma \times \Sigma \to \mathbb{R}_+ \quad , \quad d_p(\gamma, \delta) = 2^{-\inf\{t\geq 0 \mid d_p(C^t, D^t) > 0\}}$$

where $\gamma = (C^t)_{t\geq 0}$ and $\delta = (D^t)_{t\geq 0}$.

Figure 2 illustrates the distance function d_p, and Lemma 4.3 reveals that it defines a pseudometric:

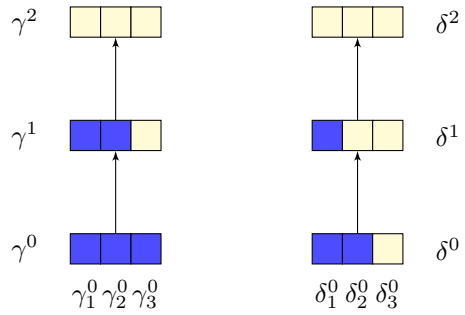

Fig. 2. Comparison of the p-view and common-prefix metric. The first three configurations of each of the two executions γ and δ with three processes and two different possible local states (dark blue and light yellow) are depicted. We have $d_{\max}(\gamma,\delta) = d_3(\gamma,\delta) = 1$ and $d_2(\gamma,\delta) = 1/2$.

Lemma 4.3 (Pseudometric d_p). *The p-view distance function d_p is a pseudometric, i.e., it satisfies:*

$$\begin{aligned} d_p(\gamma,\gamma) &= 0 \\ d_p(\gamma,\delta) &= d_p(\delta,\gamma) && \textit{(symmetry)} \\ d_p(\beta,\delta) &\le d_p(\beta,\gamma) + d_p(\gamma,\delta) && \textit{(triangle inequality)} \end{aligned}$$

Proof. We have $d_p(\gamma,\gamma) = 0$ since $d_p(C^t, C^t) = 0$ for all $t \ge 0$ where $\gamma = (C^t)_{t\ge 0}$. Symmetry follows immediately from the definition. As for the triangle inequality, write $\beta = (B^t)_{t\ge 0}$, $\gamma = (C^t)_{t\ge 0}$, and $\delta = (D^t)_{t\ge 0}$. We have:

$$\max\{d_p(\beta,\gamma), d_p(\gamma,\delta)\} = 2^{-\inf\{t\ge 0 \mid d_p(B^t,C^t)>0 \,\vee\, d_p(C^t,D^t)>0\}}$$

Since $d_p(B^t,D^t) > 0 \implies d_p(B^t,C^t) > 0 \,\vee\, d_p(C^t,D^t) > 0$, it follows that

$$\inf\{t \ge 0 \mid d_p(B^t,D^t) > 0\} \ge \inf\{t \ge 0 \mid d_p(B^t,C^t) > 0 \,\vee\, d_p(C^t,D^t) > 0\}$$

and thus

$$d_p(\beta,\delta) \le \max\{d_p(\beta,\gamma), d_p(\gamma,\delta)\} \ ,$$

which concludes the proof. □

4.2 Uniform topology for executions

The *uniform minimum topology* (abbreviated *uniform topology*) on the set Σ of executions is induced by the distance function

$$d_{\mathrm{u}}(\gamma,\delta) = \min_{p\in\Pi} d_p(\gamma,\delta) \ .$$

Note that d_{u} does not necessarily satisfy the triangle inequality (nor definiteness): There may be executions with $d_p(\beta,\gamma) = 0$ and $d_q(\gamma,\delta) = 0$ but $d_r(\beta,\delta) > 0$ for all $r \in \Pi$. Hence, the topology on $\mathcal{C}^\omega$ induced by d_{u} lacks many of the convenient (separation) properties of metric spaces, but will turn out to be sufficient for the characterization of the solvability of uniform consensus (see Theorem 5.2).

The next lemma shows that the decision function of an algorithm that solves uniform consensus is always continuous with respect to the uniform topology.

Lemma 4.4. *Let $\Delta : \Sigma \to \mathcal{V}$ be the consensus decision function of a uniform consensus algorithm. Then, Δ is continuous with respect to the uniform distance function d_{u}.*

Proof. Let $v \in \mathcal{V}$ and let $\Sigma_v = \Delta^{-1}[\{v\}]$ be its inverse image under the decision function Δ. We will show that for all executions $\gamma \in \Sigma_v$ there exists a time T such that $B_{2^{-T}}(\gamma) \subseteq \Sigma_v$, proving that Σ_v is open. Since the singleton sets $\{v\}$ form a base of the discrete topology on $\mathcal{V}$, continuity follows.

Let $\gamma \in \Sigma_v$. Let T be a time greater than both the latest decision time of the processes in $Ob(\gamma)$ and the latest time any process becomes disobedient in execution $\gamma = (C^t)_{t\geq 0}$. By the Termination property and the fact that disobedient processes cannot become obedient again, we have $T < \infty$. Because T is larger than the latest time a process becomes disobedient, we have $Ob(\gamma) = Ob(C^T)$.

Using the notation $\gamma = (C^t)_{t\geq 0}$ and $\delta = (D^t)_{t\geq 0}$, we have:

$$\begin{aligned} B_{2^{-T}}(\gamma) &= \left\{\delta \in \Sigma \mid d_{\mathrm{u}}(\gamma,\delta) < 2^{-T}\right\} \\ &= \left\{\delta \in \Sigma \mid \exists p \in \Pi\colon d_p(C^t, D^t) < 2^{-T}\right\} \\ &= \left\{\delta \in \Sigma \mid \exists p \in \Pi\ \forall t \leq T\colon C^t \sim_p D^t \wedge p \in Ob(C^t) \cap Ob(D^t)\right\} \\ &= \left\{\delta \in \Sigma \mid \exists p \in \Pi\colon C^T \sim_p D^T \wedge p \in Ob(C^T) \cap Ob(D^T)\right\} \end{aligned}$$

If $\delta \in B_{2^{-T}}(\gamma)$, then $C^T \sim_p D^T$ for some $p \in Ob(C^T) \cap Ob(D^T)$. Since p has decided $\Delta(\gamma)$ at time T in execution γ and p is obedient until time T in execution δ, process p has also decided $\Delta(\gamma)$ at time T in execution δ. By Uniform Agreement and Termination, all processes in $Ob(\delta)$ decide $\Delta(\gamma) = v$ as well. In other words $B_{2^{-T}}(\gamma) \subseteq \Sigma_v$, which concludes the proof. □

For an illustration in our non-communicating two-process example, denote by $\gamma^{(T)}$ the execution in which process 1 has initial value 0, process 2 has initial value 1, and process 1 becomes disobedient at time T. Similarly, denote by $\delta^{(U)}$ the execution with the same initial values and in which process 2 becomes disobedient at time U. Since there is no means of communication between the two processes, by Validity, each obedient process necessarily has to eventually decide on its own initial value, i.e., $\Delta(\gamma^{(T)}) = 1$ and $\Delta(\delta^{(T)}) = 0$. The uniform distance between these executions is equal to $d_{\mathrm{u}}(\gamma^{(T)}, \delta^{(U)}) = 2^{-\max\{T,U\}}$. Thus, every ε-neighborhood of $\gamma^{(T)}$ contains execution $\delta^{(U)}$ if U is chosen large enough to ensure $2^{-U} < \varepsilon$. The set of 1-deciding executions is thus not open in the uniform topology. But this means that the algorithm's decision function Δ cannot be continuous. Lemma 4.4 hence implies that there is no uniform consensus algorithm in the non-communicating two-process model (which is also confirmed by the more realistic application example in Section 8.2).

4.3 Non-uniform topology for executions

Whereas the p-view distance function given by Section 4.1 is also adequate for non-uniform consensus, this is not the case for the uniform distance function as defined in Section 4.2. The appropriate *non-uniform minimum topology* (abbreviated *non-uniform topology*) on the set Σ of executions is induced by the distance function

$$d_{\mathrm{nu}}(\gamma,\delta) = \begin{cases} \min_{p \in Ob(\gamma) \cap Ob(\delta)} d_p(\gamma,\delta) & \text{if } Ob(\gamma) \cap Ob(\delta) \neq \emptyset \\ 1 & \text{if } Ob(\gamma) \cap Ob(\delta) = \emptyset \ . \end{cases}$$

Like for d_{u}, neither definiteness nor the triangle inequality need to be satisfied by d_{nu}. The resulting non-uniform topology is finer than the uniform topology, however, since the minimum is taken over the smaller set $Ob(\gamma) \cap Ob(\delta) \subseteq \Pi$, which means that $d_{\mathrm{u}}(\gamma,\delta) \leq d_{\mathrm{nu}}(\gamma,\delta)$. In particular, this implies that every decision function that is continuous with respect to the uniform topology is also continuous with respect to the non-uniform topology.

Of course, this also follows from Lemma 4.4 and the fact that every uniform consensus algorithm also solves non-uniform consensus.

The following Lemma 4.5 is the analog of Lemma 4.4:

Lemma 4.5. *Let $\Delta : \Sigma \to \mathcal{V}$ be the consensus decision function of a non-uniform consensus algorithm. Then, Δ is continuous with respect to the non-uniform distance function d_{nu}.*

Proof. We again prove that every inverse image $\Sigma_v = \Delta^{-1}[\{v\}]$ of a value $v \in \mathcal{V}$ is open.

Let $\gamma \in \Sigma_v$. Let T be the latest decision time of the processes in $Ob(\gamma)$ in execution γ. By the Termination property, we have $T < \infty$. Using the notation $\gamma = (C^t)_{t\geq 0}$ and $\delta = (D^t)_{t\geq 0}$, we have:

$$\begin{aligned} B_{2^{-T}}(\gamma) &= \left\{\delta \in \Sigma \mid d_{\mathrm{nu}}(\gamma,\delta) < 2^{-T}\right\} \\ &= \left\{\delta \in \Sigma \mid \exists p \in Ob(\gamma) \cap Ob(\delta) \colon d_p(\gamma,\delta) < 2^{-T}\right\} \\ &= \left\{\delta \in \Sigma \mid \exists p \in Ob(\gamma) \cap Ob(\delta) \colon \forall t \leq T \colon C^t \sim_p D^t\right\} \end{aligned}$$

If $\delta \in B_{2^{-T}}(\gamma)$, then $C^T \sim_p D^T$ for some $p \in Ob(\gamma) \cap Ob(\delta)$. Denote by T_p the decision time of process p in γ. Since $T_p \leq T$, we also have $C^{T_p} \sim_p D^{T_p}$ But this means that process p decides value $\Delta(\gamma)$ at time T_p in both executions γ and δ, hence $\Delta(\delta) = \Delta(\gamma) = v$ and $B_{2^{-T}}(\gamma) \subseteq \Sigma_v$. □

For an illustration in the non-communicating two-process example used in Section 4.2, note that the trivial algorithm that immediately decides on its initial value satisfies $\Delta(\gamma^{(T)}) = 1$ and $\Delta(\delta^{(U)}) = 0$. The algorithm does solve non-uniform consensus, since it is guaranteed that one of the processes eventually becomes disobedient. In contrast to the uniform distance function, the non-uniform distance function satisfies $d_{\mathrm{nu}}(\gamma^{(T)}, \delta^{(U)}) = 1$ since $Ob(\gamma^{(T)}) \cap Ob(\delta^{(U)}) = \emptyset$. This means that the minimum distance between any 0-deciding and any 1-deciding execution is at least 1. It is hence possible to separate the two sets of executions by sets that are open in the non-uniform topology, so consensus is solvable here, according to the considerations in the following section. Again, this is confirmed by the more realistic application example in Section 8.2.

5 GENERAL CONSENSUS CHARACTERIZATION FOR FULL-INFORMATION EXECUTIONS

In this section, we will provide our main topological conditions for uniform and non-uniform consensus solvability.

Definition 5.1 (v-valent execution). We call an execution $\gamma_v \in \Sigma$, for $v \in \mathcal{V}$, v-valent, if it starts from an initial configuration I where all processes $p \in \Pi$ have the same initial value $I_p = v$.

Theorem 5.2 (Characterization of uniform consensus). *Uniform consensus is solvable if and only if there exists a partition of the set Σ of admissible executions into sets Σ_v, $v \in \mathcal{V}$, such that the following holds:*

(1) *Every Σ_v is a clopen set in Σ with respect to the uniform topology induced by d_{u}.*
(2) *If execution $\gamma \in \Sigma$ is v-valent, then $\gamma \in \Sigma_v$.*

Proof. ($\Rightarrow$): Define $\Sigma_v = \Delta^{-1}(v)$, where Δ is the decision function of a uniform consensus algorithm. This is a partition of Σ by Termination, and Validity implies property (2). It thus only remains to show openness of the Σ_v (which immediately implies clopenness, as

$\Sigma \setminus \Sigma_v = \bigcup_{v \neq w \in \mathcal{V}} \Sigma_w$ must be open), which follows from the continuity of $\Delta : \Sigma \to \mathcal{V}$, since every singleton set $\{v\}$ is open in the discrete topology.

($\Leftarrow$): We define a uniform consensus algorithm by defining the decision functions $\Delta_p : \mathcal{C} \to \mathcal{V} \cup \{\bot\}$ as

$$\Delta_p(C) = \begin{cases} v & \text{if } \{\delta \in \Sigma \mid \exists t \colon C \sim_p D^t\} \subseteq \Sigma_v, \\ \bot & \text{otherwise,} \end{cases}$$

where we use the notation $\delta = (D^t)_{t \geq 0}$. The function Δ is well defined since the sets Σ_v are pairwise disjoint.

We first show Termination of the resulting algorithm. Let $\gamma \in \Sigma$, let $v \in \mathcal{V}$ such that $\gamma \in \Sigma_v$, and let $p \in Ob(\gamma)$. Since Σ_v is open with respect to the uniform topology, there exists some $\varepsilon > 0$ such that $\{\delta \in \Sigma \mid d_{\mathrm{u}}(\gamma, \delta) < \varepsilon\} \subseteq \Sigma_v$. By definition of d_{u}, we have $d_{\mathrm{u}}(\gamma, \delta) \leq d_p(\gamma, \delta)$ and hence $\{\delta \in \Sigma \mid d_p(\gamma, \delta) < \varepsilon\} \subseteq \{\delta \in \Sigma \mid d_{\mathrm{u}}(\gamma, \delta) < \varepsilon\} \subseteq \Sigma_v$.

Writing $\gamma = (C^t)_{t \geq 0}$, let T be the smallest integer such that $2^{-\chi_p(C^t)} \leq \varepsilon$ for all $t \geq T$. Such a T exists since $\chi_p(C^t) \to \infty$ as $t \to \infty$. Then, for every $t \geq T$, we have $\{\delta \in \Sigma \mid \exists s \colon C^t \sim_p D^s\} \subseteq \left\{\delta \in \Sigma \mid d_p(\gamma, \delta) < 2^{-\chi_p(C^t)}\right\} \subseteq \Sigma_v$. In particular, $\Delta_p(C^t) = v$ for all $t \geq T$, i.e., process p decides value v in execution γ.

We next show Uniform Agreement. For the sake of a contradiction, assume that process q decides value $w \neq v$ in configuration C in execution $\gamma \in \Sigma_v$. But then, by definition of the function Δ_q, we have $\gamma \in \{\delta \in \Sigma \mid \exists t \colon C \sim_q D^t\} \subseteq \Sigma_w$. But this is impossible since $\Sigma_v \cap \Sigma_w = \emptyset$.

Validity immediately follows from property (2). □

Theorem 5.3 (Characterization of non-uniform consensus). *Non-uniform consensus is solvable if and only if there exists a partition of the set Σ of admissible executions into sets Σ_v, $v \in \mathcal{V}$, such that the following holds:*

(1) *Every Σ_v is a clopen set in Σ with respect to the non-uniform topology induced by d_{nu}.*
(2) *If execution $\gamma \in \Sigma$ is v-valent, then $\gamma \in \Sigma_v$.*

Proof. The proof is similar to that of Theorem 5.2, except that the definition of Δ_p is

$$\Delta_p(C) = \begin{cases} v & \text{if } \{\delta \in \Sigma \mid \exists t \colon C \sim_p D^t \wedge p \in Ob(\delta)\} \subseteq \Sigma_v, \\ \bot & \text{otherwise ,} \end{cases}$$

i.e., we just have to add the constraint that $p \in Ob(\delta)$ to the executions considered in the proof. □

If Σ has only finitely many connected components, i.e., only finitely many maximal connected sets, then every connected component is necessarily clopen. Consequentely, these characterizations give rise to the following meta-procedure for determining whether consensus is solvable and constructing an algorithm if it is. It requires knowledge of the connected components of the space Σ of admissible executions with respect to the appropriate topology:

(1) Initially, start with an empty set Σ_v for every value $v \in \mathcal{V}$.
(2) Add to Σ_v the connected components of Σ that contain an execution with a v-valent initial configuration.
(3) Add any remaining connected component of Σ to an arbitrarily chosen set Σ_v.
(4) If the sets Σ_v are pairwise disjoint, then consensus is solvable. In this case, the sets Σ_v determine a consensus algorithm via the universal algorithm given in the proofs of

Theorem 5.2 and Theorem 5.3. If the Σ_v are not pairwise disjoint, then consensus is not solvable.

Obviously, our solution algorithms need to know the decision sets Σ_v, $v \in \mathcal{V}$. As they usually contain uncountable many infinite executions, the question of how to obtain them in practice appears. In Section 8, we will provide several instances of *labeling algorithms*, which can be used here. They are based on labeling prefixes of executions, so can in principle even be computed incrementally by the processes on-the-fly during the executions.

6 LIMIT-BASED CONSENSUS CHARACTERIZATION

It is possible to shed some additional light on our general consensus characterization by considering limit points. In particular, Theorem 6.4 will show that consensus is impossible if and only if certain limit points in the appropriate topologies are admissible.

Definition 6.1 (Distance of sets). For $A, B \subseteq \mathcal{C}^\omega$ with distance function d, let $d(A, B) = \inf\{d(\alpha, \beta) \mid \alpha \in A,\, \beta \in B\}$.

Before we state our general results, we illustrate the underlying idea in a slightly restricted setting, namely when the underlying space Σ of configuration sequences is contained in a compact set $K \subseteq \mathcal{C}^\omega$. Whereas one cannot assume this in general, it can be safely assumed in settings where the operationalization of our system model based on process-time graphs, as described in Appendix A, applies: Since the set of all process-time graphs $\mathcal{PT}^\omega$ turns out to be compact and the transition function $\hat{\tau} : \mathcal{PT}^\omega \to \mathcal{C}^\omega$ is continuous, according to Lemma A.2, we can consider the compact set $K = \hat{\tau}(\mathcal{PT}^\omega)$ instead of $\mathcal{C}^\omega$. In this case, it is not difficult to show that $d_p(A, B) = 0$ if and only if there is a sequence of executions α_k in A and a sequence of executions β_k in B such that both sequences converge to the same limit with respect to d_p.

This distance-based characterization allows us to distinguish 3 cases that cause $d_p(A, B) = 0$: (i) If $\hat{\alpha} \in A \cap B \neq \emptyset$, one can choose the sequences defined by $\alpha_k = b_k = \hat{\alpha} = \hat{\beta}$, $k \geq 1$. (ii) If $A \cap B = \emptyset$ and $\hat{\alpha} = \hat{\beta}$, there is a "fair" execution [17] as the common limit. (iii) If $A \cap B = \emptyset$ and $\hat{\alpha} \neq \hat{\beta}$, there is a pair of "unfair" executions [17] acting as limits, which have distance 0 (and are hence also common limits w.r.t. the distance function d_p). We note, however, that due to the non-definiteness of the pseudometric d_p (recall Lemma 4.3) and the resulting non-uniqueness of limits in the p-view topology, (iii) are actually two instances of (ii). Corollary 6.7 below will reveal that consensus is solvable if and only if no decision set Σ_v contains any fair or unfair execution w.r.t any Σ_w, $v \neq w$.

Unfortunately, generalizing the above distance-based characterizaion from p-view-topologies to the uniform and non-uniform topologies is not possible: Albeit every convergent infinite sequence (α^t) w.r.t. d_{u} (Section 4.2) resp. d_{nu} (Section 4.3) also contains a convergent subsequence w.r.t. some (obedient) d_p by the pigeonhole principle, one might observe a different $d_{p'}$ for the convergent subsequence of (β^t). In this case, not even $d_p(\hat{\alpha}, \hat{\beta}) = 0$ or $d_{p'}(\hat{\alpha}, \hat{\beta}) = 0$ would guarantee $d_{\mathrm{u}}(A, B) = 0$ resp. $d_{\mathrm{nu}}(A, B) = 0$, as the triangle inequality does not hold in these topologies.

On the other hand, $d_{\mathrm{u}}(A, B) = 0$ resp. $d_{\mathrm{nu}}(A, B) = 0$ is trivially guaranteed if it is the case that $\hat{\alpha} \in B$ or $\hat{\beta} \in A$: If, say, $\hat{\alpha} \in B$, one can choose the constant sequence $(\beta^t) = (\hat{\alpha}) \in B^\omega$, which obviously converges to $\hat{\alpha}$ in any p-view topology, including the particular d_p obtained for the convergent subsequence of $(\alpha^t) \to \hat{\alpha}$ by the abovementioned pigeonhole argument. Consequently, $d_p(A, B) = 0$ and hence also $d_{\mathrm{u}}(A, B) = 0$ resp. $d_{\mathrm{nu}}(A, B) = 0$. This implies the

following "if-part" of our distance-based characterization, which even holds for non-compact $\mathcal{C}^\omega$:

Lemma 6.2 (General zero-distance condition). *Let A, B be arbitrary subsets of $\mathcal{C}^\omega$ with distance function d. If there are infinite sequences $(\alpha_k) \in A^\omega$ and $(\beta_k) \in B^\omega$ of executions, as well as $\hat{\alpha}, \hat{\beta} \in \mathcal{C}^\omega$ with $\alpha_k \to \hat{\alpha}$ and $\beta_k \to \hat{\beta}$ with $d(\hat{\alpha}, \hat{\beta}) = 0$ and $\hat{\alpha} \in B$ or $\hat{\beta} \in A$, then d(A,B)=0.*

In order to obtain the general limit-based consensus characterization stated in Theorem 6.4 below, we will not use set distances directly, however, but rather the following Separation Lemma 6.3 from [36]:

Lemma 6.3 (Separation Lemma [36, 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. Moreover, A and B of a separation of Y are clopen in 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. □

Applying Lemma 6.3 to the findings of Theorem 5.2 resp. Theorem 5.2, the following general consensus characterization can be proved:

Theorem 6.4 (Separation-based consensus characterization). *Uniform resp. non-uniform consensus is solvable in a model if and only if there exists a partition of the set of admissible executions Σ into decision sets $\Sigma_v, v \in \mathcal{V}$, such that the following holds:*

(1) *No Σ_v contains a limit point of any other Σ_w w.r.t. the uniform resp. non-uniform topology in $\mathcal{C}^\omega$.*
(2) *Every v-valent admissible execution γ_v satisfies $\gamma_v \in \Sigma_v$.*

If consensus is not solvable, then $d_{\mathrm{u}}(\Sigma_v, \Sigma_w) = 0$ resp. $d_{\mathrm{nu}}(\Sigma_v, \Sigma_w) = 0$ for some $w \neq v$.

Proof. ($\Leftarrow$) We need to prove that if (1) and (2) in the statement of our theorem hold, then consensus is solvable by means of the algorithm given in Theorem 5.2 resp. Theorem 5.3. This only requires showing that all of the finitely many Σ_v, $v \in \mathcal{V}$, are clopen in Σ, which immediately follows from Lemma 6.3 since Σ_v and $\Sigma \setminus \Sigma_v$ form a separation of Σ.

($\Rightarrow$) We prove the contrapositive, by showing that if (1) and (2) do not hold, then either some Σ_v is not closed or $\Sigma_v \cap \Sigma_w \neq 0$, which does not allow to solve consensus by Theorem 5.2 resp. Theorem 5.3. If, say, $A = \Sigma_v$ contains any limit point of $B = \Sigma_w$ for $v \neq w$, this means that there is a sequence of executions $(\beta_k) \in B^\omega$ with limit $\beta_k \to \beta$ and some $\alpha \in A \subseteq \Sigma$ with $d_{\mathrm{u}}(\alpha, \beta) = 0$ resp. $d_{\mathrm{nu}}(\alpha, \beta) = 0$. According to Lemma 6.2, we have $d_{\mathrm{u}}(A, B) = 0$ resp. $d_{\mathrm{nu}}(A, B) = 0$ in this case. If $\alpha \notin B$, then $B = \Sigma_w$ is not closed, if $\alpha \in B$, then $A \cap B \neq \emptyset$, which provides the required contradiction in either case. □

Note that Theorem 6.4 immediately implies the following properties of the distances of the decision sets in the case consensus is solvable in a model:

Corollary 6.5 (General decision set distances). *If uniform resp. non-uniform consensus is solvable in a model, it may nevertheless be the case that* $d_{\mathrm{u}}(\Sigma_v, \Sigma_w) = 0$ *resp.* $d_{\mathrm{nu}}(\Sigma_v, \Sigma_w) = 0$ *for some* $v, w \neq v$*. On the other hand, if* $d_{\mathrm{u}}(\Sigma_v, \Sigma_w) > 0$ *resp.* $d_{\mathrm{nu}}(\Sigma_v, \Sigma_w) > 0$ *for all* $v, w \neq v$*, then uniform resp. non-uniform consensus is solvable.*

Our characterization Theorem 6.4 can also be expressed via the exclusion of fair/unfair executions [17]:

Definition 6.6 (Fair and unfair executions). Consider two executions $\rho, \rho' \in \mathcal{C}^\omega$ of some consensus algorithm with decision sets Σ_v, $v \in \mathcal{V}$, in any appropriate topology:

- ρ is called *fair*, if for some $v, w \neq v \in \mathcal{V}$ there are convergent sequences $(\alpha_k) \in \Sigma_v$ and $(\beta_k) \in \Sigma_w$ with $\alpha_k \to \rho$ and $\beta_k \to \rho$.
- ρ, ρ' are called a pair of *unfair* executions, if for some $v, w \neq v \in \mathcal{V}$ there are convergent sequences $(\alpha_k) \in \Sigma_v$ with $\alpha_k \to \rho$ and $(\beta_k) \in \Sigma_w$ with $\beta_k \to \rho'$ and ρ and ρ' have distance 0.

From Theorem 6.4, we immediately obtain:

Corollary 6.7 (Fair/unfair consensus characterization). *Condition (1) in Theorem 6.4 is equivalent to requireing that the decision sets* Σ_v*,* Σ_w *for* $w \neq v$ *neither contain any fair execution nor any pair* ρ, ρ' *of unfair executions.*

An illustration of our limit-based characterizations is provided by Figure 4. Note carefully that, in the uniform case, a fair/unfair execution ρ where some process p becomes disobedient in round t implies that the same happens in all $\alpha \in B_{2^{-t}}(\rho) \cap \Sigma_v$ and $\beta \in B_{2^{-t}}(\rho) \cap \Sigma_w$. On the other hand, if p does not become disobedient in ρ, it may still be the case that p becomes disobedient in every α_k in the sequence converging to ρ, at some time t_k with $\lim_{k\to\infty} t_k = \infty$. In the non-uniform case, neither of these possibilities exists: p cannot be disobedient in the limit ρ, and any α_k where p is not obedient is also excluded as its distance to any other sequence is 1.

7 CONSENSUS CHARACTERIZATION IN TERMS OF BROADCASTABILITY

We will now develop another characterization of consensus solvability, with rests on the broadcastability of the decision sets $\Sigma_v \subseteq \Sigma$ and their connected components $\Sigma_\gamma \subseteq \Sigma_v$. It will explain topologically why the existence of a broadcaster is mandatory for solving the "standard version" of consensus, where any assignment of inputs from $\mathcal{V}$ is permitted. We start with some definitions needed for formalizing this condition:

Definition 7.1 (Heard-of sets). For every process $p \in \Pi$, there is a function $HO_p : \mathcal{C} \to 2^\Pi$ that maps a configuration $C \in \mathcal{C}$ to the set of processes $HO_p(C)$ that p has (transitively) heard of in C. Its extension to execution $\gamma = (C^t)_{t\geq 0}$ is defined as $HO_p(\gamma) = \bigcup_{t\geq 0} HO_p(C^t)$.

Heard-of sets have the following obvious properties: For executions $\gamma = (C^t)_{t\geq 0}$, $\delta = (D^t)_{t\geq 0}$ and all $t \geq 0$,

(i) $p \in HO_p(C^t)$, and $HO_p(C^t) = HO_p(D^t)$ if $C^t \sim_p D^t$,
(ii) $HO_p(C^t) \subseteq HO_p(C^{t+1})$,
(iii) for all $x \in \Pi$, if $x \in HO_q(C^t) \cap HO_q(D^t)$ and $C^t \sim_q D^t$, then $I_x(\gamma) = I_x(\delta)$ (where $I_p(\gamma)$ denotes the initial value of process p in execution γ).

The independent arbitrary input assignment condition stated in Definition 7.2 secures that, for every execution γ with initial value assignment $I(\gamma)$, there is a an isomorphic execution δ w.r.t. the HO sets of all processes that starts from an arbitrary other initial value assignment $I(\delta)$.

Definition 7.2 (Independent arbitrary input assignment condition). Let $I : \Pi \to \mathcal{V}$ be some assignment of initial values to the processes, and $\Sigma^{(I)} \subseteq \Sigma$ be the set of admissible executions with that initial value assignment. We say that Σ satisfies the *independent input assignment condition*, if and only if for any two assignments I and J, we have $\Sigma^{(I)} \cong \Sigma^{(J)}$, that is, there is a bijective mapping $f_{I,J} : \Sigma^{(I)} \to \Sigma^{(J)}$ such that for all $\gamma = (C^t)_{t\geq 0} \in \Sigma^{(I)}$ and $\delta = (D^t)_{t\geq 0} \in \Sigma^{(I)}$, writing $f_{I,J}(\gamma) = (C^t_f)_{t\geq 0}$ and $f_{I,J}(\delta) = (D^t_f)_{t\geq 0}$, the following holds for all $t \geq 0$ and all $p \in \Pi$:

(1) $Ob(C^t) = Ob(C^t_f)$
(2) $C^t \sim_p D^t$ if and only if $C^t_f \sim_p D^t_f$
(3) $HO_p(C^t) = HO_p(C^t_f)$
(4) $C^t \sim_p C^t_f$ if $I_q = J_q$ for all $q \in HO_p(C^t)$

We say that Σ satisfies the *independent arbitrary input assignment condition*, if it satisfies the independent input assignment condition for every choice of $I : \Pi \to \mathcal{V}$.

In the main results of this section (Theorem 7.12 resp. Theorem 7.13), we will not only provide a necessary and sufficient condition for solving this variant of uniform resp. non-uniform consensus based on broadcastability, but also establish the general equivalence of weak validity (V) and strong validity (SV) (recall Definition 3.1). For binary consensus, i.e., $|\mathcal{V}| = 2$, this is a well-known fact [7, Ex. 5.1], for larger input sets, it was, to the best of our knowledge, not known yet.

Since the concise but quite technical proofs of Theorem 7.12 and Theorem 7.13 somehow obfuscate the actual cause of this equivalence (and the way we actually discovered it), we first provide an alternative explanation based on the broadcastability of connected components in the following Section 7.1, which also allows us to establish some basic results needed in Section 8.4.

7.1 Broadcastability of connected components

Lemma 7.4 below reveals that if consensus (with weak validity) and independent arbitrary inputs is solvable, then every connected component of Σ needs to be broadcastable.

Definition 7.3 (Broadcastability). We call a subset $A \subseteq \Sigma$ of admissible executions *broadcastable* by the broadcaster $p \in \Pi$, if, in every execution $\gamma \in A$, every obedient process $q \in Ob(\gamma)$ eventually hears from process p, i.e., $p \in HO_q(\gamma)$, and hence knows $I_p(\gamma)$.

Lemma 7.4 (Broadcastable connected components). *A connected component Σ_γ of a set of admissible executions Σ for uniform resp. non-uniform consensus with independent arbitrary input assignments that is not broadcastable for some process contains w-valent executions for every $w \in \mathcal{V}$. In order to solve uniform resp. non-uniform consensus with independent arbitrary input assignments, every connected component must hence be broadcastable by some process, and lead to the same decision value in each of its executions.*

Proof. To prove the first part of our lemma, we consider the finite sequence of executions $\gamma = \alpha_0, \alpha_1, \ldots, \alpha_n = \gamma_w$ obtained from γ by changing the initial values of the processes $1, \ldots, n$ in $I(\gamma)$ to an arbitrary but fixed w, one by one (it is here where we need the arbitrary input assignment assumption). We show by induction that $\alpha_p \in \Sigma_\gamma$ for every $p \in \{0, \ldots, n\}$, which proves our claim since $\alpha_n = \gamma_w$.

The induction basis $p = 0$ is trivial, so suppose $\alpha_{p-1} \in \Sigma_\gamma$ according to the induction hypothesis. If it happens that $I_p(\alpha_{p-1}) = I_p(\gamma) = w$ already, nothing needs to be done and we just set $\alpha_p = \alpha_{p-1} \in \Sigma_\gamma$. Otherwise, α_p is α_{p-1} with the initial value $I_p(\alpha_p)$ changed to w. Now suppose for a contradiction that $\alpha_p \in \Sigma_{\alpha_p} \neq \Sigma_\gamma$.

Since Σ_γ is not broadcastable by any process, hence also not by p, there is some execution $\eta \in \Sigma_\gamma$ with $\eta = (C^t)_{t\geq 0}$ and a process $q \neq p$ with $q \in Ob(C^t)$ and the initial value $I_p(\eta)$ not in q's view $V_q(C^t)$ for every $t \geq 0$. Thanks to the independent input assignment property Definition 7.2, there is also an execution $\delta = f_{I(\eta),I'}(\eta) \in \Sigma_{\alpha_p}$ that matches η, i.e., is the same as η except that $I(\delta) = I'$ with $I'_q = I_q(\eta)$ for $p \neq q \in \Pi$ but possibly $I'_p \neq I_p(\eta)$. It follows that $d_q(\eta, \delta) = 0$ with $q \in Ob(\eta) \cap Ob(\delta)$ and hence $d_{\mathrm{u}}(\eta, \delta) = 0$ resp. $d_{\mathrm{nu}}(\eta, \delta) = 0$. Consequently, $\delta \in \Sigma_\gamma$ and hence $\Sigma_{\alpha_p} = \Sigma_\gamma$, which provides the required contradiction and completes the induction step.

For the second part of our lemma, assume for a contradiction that there is a non-broadcastable connected component Σ_γ in the decision set Σ_v containing all the v-valent executions γ_v. By our previous result, it would also contain some w-valent execution γ_w, $w \neq v$. Consequently, $\Sigma_v \cap \Sigma_w \neq \emptyset$, which makes consensus impossibly by Theorem 5.2 resp. Theorem 5.3. That every $\delta \in \Sigma_\gamma$ leads to the same decision value $\Delta(\delta) = \Delta(\gamma)$ follows from the continuity of the decision function and the connectedness of Σ_γ. □

In addition, Lemma 7.6 below reveals that *any* connected broadcastable set has a diameter strictly smaller than 1.

Definition 7.5 (Diameter of a set). For $A \subseteq \mathcal{C}^\omega$, depending on the distance function d that induces the appropriate topology, define A's diameter as $d(A) = \sup\{d(\gamma, \delta) \mid \gamma, \delta \in A\}$.

Lemma 7.6 (Diameter of broadcastable connected sets). *If a connected set $A \subseteq \Sigma$ of admissible executions is broadcastable by some process p, then $d_{\mathrm{u}}(A) \leq d_p(A) \leq 1/2$, as well as $d_{\mathrm{nu}}(A) \leq 1/2$, i.e., p's initial value satisfies $I_p(\gamma) = I_p(\delta)$ for all $\gamma, \delta \in A$.*

Proof. Our proof below for $d_p(A) \leq 1/2$ translates literally to any $d \in \{d_p, d_{\mathrm{u}}, d_{\mathrm{nu}}\}$; the statement $d_{\mathrm{u}}(A) \leq d_p(A)$ follows from the definition in Section 4.2.

Broadcastability by p implies that, for any $\gamma \in A$ with $\gamma = (C^t)_{t\geq 0}$, every process q has $I_p(\gamma)$ in its local view $V_q(C^{T(\gamma)})$ for some $0 < T(\gamma) < \infty$ or is not obedient any more. Abbreviating $t = T(\gamma)$, consider any $\delta \in B_{2^{-t}}(\gamma) \cap A$ with $\delta = (D^t)_{t\geq 0}$. By definition of $B_{2^{-t}}(\gamma)$, there must be some process $q \in Ob(D^t) \cap Ob(C^t)$ with $V_q(D^t) = V_q(C^t)$. Definition 7.1.(iii) thus guarantees $I_p(\delta) = I_p(\gamma)$.

We show now that this argument can be continued to reach every $\delta \in A$. For a contradiction, suppose that this is not the case and let $U(\gamma)$ be the union of the balls recursively defined as follows: $U_0(\gamma) = \{\gamma\}$, for $m > 0$, $U_m(\gamma) = \bigcup_{\delta \in U_{m-1}(\gamma)} (B_{2^{-T(\delta)}}(\delta) \cap A)$, and finally $U(\gamma) = \bigcup_{m\geq 0} U_m(\gamma)$. As a union of open balls intersected with A, which are all open in A, both $U_m(\gamma)$ for every $m > 0$ and $U(\gamma)$ is hence open in A. For every $\delta \in A \setminus U(\gamma)$, $U(\delta)$ is also open in A, and so is $V(\gamma) = \bigcup_{\delta \in A\setminus U(\gamma)} U(\delta)$. However, the open sets $U(\gamma)$ and $V(\gamma)$ must satisfy $U(\gamma) \cap V(\gamma) = \emptyset$ (as they would be the same otherwise) and $U(\gamma) \cup V(\gamma) = A$, hence A cannot be connected. □

Together, Lemma 7.4 and Lemma 7.6 imply:

Corollary 7.7 (Broadcastable Σ_γ). *If uniform resp. non-uniform consensus with independent arbitrary input assignments is solvable, then every connected component $\Sigma_\gamma \subseteq \Sigma$ must be broadcastable by some process p. In every execution $\gamma' \in \Sigma_\gamma$, the broadcaster p has the same initial value $I_p(\gamma')$, and the decision value is the same $\Delta(\gamma') = \Delta(\gamma)$.*

To emphasize the key role of the consequences of Corollary 7.7 for the equivalence of weak validity (V) and strong validity (SV), where in (SV) the consensus decision value must be the initial value of some process, we first observe that the transition from (V) to

(SV) in our Theorem 5.2 resp. Theorem 5.3 just requires the replacement of condition 2., i.e., *"If execution $\gamma \in \Sigma$ is v-valent, then $\gamma \in \Sigma_v$"*, by *"If execution $\gamma \in \Sigma_v$, then there is a process p with initial initial value $I_p(\gamma) = v$"*. This change would result in strong versions of our theorems, since the above modification is in fact transparent for the proofs of Theorem 5.2 and Theorem 5.3. Note also that both versions are equivalent for v-valent executions. Similarly, to obtain a strong version of our meta-procedure, step (3) *"Add any remaining connected component of Σ to an arbitrarily chosen set Σ_v"* must be replaced by *"Add every remaining connected component $\Sigma_\gamma \subseteq \Sigma$, where execution $\gamma \in \Sigma_\gamma$ is arbitrary, to any set Σ_v, where v is the initial value $I_b(\gamma) = v$ of a process b that is a broadcaster in every execution $\gamma' \in \Sigma_\gamma$"*.

The crucial role of Corollary 7.7 is that it makes this modification *always* possible also in the case of multi-valued consensus (in the case of binary consensus, it is obvious), as it reveals that if weak consensus is solvable, then every connected component Σ_γ must have at least one common broadcaster $b = b(\gamma') = b(\Sigma_\gamma)$ that has the same initial value $I_b(\gamma') = I_b(\gamma) = I_b(\Sigma_\gamma)$ in all executions $\gamma' \in \Sigma_\gamma$. Consequently, if decision sets resp. a meta-procedure exists that allows to solve consensus with weak validity according to Theorem 5.2 and Theorem 5.3, one can always reshuffle the connected components to form *strong* decision sets, which use the initial value of some broadcaster for assigning a connected component to a decision set:

Definition 7.8 (Strong decision sets). Let Σ be the set of admissible executions of any (weak or strong) consensus algorithm with independent arbitrary input assignments. A *strong broadcaster decision set* Σ_v^p for broadcaster $p \in \Pi$ and decision value $v \in \mathcal{V}$ resp. a *strong decision set* Σ_v for $v \in \mathcal{V}$ satisfies

$$\Sigma_v^p = \bigcup_{\substack{\gamma \in \Sigma \\ b(\Sigma_\gamma)=p \\ I_p(\gamma)=v}} \Sigma_\gamma \qquad \text{resp.} \qquad \Sigma_v = \bigcup_{p \in \Pi} \Sigma_v^p.$$

Note that strong decision sets need not be unique, as some connected component Σ_γ might have several broadcasters, any of which could be used for determining its decision value v. The canonical choice to make it uniquely defined is to take the lexically smallest $p = b(\Sigma_\gamma)$ among all broadcasters $p' \geq p$ in Σ_γ. In the rest of our paper, all strong decision sets will be canonical.

Since the canonical strong decision sets that can be formed via the abovementioned reshuffling are easily shown to satisfy the strong versions of Theorem 5.2 and Theorem 5.3, one obtains the broadcast-based characterization of consensus stated in Theorem 7.9. Rather than proving it by formalizing the reasoning sketched above, however, we will rely on the general equivalence results Theorem 7.12 resp. Theorem 7.13 developed in in the following Section 7.2. This way, the somewhat tedious and non-constructive reshuffling of connected components involved in the direct proof can be replaced by an explicit construction of the canonical strong decision sets, which utilizes binary consensus.

Theorem 7.9 (Consensus characterization via broadcastability). *A model allows to solve uniform resp. non-uniform consensus with independent arbitrary input assignments if and only if it guarantees that (i) every connected component Σ_γ of the set Σ of admissible executions is broadcastable for some process $p = b(\Sigma_\gamma)$ starting with the same input $I_p(\gamma')$ in $\gamma' \in \Sigma_\gamma$, and (ii) that the strong broadcaster decision sets Σ_v^p, $p \in \Pi$, $v \in \mathcal{V}$, as specified in Definition 7.8, are clopen in Σ in the uniform topology resp. the non-uniform topology.*

Proof. The broadcastability of the connected components follows from Corollary 7.7, the clopenness of the strong broadcaster decision sets will be established in the proofs of Theorem 7.12 resp. Theorem 7.13. □

7.2 General broadcast-based characterization

We will now provide our general broadcast-based characterization for uniform and non-uniform consensus with arbitrary and independent input assignments according to Definition 7.2. In a nutshell, it uses a reduction to (the solvability of) binary consensus, where weak and strong validity are trivially equivalent, for explicitly constructing the canonical strong broadcaster decision sets.

Let $\hat{\Sigma} \subseteq \Sigma$ denote the set of admissible executions of a multi-valued consensus algorithm starting from a single initial value assignment $\hat{I} : \Pi \to \mathcal{V}$ (any will do, the choice is arbitrary).

Definition 7.10 (Uniform/non-uniform broadcastability). We say that $\hat{\Sigma}$ is *uniformly resp. non-uniformly broadcastable* if there exist sets $\hat{\Sigma}_p \subseteq \hat{\Sigma}$ for $p \in \Pi$ such that:

(1) The sets $\hat{\Sigma}_p$ are pairwise disjoint and $\bigcup_{p\in\Pi} \hat{\Sigma}_p = \hat{\Sigma}$.
(2) Every $\hat{\Sigma}_p$ is d_{u}-clopen resp. d_{nu}-clopen in $\hat{\Sigma}$.
(3) Every $\hat{\Sigma}_p$ is broadcastable by p but not by any lexically smaller $p' < p$, i.e., every obedient process $q \in Ob(\gamma)$ satisfies $p \in HO_q(\gamma)$ for every $\gamma \in \hat{\Sigma}_p$.

Theorem 7.11. *If uniform resp. non-uniform binary consensus with arbitrary and independent input assignments is solvable, then $\hat{\Sigma}$ is uniformly resp. non-uniformly broadcastable.*

Proof. By Theorem 5.2 resp. Theorem 5.3 restricted to $|\mathcal{V}| = 2$, there exists a clopen partition (Σ_0, Σ_1) of Σ such that Σ_0 includes all 0-valent executions and Σ_1 includes all 1-valent executions.

For $0 \le p \le n$, let I_p be the initial value assignment in which all processes $q \le p$ have initial value 1 and all processes $q > p$ have initial value 0. The assignment I_0 is the all-0 assignment and I_n is the all-1 assignment. According to Definition 7.2, there is an isomorphism $g_p = f_{I_{p-1},I_p} : \Sigma^{(I_{p-1})} \to \Sigma^{(I_p)}$ for every $1 \le p \le n$, as well as an isomorphism $h = f_{I_n,\hat{I}} : \Sigma^{(I_n)} \to \hat{\Sigma}$.

We now inductively define the set $\Sigma_{p,q}$, for $1 \le p \le n$ and $1 \le q \le p$, which consists of all 1-deciding executions starting from I_p where q is the lexically smallest broadcaster. Note that both p and p' might be broadcasters in $\gamma \in \Sigma_{p,q}$, provided $p < p'$.

(i) $\Sigma_{1,1} = \Sigma_1^{(I_1)}$, the set of 1-deciding executions when starting with initial value assignment I_1.
(ii) For $2 \le p \le n$ and $1 \le q \le p-1$, we set $\Sigma_{p,q} = g_p[\Sigma_{p-1,q}]$.
(iii) For $2 \le p \le n$ and $q = p$, we set $\Sigma_{p,p} = \Sigma_1^{(I_p)} \setminus \bigcup_{q=1}^{p-1} \Sigma_{p,q}$.

A trivial induction reveals that, for every $1 \le p \le n$, $\Sigma_{p,q} \subseteq \Sigma^{(I_p)}$, and that the sets $\Sigma_{p,q}$ are pairwise disjoint since all the g_p are bijective. Furthermore, since the decision sets $\Sigma_1^{(I_p)}$ are clopen in $\Sigma^{(I_p)}$ and the g_p are homeomorphisms, every $\Sigma_{p,q}$ is clopen in $\Sigma^{(I_p)}$.

We now prove, by induction on p, that every $1 \le q \le p$ is the lexically smallest broadcaster in every execution $\gamma \in \Sigma_{p,q}$, i.e., that $q \in HO_r(\gamma)$ for every $r \in Ob(\gamma)$, and that there is no smaller q' with this property. We start with the base case $p = q = 1$, which is obviously the lexically smallest. Let $\gamma \in \Sigma_1^{(I_1)}$ and $r \in Ob(\gamma)$. Assuming by contradiction that $1 \notin HO_r(\gamma)$, we get $\Delta_r(\gamma) = \Delta_r(g_1^{-1}(\gamma)) = 0$ by Definition 7.2.(4) and Validity (V). This contradicts $\gamma \in \Sigma_1^{(I_1)}$, however. Now let $2 \le p \le n$. For all $1 \le q \le p-1$ and all $\gamma \in \Sigma_{p,q}$, we

have that $q \in HO_r(\gamma) = HO_r(g_q^{-1}(\gamma))$ for all $r \in Ob(\gamma) = Ob(g_q^{-1}(\gamma))$ is the lexically smallest broadcaster by the induction hypothesis. For $q = p$, assuming by contradiction that $p \notin HO_r(\gamma)$ for $\gamma \in \Sigma_{p,p}$ and $r \in Ob(\gamma)$, we get $\Delta_r(\gamma) = \Delta_r(g_p^{-1}(\gamma)) = 0$ by Definition 7.2.(4) and the fact that (iii) implies $\Sigma_1^{(I_{p-1})} \subseteq \bigcup_{q=1}^{p-1} \Sigma_{p-1,q}$ since $\Sigma_{p-1,p-1} = \Sigma_1^{(I_{p-1})} \setminus \bigcup_{q=1}^{p-2} \Sigma_{p-1,q}$; the latter also guarantees that there is no lexically smaller broadcaster. This completes our induction proof.

We finally set $\hat{\Sigma}_p = h[\Sigma_{n,p}]$ for $1 \leq p \leq n$ and show that the result satisfies uniform broadcastability according to Definition 7.10: (1) Pairwise disjointness of the $\hat{\Sigma}_p$ follows from pairwise disjointness of the $\Sigma_{n,p}$. The fact that $\bigcup_{p=1}^{n} \hat{\Sigma}_p = \hat{\Sigma}$ follows from the definition of $\Sigma_{n,n}$ and the fact that $\Sigma_1^{(I_n)} = \Sigma^{(I_n)}$ by Validity. (2) Clopenness of the $\hat{\Sigma}_p$ follows from clopenness of the $\Sigma_{n,p}$ and the fact that h is a homeomorphism. (3) For every $\gamma \in \hat{\Sigma}_p$ and $q \in Ob(\gamma)$, we have $p \in HO_q(\gamma) = HO_q(h^{-1}(\gamma))$. This concludes the proof. □

With this result, we can prove the following equivalences Theorem 7.12 resp. Theorem 7.13 for uniform and non-uniform consensus:

Theorem 7.12. *For a set of admissible executions Σ where uniform consensus with arbitrary and independent input assignments is solvable, the following statements are equivalent:*

(1) *Uniform binary consensus is solvable.*
(2) *Foy any input assignment $\hat{I} : \Pi \to \mathcal{V}$, the subset of admissible executions $\hat{\Sigma} \subseteq \Sigma$ using $\hat{I}$ is uniformly broadcastable.*
(3) *Strong uniform consensus is solvable for any set $\mathcal{V}$ of initial values.*
(4) *Weak uniform consensus is solvable for any set $\mathcal{V}$ of initial values.*

Proof. The implications (3)⇒(4)⇒(1) are trivial. The implication (1)⇒(2) follows from Theorem 7.11. To prove the implication (2)⇒(3), we give an algorithm that solves strong consensus, akin to those used in the proofs of Theorem 5.2.

Let $\hat{\Sigma}$ be broadcastable and let $\hat{\Sigma}_p$ be sets as in the definition of broadcastability. For every initial value assignment $I : \Pi \to \mathcal{V}$, let $g_I = f_{\hat{I},I} : \hat{\Sigma} \to \Sigma^{(I)}$ be the corresponding isomorphism. For $p \in \Pi$ and $v \in \mathcal{V}$, we define the canonical strong broadcaster decision sets

$$\Sigma_v^p = \bigcup_{\substack{I:\Pi\to\mathcal{V} \\ I_p=v}} g_I[\hat{\Sigma}_p] \qquad \text{and} \qquad \Sigma_v = \bigcup_{p\in\Pi} \Sigma_v^p \ .$$

The sets Σ_v^p are d_{u}-open in Σ: For any $\gamma \in \Sigma_v^p$, let T be a time at which, (i) in execution γ, all processes have heard from p and (ii) $B_{2^{-T}}(g_I^{-1}(\gamma)) \subseteq \hat{\Sigma}_p$ in $\hat{\Sigma}$ for all $I : \Pi \to \mathcal{V}$ with $I_p = v$, and choose the neighborhood

$$\begin{aligned}
\mathcal{N} &= \left\{\delta \in \Sigma \mid d_{\mathrm{u}}(\gamma,\delta) < 2^{-T}\right\} \\
&= \left\{\delta \in \Sigma \mid \exists q \in \Pi\colon C^T \sim_q D^T\right\} \\
&= \left\{\delta \in \Sigma \mid \exists q \in \Pi\colon C^T \sim_q D^T \wedge p \in HO_q(C^T) = HO_q(D^T)\right\} \\
&\subseteq \{\delta \in \Sigma \mid I_p(\gamma) = I_p(\delta) = v\} \subseteq \bigcup_{\substack{I:\Pi\to\mathcal{V} \\ I_p=v}} g_I[\hat{\Sigma}]
\end{aligned}$$

where we use the notation $\gamma = (C^t)_{t\geq 0}$ and $\delta = (D^t)_{t\geq 0}$. By assumption (ii) on the choice of T, for every $I : \Pi \to \mathcal{V}$ with $I_p = v$, we have

$$\begin{aligned}\mathcal{N} \cap g_I[\hat{\Sigma}] &= \left\{\delta \in g_I[\hat{\Sigma}] \mid d_{\mathrm{u}}(\gamma,\delta) < 2^{-T}\right\}\\ &= \left\{\delta \in g_I[\hat{\Sigma}] \mid d_{\mathrm{u}}\left(g_I^{-1}(\gamma), g_I^{-1}(\delta)\right) < 2^{-T}\right\}\\ &= \left\{g_I(\delta) \mid \delta \in \hat{\Sigma} \wedge d_{\mathrm{u}}\left(g_I^{-1}(\gamma), \delta\right) < 2^{-T}\right\}\\ &\subseteq \left\{g_I(\delta) \mid \delta \in \hat{\Sigma}_p\right\} = g_I[\hat{\Sigma}_p]\ .\end{aligned}$$

Combining the last two equations, we get

$$\mathcal{N} = \bigcup_{\substack{I:\Pi\to\mathcal{V}\\ I_p=v}} \left(\mathcal{N} \cap g_I[\hat{\Sigma}]\right) \subseteq \bigcup_{\substack{I:\Pi\to\mathcal{V}\\ I_p=v}} g_I[\hat{\Sigma}_p] = \Sigma_v^p\ .$$

The sets Σ_v^p, as well as the sets Σ_v as unions of the Σ_v^p, are thus d_{u}-open in Σ. The Σ_v are pairwise disjoint since the $\hat{\Sigma}_p$ are. We further have $\Sigma = \bigcup_{v\in\mathcal{V}} \Sigma_v$.

We now define the strong consensus algorithm. For every configuration $C \in \mathcal{C}$, we set

$$\Delta_q(C) = \begin{cases} v & \text{if } \{\delta \in \Sigma \mid \exists t\colon C \sim_q D^t\} \subseteq \Sigma_v\\ \bot & \text{otherwise}\end{cases}$$

The function Δ_q is well-defined since the sets Σ_v are pairwise disjoint.

We first show Termination. Let $\gamma \in \Sigma$, let $I : \Pi \to \mathcal{V}$ be the initial value assignment of γ, and let $q \in Ob(\gamma)$. Since Σ_v is d_{u}-open in Σ, there exists some $\varepsilon > 0$ such that $\{\delta \in \Sigma \mid d_q(\gamma,\delta) < \varepsilon\} = \{\delta \in \Sigma \mid d_{\mathrm{u}}(\gamma,\delta) < \varepsilon\} \subseteq \Sigma_v$. Letting T be the smallest integer such that $2^{-\chi_q(C^t)} \leq \varepsilon$ for all $t \geq T$, we get $\Delta_q(C^t) = v$ for all $t \geq T$, just like in the proof of Theorem 5.2.

To show Uniform Agreement, assume by contradiction that process q decides a value $w \neq v$ in configuration C in execution $\gamma \in \Sigma_v$. Then, by definition of Δ_q, we have $\gamma \in \{\delta \in \Sigma \mid \exists t\colon C \sim_q D^t\} \subseteq \Sigma_w$. But this is impossible since $\Sigma_v \cap \Sigma_w = \emptyset$.

We finish the proof by showing Strong Validity. Let $\gamma \in \Sigma_v$. Then, by definition, there exists a $p \in \Pi$ and an $I : \Pi \to \mathcal{V}$ with $I_p = v$ such that $\gamma \in g_I[\hat{\Sigma}_p] \subseteq \Sigma^{(I)}$. But then, in particular, $I_p(\gamma) = I_p = v$. □

Theorem 7.13. *For a set of admissible executions Σ where non-uniform consensus with arbitrary and independent input assignments is solvable, the following statements are equivalent:*

(1) *Non-uniform binary consensus is solvable.*
(2) *Foy any input assignment $\hat{I} : \Pi \to \mathcal{V}$, the subset of admissible executions $\hat{\Sigma} \subseteq \Sigma$ using $\hat{I}$ is uniformly broadcastable.*
(3) *Strong non-uniform consensus is solvable for any set $\mathcal{V}$ of initial values.*
(4) *Weak non-uniform consensus is solvable for any set $\mathcal{V}$ of initial values.*

Proof. The proof is similar to that of Theorem 7.12.

The implications (3)⇒(4)⇒(1) are trivial. The implication (1)⇒(2) follows from Theorem 7.11. To prove the implication (2)⇒(3), we give an algorithm that solves strong consensus, akin to those used in the proofs of Theorem 5.3.

Let $\hat{\Sigma}$ be broadcastable and let $\hat{\Sigma}_p$ be sets as in the definition of broadcastability. For an initial value assignment $I : \Pi \to \mathcal{V}$, let $g_I = f_{\hat{I},I} : \hat{\Sigma} \to \Sigma^{(I)}$ be the isomorphism. For $p \in \Pi$ and $v \in \mathcal{V}$, we define the canonical strong broadcaster decision sets

$$\Sigma_v^p = \bigcup_{\substack{I:\Pi\to\mathcal{V}\\ I_p=v}} g_I[\hat{\Sigma}_p] \qquad \text{and} \qquad \Sigma_v = \bigcup_{p\in\Pi} \Sigma_v^p \ .$$

The sets Σ_v^p are d_{nu}-open in Σ: For any $\gamma \in \Sigma_v^p$, let T be a time at which, (i) in execution γ, all processes have heard from p and (ii) $B_{2^{-T}}(g_I^{-1}(\gamma)) \subseteq \hat{\Sigma}_p$ in $\hat{\Sigma}$ for all $I : \Pi \to \mathcal{V}$ with $I_p = v$, and choose the neighborhood

$$\begin{aligned}
\mathcal{N} &= \left\{\delta \in \Sigma \mid d_{\mathrm{nu}}(\gamma,\delta) < 2^{-T}\right\} \\
&= \left\{\delta \in \Sigma \mid \exists q \in \Pi\colon C^T \sim_q D^T \wedge q \in Ob(\gamma) \cap Ob(\delta)\right\} \\
&\subseteq \left\{\delta \in \Sigma \mid \exists q \in \Pi\colon C^T \sim_q D^T \wedge p \in HO_q(C^T) = HO_q(D^T)\right\} \\
&\subseteq \{\delta \in \Sigma \mid I_p(\gamma) = I_p(\delta) = v\} \subseteq \bigcup_{\substack{I:\Pi\to\mathcal{V}\\ I_p=v}} g_I[\hat{\Sigma}]
\end{aligned}$$

where we use the notation $\gamma = (C^t)_{t\geq 0}$ and $\delta = (D^t)_{t\geq 0}$. By assumption (ii) on the choice of T, for every $I : \Pi \to \mathcal{V}$ with $I_p = v$, we have

$$\begin{aligned}
\mathcal{N} \cap g_I[\hat{\Sigma}] &= \left\{\delta \in g_I[\hat{\Sigma}] \mid d_{\mathrm{nu}}(\gamma,\delta) < 2^{-T}\right\} \\
&= \left\{\delta \in g_I[\hat{\Sigma}] \mid d_{\mathrm{nu}}\left(g_I^{-1}(\gamma), g_I^{-1}(\delta)\right) < 2^{-T}\right\} \\
&= \left\{g_I(\delta) \mid \delta \in \hat{\Sigma} \wedge d_{\mathrm{nu}}\left(g_I^{-1}(\gamma), \delta\right) < 2^{-T}\right\} \\
&\subseteq \left\{g_I(\delta) \mid \delta \in \hat{\Sigma}_p\right\} = g_I[\hat{\Sigma}_p] \ .
\end{aligned}$$

Combining the last two equations, we get

$$\mathcal{N} = \bigcup_{\substack{I:\Pi\to\mathcal{V}\\ I_p=v}} \left(\mathcal{N} \cap g_I[\hat{\Sigma}]\right) \subseteq \bigcup_{\substack{I:\Pi\to\mathcal{V}\\ I_p=v}} g_I[\hat{\Sigma}_p] = \Sigma_v^p \ .$$

The sets Σ_v^p, as well as the sets Σ_v as unions of the Σ_v^p, are thus d_{nu}-open in Σ. The Σ_v are pairwise disjoint since the $\hat{\Sigma}_p$ are. We further have $\Sigma = \bigcup_{v\in\mathcal{V}} \Sigma_v$.

We now define the strong consensus algorithm. For every configuration $C \in \mathcal{C}$, we set

$$\Delta_q(C) = \begin{cases} v & \text{if } \{\delta \in \Sigma \mid \exists t\colon C \sim_q D^t \wedge q \in Ob(\delta)\} \subseteq \Sigma_v \\ \bot & \text{otherwise} \end{cases}$$

The function Δ_q is well-defined since the sets Σ_v are pairwise disjoint.

We first show Termination. Let $\gamma \in \Sigma$, let $I : \Pi \to \mathcal{V}$ be the initial value assignment of γ, and let $q \in Ob(\gamma)$. Since Σ_v is d_{nu}-open in Σ, there exists some $\varepsilon > 0$ such that $\{\delta \in \Sigma \mid d_q(\gamma,\delta) < \varepsilon \wedge q \in Ob(\delta)\} = \{\delta \in \Sigma \mid d_{\mathrm{nu}}(\gamma,\delta) < \varepsilon\} \subseteq \Sigma_v$. Letting T be the smallest integer such that $2^{-\chi_p(C^t)} \leq \varepsilon$ for all $t \geq T$, we get $\Delta_p(C^t) = v$ for all $t \geq T$.

To show Agreement, assume by contradiction that process q decides a value $w \neq v$ in configuration C in execution $\gamma \in \Sigma_v$. Then, by definition of Δ_q, we have $\gamma \in \{\delta \in \Sigma \mid \exists t\colon C \sim_q D^t \wedge q \in Ob(\delta)\} \subseteq \Sigma_w$. But this is impossible since $\Sigma_v \cap \Sigma_w = \emptyset$.

We finish the proof by showing Strong Validity. Let $\gamma \in \Sigma_v$. Then, by definition, there exists a $p \in \Pi$ and an $I : \Pi \to \mathcal{V}$ with $I_p = v$ such that $\gamma \in g_I[\hat{\Sigma}_p] \subseteq \Sigma^{(I)}$. But then, in particular, $I_p(\gamma) = I_p = v$. □

We conclude this section by pointing that the practical utility of the equivalence of consensus with weak and strong validity established in Theorem 7.12 and Theorem 7.13 is somewhat limited: Since the solution algorithms depend on the a priori knowledge of the decision sets, they do not give a clue on how to develop a strong consensus algorithm from a weak consensus algorithm in a given model. In fact, determining and agreeing upon a broadcaster in executions that are not v-valent is a very hard problem.

8 APPLICATIONS

In this section, we will apply our topological characterizations of consensus solvability to several different examples. Apart from providing a topological explanation of bivalence proofs (Section 8.1) and folklore results for synchronous consensus under general omission faults (Section 8.2), we will provide a novel characterization of condition-based asynchronous consensus [34] with strong validity (Section 8.3), a complete characterization of consensus solvability for dynamic networks with both closed (Section 8.4) and non-closed (Section 8.5) message adversaries, and a consensus algorithm for asynchronous systems with weak timely links that does not rely on an implementation of the Ω failure detector (Section 8.6).

8.1 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 [19], which have been generalized [33] and used in many different contexts: Our results reveal that the forever bivalent executions constructed inductively in bivalence proofs [10, 44, 45, 49] are just the common limit of two infinite sequence of executions $\alpha_0, \alpha_1, \dots$ in the 0-decision set Σ_0 and $\beta_0, \beta_1, \dots$ in the 1-decision set Σ_1.

More specifically, what is common to these proofs is that one shows that, for any consensus algorithm, there is an admissible forever bivalent execution γ. 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 α_t with decision value 0 and β_t with decision value 1 starting out from C^t, i.e., having a common prefix that leads to C^t. Consequently, their distance satisfies $d_{\mathrm{nu}}(\alpha_t, \gamma) < 2^{-t}$ and $d_{\mathrm{nu}}(\beta_t, \gamma) < 2^{-t}$. But then closedness of Σ_0 and Σ_1 implies that $\gamma \in \Sigma_0 \cap \Sigma_1$, a contradiction to their disjointness.

By construction, the $(t-1)$-prefixes of α_t and α_{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 4 for an illustration. Therefore, these executions match Definition 6.6, and Corollary 6.7 implies that the stipulated consensus algorithm cannot be correct. A specific example is the lossy-link impossibility [44], 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 [10, 49]. In the case of the oblivious lossy-link message adversary using the reduced set $\{\leftarrow, \rightarrow\}$ considered by Coulouma, Godard, and Peters [14], consensus is solvable and there is no forever bivalent execution. Indeed, there exists a consensus algorithm where all configurations reached after the first round are already univalent, see Section 8.4.

8.2 Consensus in synchronous systems with general omission process faults

As a more elaborate example of systems where the solvability of non-uniform and uniform consensus may be different (which also cover the simple running examples used in Section 4), we take synchronous systems with up to f general omission process faults [40]. For $n \geq f+1$, non-uniform consensus can be solved in $f+1$ rounds, whereas solving uniform consensus requires $n \geq 2f+1$.

The impossibility proof of uniform consensus for $n \leq 2f$ uses a standard partitioning argument, splitting Π into a set P of processes with $|P| = f$ and Q with $|Q| = n - f \leq f$. One considers an admissible execution α_0 where all processes $p \in \Pi$ start with $I_p = 0$, the ones in P are correct, and the ones in Q are initially mute; the decision value of the processes in P must be 0 by validity. Similarly, α_1 starts from $I_p = 1$, all processes in Q are correct and the ones in P are initially mute; the decision value is hence 1. For another execution α, where the processes in Q are correct and the ones in P are general omission faulty, in the sense that every $p \in P$ does not send and receive any message to/from Q, one observes $\alpha \sim_p \alpha_0$, i.e., $d_p(\alpha, \alpha_0) < 2^{-t}$ for all $t \geq 0$ and all $p \in P$. Similarly, $\alpha \sim_q \alpha_1$ for every $q \in Q$. Hence, p and q decide on different values in α.

Topologically, this is equivalent to $d_{\mathrm{u}}(\alpha, \alpha_0) = 0$ as well as $d_{\mathrm{u}}(\alpha, \alpha_1) = 0$, which implies $\alpha \in \Sigma_0$ as well as $\alpha \in \Sigma_1$. Consequently, Σ_0 and Σ_1 cannot be disjoint, as needed for uniform consensus solvability. Clearly, for $n \geq 2f+1$, this argument is no longer applicable. And indeed, algorithms like the one proposed by Parvedy and Raynal [39] can be used for solving uniform consensus.

If one revisits the topological equivalent of the above partitioning argument for $n \leq 2f$ in the *non-uniform* case, it turns out that still $d_{\mathrm{nu}}(\alpha, \alpha_0) = 0$, but $d_{\mathrm{nu}}(\alpha, \alpha_1) = 1$ as all processes in Q are faulty. Consequently, $\alpha \notin \Sigma_1$. So Σ_0 and Σ_1 could partition the space of admissible executions. And indeed, non-uniform consensus can be solved in $f+1$ rounds here. In order to demonstrate this by means of our Theorem 5.3, we will sketch how the required decision sets Σ_v can be constructed. We will do so by means of a simple *labeling algorithm*, which assigns a decision value $v \in \mathcal{V}$ to every admissible execution γ. Note that synchronous systems are particularly easy to model in our setting, since we can use the number of rounds as our global time t.

Clearly, every process that omits to send its state in some round to a (still) correct processor is revealed to every other (still) correct processor at the next round at the latest. This implies that every correct process p seen by *some* correct process q by the end of the $(f+1)$-round prefix $\gamma|_{f+1}$ in the admissible execution γ has also been seen by every other correct process during $\gamma|_{f+1}$ as well, since one would need a chain of $f+1$ *different* faulty processes for propagating p's state to q otherwise. Thus, p must have managed to broadcast its initial value $I_p(\gamma)$ to all correct processes during $\gamma|_{f+1}$.

Consequently, if $\gamma|_{f+1} \sim \rho|_{f+1}$, where $\sim$ denotes the transitive closure (over all processes $p \in \Pi$) of the indistinguishability relation $\sim_p$ for prefixes, they must have the same set of broadcasters. Our labeling algorithm hence just assigns to γ the initial value I_p of the, say, lexically smallest broadcaster p in $\gamma|_{f+1}$. The resulting decision sets are trivially open since, for every $\gamma \in \Sigma_v$, we have $B_{2^{-(f+1)}}(\gamma) \subseteq \Sigma_v$ as well. The generic non-uniform consensus algorithm from Theorem 5.3 resp. Theorem 7.12 can hence be used for solving weak resp. strong consensus.

8.3 Asynchronous condition-based consensus

As an example of asynchronous consensus in shared-memory systems, we consider the condition-based approach presented by Mostefaoui, Rajsbaum, and Raynal [34]. In order to circumvent the FLP impossibility [19] of consensus in the presence of process crashes, the authors considered restrictions of the vectors of allowed initial values $I(\gamma)$ for the admissible executions $\gamma \in \Sigma$ of the n processes in the system. To ensure compatibility with the notation used in the original paper [34], we will write $I[1], \ldots, I[n]$ instead of $I_1, \ldots, I_n$ for the initial value assignment of a given admissible execution in this section. For a set $C \subseteq \mathcal{V}^n$ of allowed input vectors (called a *condition*) that is a priori known to all processes, the authors asked for properties C must satisfy such that uniform consensus can be solved in the presence of up to f crashes. Note carefully that this is an instance of consensus where the *arbitrary* input assumption does not apply, albeit the independent input assumption (recall Definition 7.2) is needed.

Two such properties were identified in [34]: (i) the more practical *f-acceptability* property, which consists of "elements" that can be directly utilized in a generic solution algorithm, and (ii) the more abstract *f-legality* condition. Moreover, two different variants of consensus were considered: (a) non-safe consensus, which only needs to terminate when the initial values are indeed from C, and (b) safe consensus, where the processes must also terminate for arbitrary inputs in well-behaved (in particular, fault-free) executions. Interestingly, it turned out that (i) and (ii), as well as (a) and (b), are equivalent, and that either variant of consensus can be solved in the presence of up to f crashes if and only if C is f-legal or/and f-acceptable [34, Thm. 5.7] .

The generic non-safe solution algorithm for an f-acceptable condition C is extremely simple: It only uses one round, where process p_i first writes its initial value $I[i]$ into its entry $V[i]$ of a snapshot object V that is initialized to $V[*] = \perp$, and then performs snapshot reads that provide its current local view V_i until it finds at least $n - f$ non-$\perp$ entries in V_i. The latter condition terminates the round, at the end of which p_i uses the "elements" making up f-acceptability for computing the decision value from its final view V_i. Note that a $\perp$ entry in $V_i[j]$ can be due to a crash of p_j or just a consequence of the fact that p_j has just been slow compared to the at least $n - f$ other processes that managed to provide non-$\perp$ entries. To make this algorithm compatible with our setting, where all executions are infinite, we just add infinitely many empty rounds (where no process changes its state or reads/writes V). Moreover, we consider all processes to be obedient and just make at most f of them very slow when needed, which allows us to directly use our uniform topology.

The definition of f-legality is based on an undirected graph $H(C, f)$, whose vertices are the vectors in C and where there is an edge $(I1, I2)$ if and only if the Hamming distance between $I1 \in C$ and $I2 \in C$ is at most f. The graph $H(C, f)$ can be expanded into a graph $Gin(C, f)$ of all the views V_i possibly obtained by any process p_i in the above algorithm: For every $I \in C$, $Gin(C, f)$ contains all the vertices that are obtained by replacing up to f entries of I by $\perp$. Two vertices $J1, J2 \in Gin(C, f)$ are connected by an undirected edge if $J1[i] \neq \perp \Rightarrow J1[i] = J2[i]$ for every $1 \leq i \leq n$, or vice versa. It is not difficult to see that $I1, I2 \in H(C, f)$ are connected by an edge if and only if the same vertices $I1, I2 \in Gin(C, f)$ are connected by a path.

A condition C is f-legal if, for each connected component $G_1, \ldots, G_x$ of $Gin(C, f)$, all the vertices in the component have at least one input value v in common [34, Def. 5.2]. This property translates to the corresponding connected components $H_1, \ldots, H_x$ of $H(C, f)$ as: all vertices in a component must have at least one entry with input value v in common, and

v appears in $f+1$ entries in every vertex. In fact, without the latter, v would disappear from the view J in $Gin(C,f)$ where the at most f entries holding v in $I \in H(C,f)$ are replaced by $\perp$.

The setting for condition-based consensus in [34] differs from the one underlying our topological results in the previous sections in two aspects: (1) It uses a validity condition that is stronger than our strong validity (SV), as it does not allow processes to decide on the initial value of an initially dead process. (2) It does not allow arbitrary input assignments, which is a pivotal assumption in all our broadcasting-based characterizations in Section 7. And indeed, as it will turn out, we do not usually have a *common* broadcaster p in the connected components of a decision set Σ_v here.

In Theorem 8.1 below, we will characterize the solvability of condition-based consensus with strong validity (SV) using our topological approach. To model (SV), the original f-legality condition must be weakened to *f-quasilegality*: Rather than assuming that all input assignments I in a connected component G_i in $Gin(C,f)$, i.e., the vertices also lying in the corresponding connected component H_i in $H(C,f)$, must have a value v in common that appears in at least $f+1$ entries in I, f-quasilegality only requires a common value v.

For our proof, we exploit the very simple structure of the set of admissible executions Σ of the generic condition-based consensus algorithm, and the close relation between Σ and $Gin(C,f)$. In fact, $Gin(C,f)$ is a graph on all the possible views of the processes (at the end of the first round) in any execution. More specifically, for the admissible execution $\alpha = \alpha(I) \in \Sigma$ starting from the initial value assignment I, the configuration $\alpha^1 = (J_1, \_, \ldots, J_n)$ after round 1 satisfies $J_j \in G_i \subseteq Gin(C,f)$ for every $p_j \in \Pi$ and $J_j = \perp$ otherwise. Herein, G_i is the connected component in $Gin(C,f)$ that contains I. This holds since every J_j is obtained from I by replacing at most f entries with $\perp$ in $Gin(C,f)$. Note carefully that every process can hence unambigously identify the connected component G_i the current execution belongs to, as it only needs to check in which connected component its local view lies. Recall that it is assumed that every process knows C and hence $H(C,f)$ and $Gin(C,f)$ a priori.

Theorem 8.1 (Condition-based consensus characterization). *In the asynchronous shared memory system with at most f crash faults, condition-based consensus with strong validity (SV) can be solved for condition C if and only if C is f-quasilegal, in the sense that all the vertices in a connected component of $H(C,f)$ have a value v in common.*

Proof. We first prove that if C is f-quasilegal, then strong consensus is solvable. With $v_i \in \mathcal{V}$ denoting the common value a priori chosen for the connected component H_i (and hence G_i), we define the decision sets as $\Sigma_{v_i} = \{\gamma | \gamma^1 \in G_i\}$, where $\gamma^1 = D^1$ for $\gamma = (D^t)_{t\geq 0}$. By construction, Σ_v and Σ_w are disjoint for $w \neq v$. Since our topology is discrete, as the finiteness of $Gin(C,f)$ implies that there are only finitely many different admissible executions in Σ, all decision sets (and their connected components) are clopen in Σ. Applying the algorithm given in Theorem 7.12 hence allows to solve consensus.

On the other hand, to show that consensus cannot be solved if C is not f-quasilegal, suppose for a contradiction that there is a correct strong consensus algorithm without it. We first prove that all executions starting from an input value assignment $I \in G_i$ in a connected component $G_i \subseteq Gin(C,f)$, which necessarily also contains all the possible views of all processes in G_i, lie in the same connected component in Σ. To prove this, it suffices to show by induction that, for any two executions $\gamma = \gamma(I)$ and $\delta = \delta(I')$ with $I, I' \in G_i$, there is a finite sequence of executions $\gamma = \alpha_0, \alpha_1, \ldots, \alpha_{k+1} = \delta$ such that, for every $0 \leq j < k+1$,

$\alpha_j \in G_i$ and $\alpha_j \sim_{q_j} \alpha_{j+1}$ for some process q_j. This implies $d_{\mathrm{u}}(\alpha_j, \alpha_{j+1}) = 0$ and hence also $d_{\mathrm{u}}(\gamma, \delta) = 0$ as needed.

Since G_i is a connected component containing I, I', there must be a chain of $k \geq 2$ different initial value assignments $I_0 = I_1 = I, I_1, \ldots, I_k = I_{k+1} = I'$ in G_i where I_ℓ and $I_{\ell+1}$, $1 \leq \ell \leq k-1$, are connected by an edge in $H(C, f)$ (and hence by a path in G_i). Moreover, there must be processes $p_1, \ldots, p_{k-1}$ such that $I_\ell[p_\ell] \neq I_{\ell+1}[p_\ell]$. For the induction basis $\ell = 1$, we choose $\alpha_1 = \alpha_1(I_1)$ to be any execution where some process q_0 has the same view in α_0^1 and in α_1^1, and process q_1 has the same view J_1 in α_1^1 and in α_2^1, so $\alpha_0 \sim_{q_0} \alpha_1 \sim_{q_1} \alpha_2$. This choice of α_1 is possible, since α_0 and α_1 start from the same I, and since I_1 and I_2 have a Hamming distance between 1 and f and can hence have a common view J_1 with $\perp$ for all processes q, including q_1, where $I_1[q] \neq I_2[q]$. Note that it is here where we need the independent (but not arbitrary!) input assignment property Definition 7.2. For the induction step, assume that we have already constructed α_ℓ for $\ell \geq 1$. For $\alpha_{\ell+1}$, we choose an execution where q_ℓ has the same view J_ℓ in α_ℓ and $\alpha_{\ell+1}$ (necessarily with $J_\ell[q_\ell] = \perp$), and $q_{\ell+1}$ has the same view $J_{\ell+1}$ in $\alpha_{\ell+1}$ and $\alpha_{\ell+2}$ (necessarily with $J_{\ell+1}[q_{\ell+1}] = \perp$, unless $\ell + 1 = k$ already, in which case both $\alpha_{\ell+1}$ and $\alpha_{\ell+2}$ start from I'), which leads to $\alpha_\ell \sim_{q_\ell} \alpha_{\ell+1} \sim_{q_{\ell+1}} \alpha_{\ell+2}$ and completes our induction proof.

Since C is not f-quasilegal by assumption, there must be a connected component $G_i \subseteq Gin(C, f)$ that contains initial configurations I and I', such that I' does not contain any value present in I. In order not to violate strong validity, no executions $\gamma = \gamma(I)$ and $\delta = \delta(I')$ may lie in the same decision set. However, we have just shown that they lie in the same connected component in Σ, which provides the required contradiction. □

8.4 Dynamic networks with limit-closed message adversaries

In this section, we will consider consensus with independent and arbitrary input assignments in dynamic networks under message adversaries [2] that are *limit-closed* [47], in the sense that every convergent sequence of executions $\alpha_0, \alpha_1, \ldots$ with $\alpha_k \in \Sigma$ for every i has a limit $\alpha \in \Sigma$. An illustration is shown in Figure 3, where the purple dots represent a sequence of executions α_i taken from the connected component Σ_{γ_0} and $\times$ the limit point α at the boundary. The most prominent examples of limit-closed message adversaries are oblivious ones [14, 44, 46].

We recall that dynamic networks consist of a set of n lock-step synchronous fault-free processes, which execute a deterministic consensus algorithm that broadcasts its entire local state via message-passing in each of the communication-closed rounds $1, 2, \ldots$ A message adversary determines which process q receives the message broadcast by a process p in some round t, via the directed round-t communication graph $\mathcal{G}^t$. Together with the initial configuration C^0 of all the processes, the particular sequence of communication graphs $\mathcal{G}^1, \mathcal{G}^2, \ldots$, which is called communication pattern, uniquely determines an execution. For example, an oblivious message adversary is defined by a set $\mathbf{D}$ of allowed communication graphs and picks every $\mathcal{G}^t$ arbitrarily from this set.

Since all processes are obedient here, we will only consider the uniform topology in the sequel. The set of all process-time graphs $\mathcal{PT}^\omega$ is compact and the transition function $\hat{\tau} : \mathcal{PT}^\omega \to \mathcal{C}^\omega$ is continuous, according to Lemma A.2, so taking $\hat{\tau}(\mathcal{PT}^\omega)$ results in a set of configuration sequences that is indeed compact. Note that limit-closed message adversaries are hence sometimes refered to as *compact* message adversaries.

The following consensus characterization holds even for general message adversaries:

Corollary 8.2 (Consensus characterization for general MAs). *Consensus with independent arbitrary input assignments is solvable under a general message adversary if and only if (i) all connected components of the set Σ of admissible executions are broadcastable for some process, and (ii) the strong broadcaster decision sets Σ_v^p, $p \in \Pi$, $v \in \mathcal{V}$, given in Definition 7.8, are closed in Σ.*

Proof. Since there are only finitely many Σ_v^p, $p \in \Pi$, $v \in \mathcal{V}$, closedness is equivalent to clopenness here. Hence, Theorem 7.9 can be applied. □

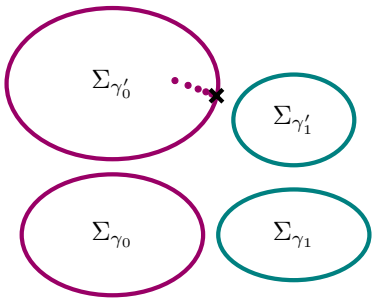

Fig. 3. Examples of two connected components of the decision sets $\Sigma_0 = \Sigma_{\gamma_0} \cup \Sigma_{\gamma_0'}$ and $\Sigma_1 = \Sigma_{\gamma_1} \cup \Sigma_{\gamma_1'}$ for consensus under a limit-closed message adversary. contain all their limit points (marked by $\times$) and have a distance > 0 by cor:closeddecsetscompact.

We will start our considerations for limit-closed message adversaries by exploring the structure of the strong decision sets (recall Definition 7.8) of correct consensus algorithms, see Fig. 3 for an illustration.

Corollary 8.3 (Strong decision sets for limit-closed MAs). *For every correct consensus algorithm for a limit-closed message adversary, both the strong broadcaster decision sets Σ_v^p, $p \in \Pi$, $v \in \mathcal{V}$, and the strong decision sets Σ_v, $v \in \mathcal{V}$, are disjoint, compact and clopen in Σ. Moreover, there is some $d > 0$ such that $d_{\mathrm{u}}(\Sigma_v^p, \Sigma_w^q) \geq d > 0$ for any $(v,p),(v,p) \neq (w,q) \in \Pi \times \mathcal{V}$, as well as $d_{\mathrm{u}}(\Sigma_v, \Sigma_w) \geq d > 0$ for every $v, v \neq w \in \mathcal{V}$.*

In addition, every connected component $\Sigma_\gamma \subseteq \Sigma$ is closed and compact, and for every γ, δ with $\Sigma_\gamma \neq \Sigma_\delta$, it holds that $d_{\mathrm{u}}(\Sigma_\gamma, \Sigma_\delta) > 0$.

Proof. According to Theorem 7.9, all strong broadcaster decision sets Σ_v^p are clopen, and hence closed, in Σ. Since Σ is compact for a limit-closed message adversary, it follows that every Σ_v^p is also compact. Corollary 6.5 thus implies $d_{\mathrm{u}}(\Sigma_v^p, \Sigma_w^q) > 0$. Since there are only finitely many Σ_v^p, there is hence some $d > 0$ that guarantees $d_{\mathrm{u}}(\Sigma_v^p, \Sigma_w^q) \geq d > 0$ for every $(v,p) \neq (w,q) \in \Pi \times \mathcal{V}$. As $\Sigma_v = \bigcup_{p \in \Pi} \Sigma_v^p$ is a finite union, the respective results for the strong decision sets follow immediately as well.

Since every connected component Σ_γ of Σ that contains γ is closed in Σ, as the closure of a connected subspace is also connected [36, Lem. 23.4] and a connected component is maximal, Σ_γ is also compact, and $d_{\mathrm{u}}(\Sigma_\gamma, \Sigma_\delta) > 0$ follows from Corollary 6.5. □

Unfortunately, Corollary 8.3 does *not* allow us to also infer some minimum distance $d > 0$ also for the connected components in general. It does hold true, however, if there are only finitely many connected components. The latter is ensured, in particular, when Σ is *locally connected*, in the sense that every open set $U(\delta) \subseteq \Sigma$ containing δ also contains

some *connected* open set $V(\delta)$: According to [36, Thm. 25.3], all connected components of Σ are also open in this case. Hence, $\Sigma = \bigcup_{\gamma\in\Sigma} \Sigma_\gamma$ is an open covering of Σ, and since Σ is compact, there is a finite sub-covering $\Sigma = \Sigma'_{\gamma^1} \cup \cdots \cup \Sigma'_{\gamma^m}$. Every Σ_γ must hence be equal to one of $\Sigma_{\gamma^1}, \ldots, \Sigma_{\gamma^m}$, as connected components are either disjoint or identical.

Unfortunately, however, most limit-closed message adversaries do not guarantee local connectedness. In the case of oblivious message adversaries, in particular, it has been argued [22] that isolated "islands" are created in the evolution of the protocol complex, which further develop like the original protocol complex (that must be not connected for consensus to be solvable). This phenomenon may be viewed as the result of the "self-similarity" that is inherent in the communication patterns created by such message adversaries, which is not compatible with local connectedness.

In general, for limit-closed message adversaries that induce infinitely many connected components in Σ, one cannot infer openness (and hence clopenness) of the connected components: Consider a decision set Σ_v that consists of infinitely many connected components. Whereas any connected component Σ_γ is closed, the set of all remaining connected components $\Sigma_v \setminus \Sigma_\gamma$ need not be closed. It may hence be possible to pick a sequence of executions (α_k) from $\Sigma_v \setminus \Sigma_\gamma$ that converges to a limit α, and a sequence (β_k) from Σ_γ that converges to $\beta \in \Sigma_\gamma$, satisfying $d_{\mathrm{u}}(\alpha, \beta) = 0$. By Lemma 6.2, this implies $d_{\mathrm{u}}(\Sigma_\gamma, \Sigma_v \setminus \Sigma_\gamma) = 0$. It is important to note, however, that this can only happen for connected components in the *same* strong broadcaster decision set Σ_v^p, as $d_{\mathrm{u}}(\Sigma_v^p, \Sigma_w^q) \geq d > 0$ prohibits a common limit across different decision sets. Consequently, consensus solvability is not per se impaired by infinitely many connected components, as Corollary 8.2 has shown.

We will now make the characterization of Corollary 8.2 for limit-closed message adversaries more operational, by introducing the ε-approximation of connected components and strong broadcaster decision sets, typically for some $\varepsilon = 2^{-t}$, $t \geq 0$. Informally, it provides the executions that have a t-prefix that cannot be transitively distinguished by some process. Since the number of different possible t-prefixes is finite, it can be constructed iteratively using finitely many iterations:

Definition 8.4 (ε-approximations). Let $\gamma \in \Sigma$ be an admissible execution. In the minimum topology, we iteratively define $\Sigma_\gamma^\varepsilon$, for $\varepsilon > 0$, as follows: $\Sigma_\gamma^\varepsilon[0] = \{\gamma\}$; for $\ell > 0$, $\Sigma_\gamma^\varepsilon[\ell] = \bigcup_{\alpha\in\Sigma_\gamma^\varepsilon[\ell-1]} (B_\varepsilon(\alpha) \cap \Sigma)$; and $\Sigma_\gamma^\varepsilon = \Sigma_\gamma^\varepsilon[m]$ where $m < \infty$ is such that $\Sigma_\gamma^\varepsilon[m] = \Sigma_\gamma^\varepsilon[m+1]$.

For $p \in \Pi$, $v \in \mathcal{V}$, the ε-approximation $\Sigma_v^{p,\varepsilon}$ is defined as $\Sigma_v^{p,\varepsilon} = \bigcup_{\Sigma_\gamma\subseteq\Sigma_v^p} \Sigma_\gamma^\varepsilon$.

Note carefully that $\Sigma_\gamma^\varepsilon$ is generally different (in fact, larger) than the covering of Σ_γ with ε-balls defined by $\bigcup_{\delta\in\Sigma_\gamma} B_\varepsilon(\delta) \cap \Sigma$. Our ε-approximations satisfy the following properties (that actually hold for general message adversaries):

Lemma 8.5 (Properties of ε-approximations of connected components). *For every $\varepsilon > 0$ and every $\gamma, \delta \in \Sigma$, ε-approximations have the following properties:*

(i) $\Sigma_\gamma^{\varepsilon'} \subseteq \Sigma_\gamma^\varepsilon$ *for every* $0 < \varepsilon' \leq \varepsilon$.
(ii) $\Sigma_\gamma^\varepsilon \cap \Sigma_\delta^\varepsilon \neq \emptyset$ *implies* $\Sigma_\gamma^\varepsilon = \Sigma_\delta^\varepsilon$.
(iii) $\Sigma_\gamma \subseteq \Sigma_\gamma^\varepsilon$.

Proof. To prove (i), it suffices to mention $B_{\varepsilon'}(\alpha) \subseteq B_\varepsilon(\alpha)$. As for (ii), if $\alpha \in \Sigma_\gamma^\varepsilon \cap \Sigma_\delta^\varepsilon \neq \emptyset$, the iterative construction of $\Sigma_\gamma^\varepsilon$ would reach α, which would cause it to also include the whole $\Sigma_\delta^\varepsilon$, as the latter also reaches α. If (iii) would not hold, Σ_γ could be separated into disjoint open sets, which contradicts its connectivity. □

Obviously, properties (i) and (iii) of the ε-approximation of connected components also extend to arbitrary unions of those, and hence to strong broadcaster decision sets. In fact, for limit-closed message adversaries, provided ε is chosen sufficiently small, we get the following result:

Lemma 8.6 (ε-approximation of strong broadcaster decision sets). *For a limit-closed message adversary that allows to solve consensus, there is some $\varepsilon > 0$ such that, for any $0 < \varepsilon' \leq \varepsilon$, it holds that $d_{\mathrm{u}}(\Sigma_v^{p,\varepsilon'}, \Sigma_w^{q,\varepsilon'}) > 0$ for any $(v,p),(v,p) \neq (w,q) \in \Pi \times \mathcal{V}$.*

Proof. According to Corollary 8.3, there is some $d > 0$ such that $d_{\mathrm{u}}(\Sigma_v^p, \Sigma_w^q) \geq d > 0$. By the extension of Lemma 8.5.(iii) to strong broadcaster decision sets, for any $\varepsilon > 0$, $\Sigma_v^p \subseteq \Sigma_v^{p,\varepsilon}$ and $\Sigma_w^q \subseteq \Sigma_w^{q,\varepsilon}$. Therefore, setting $\varepsilon < d/2$ secures $d_{\mathrm{u}}(\Sigma_v^{p,\varepsilon}, \Sigma_w^{q,\varepsilon}) > 0$. By the extension of Lemma 8.5.(i) to strong broadcaster decision sets, we hence also get $d_{\mathrm{u}}(\Sigma_v^{p,\varepsilon'}, \Sigma_w^{q,\varepsilon'}) > 0$. □

Corollary 8.7 (Matching ε-approximation). *For a limit-closed message adversary that allows to solve consensus, if $\varepsilon > 0$ is chosen in accordance with Lemma 8.6, then $\Sigma_v^{p,\varepsilon} = \Sigma_v^p$ for every $p \in \Pi$, $v \in \mathcal{V}$.*

Theorem 8.8 (Operational consensus characterization for limit-closed MAs). *A limit-closed message adversary allows to solve consensus if and only if there is some $\varepsilon > 0$ such that (i) every $\Sigma_\gamma^\varepsilon$, $\Sigma_\gamma \subseteq \Sigma$, is broadcastable for some process, and (ii) every $\Sigma_v^{p,\varepsilon}$, $p \in \Pi$, $v \in \mathcal{V}$, is closed in Σ.*

Proof. Our theorem follows from Corollary 8.2 in conjunction with Corollary 8.7. □

Theorem 8.8 implies that if consensus is solvable, then, for every $0 < \varepsilon' \leq \varepsilon$, the universal algorithm from Theorem 7.12 applied to the strong decision sets can be used for actually solving it. And indeed, the consensus algorithm given by Winkler, Schmid, and Moses [47, Alg. 1] can be viewed as an instantiation of this fact.

Moreover, Corollary 8.7 implies that checking the broadcastability of all the executions in $\Sigma_v^{p,\varepsilon}$ can be done by checking the broadcastability of *finite* prefixes. More specifically, like the decision function Δ of consensus, the function $T(\alpha)$ that gives the round by which every process in $\alpha \in \Sigma$ has the initial value $I_p(\alpha)$ of the broadcaster p in its view is locally constant for a sufficiently small neighborhood, namely, $B_{2^{-T(\alpha)}}(\alpha)$, and is hence continuous in any of our topologies. Since $\Sigma_v^p = \Sigma_v^{p,\varepsilon}$ is compact, $T(\alpha)$ is in fact uniformly continuous and hence attains its maximum $\hat{T}$ in $\Sigma_v^{p,\varepsilon}$. It hence suffices to check broadcastability in the t-prefixes of $\Sigma_v^{p,\varepsilon}$ for $t = \max\{\lfloor \log_2(1/\varepsilon) \rfloor, \hat{T}\}$ in Theorem 8.8.

In [47], this has been translated into the following non-topological formulation (where MA corresponds to Σ, $[\sigma|_r]$ is the set of r-prefixes of the executions in $\Sigma_\sigma^{2^{-r}}$ in the uniform topology, and $\mathrm{Ker}(x)$ is the set of broadcasters in the prefix x):

Theorem 8.9 ([47, Thm. 1]). *Consensus is solvable under a limit-closed message adversary MA if and only if for each $\sigma \in$ MA there is a round r such that $\bigcap_{x \in [\sigma|_r]} Ker(x) \neq \emptyset$.*

8.5 Dynamic networks with non-limit closed message adversaries

In this section, we consider consensus with independent and arbitrary input assignments under message adversaries that are not limit-closed [17, 41, 49]. A simple example would be a message adversary, which guarantees that there is some *finite* round r where the communication graph $\mathcal{G}^r$ is a clique. The communication pattern where $r = \infty$, i.e., the limiting case $r \to \infty$ (where the clique graph never happens) is forbidden.

As already mentioned in Section 8.4, our consensus characterization Corollary 8.2 also applies here, as does the generic one in Theorem 7.12, of course. Moreover, they can be combined with our limit-based characterization Theorem 6.4 and Corollary 6.7.

What does not work here, however, are our ε-approximations according to Definition 8.4, and everything built on top of it: Even if ε would be made arbitrarily small, Lemma 8.6 does not hold. An illustration is shown in Figure 4. It is apparent that adding a ball $B_\varepsilon(\alpha)$ in the iterative construction of some $\Sigma^\varepsilon_\gamma$, where $d_{\mathrm{u}}(\alpha, \rho) < \varepsilon$ for some forbidden limit sequence ρ, inevitably lets the construction grow into some $\Sigma^\varepsilon_\delta$ lying in a different strong broadcaster decision set. Whereas this could be avoided by adapting ε when coming close to r, the resulting approximation would not provide any advantage over directly using our characterization Corollary 8.2.

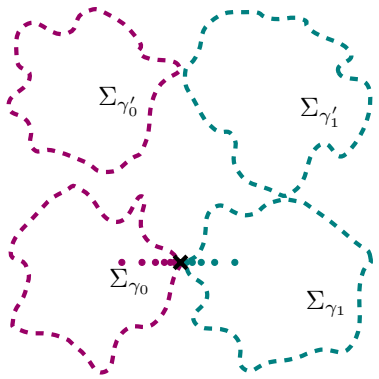

Fig. 4. Examples of two connected components of the decision sets $\Sigma_0 = \Sigma_{\gamma_0} \cup \Sigma_{\gamma'_0}$ and $\Sigma_1 = \Sigma_{\gamma_1} \cup \Sigma_{\gamma'_1}$ for a non-compact message adversary. They are not closed in $\mathcal{C}^\omega$ and may have distance 0; common limit points (like for Σ_{γ_0} and Σ_{γ_1}, marked by $\times$) must hence be excluded by Corollary 6.4.

These topological findings are of course in accordance with the results on non-limit closed message adversaries we are aware of. In particular, the binary consensus algorithm for $n = 2$ by Fevat and Godard [17] assumes that the algorithm knows a fair execution or a pair of unfair executions according to Definition 6.6 a priori, which effectively partition the execution space into two connected components.[3] Such a limit exclusion is also exploited in the counterexample to consensus task solvability for $n = 2$ via a decision map that is not continuous [20], which has been suggested by Godard and Perdereau [23]: It excludes just the unfair execution α based on $\{\leftrightarrow, \leftarrow, \leftarrow, \dots\}$, but not the unfair execution β caused by $\{\rightarrow, \leftarrow, \leftarrow, \dots\}$, which satisfies $d_p(\alpha, \beta) = 0$ for the right process p and hence makes consensus impossible.

The $(D+1)$-VSRC message adversary $\Diamond\mathsf{STABLE}_n(D+1)$ [49] generates executions that are based on single-rooted communication graphs in every round, with the additional guarantee that, eventually, a $D+1$-*vertex-stable root component* ($D+1$-VSRC) occurs. Herein, a root component is a strongly connected component without in-edges from outside the component, and a x-VSRC is a root component made up of the same *set* of processes in x consecutive rounds. $D \leq n-1$ is the dynamic diameter of a VSRC, which guarantees that all root members reach all processes. It has been proved [49] that consensus is impossible with $\Diamond\mathsf{STABLE}_n(x)$ for $x \leq D$, whereas an algorithm exists for $\Diamond\mathsf{STABLE}_n(D+1)$. Obviously,

[3]Note that there are uncountably many choices for separating Σ_0 and Σ_1 here, however.

$\Diamond\mathsf{STABLE}_n(D+1)$ effectively excludes all communication patterns without any $D+1$-VSRC. And indeed, the choice $x = D+1$ renders the connected components of Σ broadcastable by definition, which is in accordance with Corollary 8.2.

We also introduced and proved correct an explicit labeling algorithm for $\Diamond\mathsf{STABLE}_n(n)$ in [48], which effectively operationalizes the universal consensus algorithm of Theorem 7.12: By assigning a (persistent) *label* $\Delta(\sigma'|_r)$ to the r-prefixes of $\sigma \in \Sigma$, it effectively assigns a corresponding unique decision value $v \in \mathcal{V}$ to σ, which in turn specifies the strong decision set Σ_v containing σ. In the language of [48], the requirement of every Σ_v being open (and closed) in Theorem 7.12 translates into a matching assumption on this labeling function as follows (herein, MA corresponds to Σ, $\sigma|_r$ denotes the r-round prefix of execution σ, and $\sim$ is the transitive closure over all processes p of the prefix indistinguishability relation $\sim_p$):

Assumption 1 ([48, Assumpt. 1]). $\forall \sigma \in \mathsf{MA}\ \exists r \in \mathbb{N}\ \forall \sigma' \in \mathsf{MA}\colon \sigma'|_r \sim \sigma|_r \Rightarrow \Delta(\sigma'|_r) = \Delta(\sigma|_r) \neq \emptyset$.

For $\Diamond\mathsf{STABLE}_n(n)$, it has been proved [48, Thm. 12] that the given labeling algorithm satisfies this assumption for $r = r_{stab} + 4n$, where r_{stab} is the round where the (first) $D+1$-VSRC in σ starts. Consensus is hence solvable by a suitable instantiation of the universal consensus algorithm of Theorem 7.12.

8.6 Consensus in systems with an eventually timely f-source

It is well-known [15] that consensus cannot be solved in distributed systems of $n \geq 2f+1$ (partially) synchronous processes, up to which f may crash, which are connected by reliable *asynchronous* communication links. For solving consensus, the system model has been strengthened by a *weak timely link* (WTL) assumption [3, 26]: there has to be at least one correct process p that eventually sends timely to a sufficiently large subset of the processes.

In previous work [3], at least one *eventually timely f-source* p was assumed: After some unknown initial period where all end-to-end message delays are arbitrary, every broadcast of p is received by a fixed subset $P \subseteq \Pi$ with $p \in P, |P| \geq f+1$ within some possibly unknown maximum end-to-end delay Θ. The authors showed how to build the Ω failure detector in such a system, which, in conjunction with any Ω-based consensus algorithm (like the one by Mostéfaoui and Raynal [35]) can be used to solve uniform consensus.

Their Ω implementation lets every process broadcast a *heartbeat message* every η steps, which forms partially synchronized rounds, and maintains an *accusation counter* for every process q that counts the number of rounds the heartbeats of which were not received timely by more than f processes. This is done by letting every process who does not receive q's broadcast within Θ send an *accusation message* for q, and incrementing the accusation counter for q if more than f such accusation messages from different receivers came in. It is not difficult to see that the accusation counter of a process that crashes grows unboundedly, whereas the accusation counter of every timely f-source eventually stops being incremented. Since the accusation counters of all processes are exchanged and agreed-upon as well, choosing the process with the smallest accusation counter (with ties broken by process ids) is a legitimate choice for the output of Ω.

This WTL model was further relaxed [26], which allows the set $P(k)$ of witnessing receivers of every *eventually moving timely f-source* to depend on the sending round k. The price to be paid for this relaxation is the need to incorporate the sender's round number in the heartbeat and accusation messages.

In this subsection, we will use our Theorem 7.12 to prove topologically that consensus with strong validity and independent arbitrary input assignments can indeed be solved in

the WTL model: We will give and prove correct an explicit labeling algorithm Algorithm 1, which assigns a decision value $v \in \mathcal{V}$ to every execution σ that specifies the decision set Σ_v containing σ. Applying our universal algorithm to these decision sets hence allows to solve consensus in this model. Obviously, unlike the existing algorithms, our algorithm does not rely on an implementation of Ω.

We assume a (slightly simplified) WTL model with synchronous processes and asynchronous links that are reliable and FIFO, with known Θ for timely links. Whereas we will use the time $t = 0, 1, 2, \ldots$ our synchronous processes take their steps as global time, we note that we do not have communication-closed rounds here, i.e., have to deal with general executions according to Definition A.1 in the appendix. In an admissible execution σ, we denote by $F(\sigma)$ the set of up to f processes that crash in σ, and $C(\sigma) = Ob(\sigma) = \Pi \setminus F(\sigma)$ the set of correct processes. For an eventual timely f-source p, we will denote with $r_{\mathrm{stab},p}$ the *stabilization round*, by which it has already started to send timely: a message sent in round $t \geq r_{\mathrm{stab},p}$ is received by every $q \in P(t)$ no later than in round $t + \Theta - 1$, hence is present in q's state at time $t + \Theta - 1$. Note carefully that this condition is automatically satisfied when q has crashed by that round. We again assume that the processes execute a full-information protocol, i.e., send their whole state in every round. For keeping the relation to the existing algorithms, we consider the state message sent by p in round t to be its $\mathsf{heartbeat}(t)$. Moreover, if the state of process q at time $t + \Theta - 1$ does not contain the reception $\mathsf{heartbeat}(t)$ from process p, we will say that q broadcasts an *accusation message* $\mathsf{accusation}(p, t)$ for round t of p in round $t + \Theta$ (which is of course just part of q's state sent in this round). If q crashes before round $t + \Theta$, it will never broadcast $\mathsf{accusation}(p, t)$. If q crashes exactly in round $t + \Theta$, we can nevertheless assume that it either manages to eventually communicate $\mathsf{accusation}(p, t)$ to all correct processes in the system, or to none: In our full information protocol, every process that receives $\mathsf{accusation}(p, t)$ will forward also this message to all other processes when it broadcasts its own state later on.

Definition 8.10 (WTL elementary state predicates and variables). For process s at time $r \geq 1$, i.e., the end of round r, we define the following predicates and state variables:

- $\mathsf{accuse}_s^r(p) = \mathsf{true}$ if and only if s did not receive $\mathsf{heartbeat}(r - \Theta)$ from p by time r and thus sent $\mathsf{accusation}(p, t)$.
- $\mathsf{nottimelyrec}_s^r(q, p, t) = \mathsf{true}$ if and only if s recorded the reception of $\mathsf{accusation}(p, t)$ from q by time r.
- $\mathsf{nottimely}_s^r(p, t) = \mathsf{true}$ if and only if $\mathsf{nottimelyrec}_s^r(q, p, t) = \mathsf{true}$ for at least $n - f$ different $q \in \Pi$.
- $\mathsf{accusationcounter}_s^r(p) = (|\{k \leq r : \mathsf{nottimely}_s^r(p, k) = \mathsf{true}\}|, p)$.
- $\mathsf{heardof}_s^r(p) = |\{k \leq r : s \text{ received } \mathsf{heartbeat}(k) \text{ from } p \text{ (directly or indirectly) by time } r\}|$.

Note that a process q that crashes before time $t + \Theta$ causes $\mathsf{nottimelyrec}_s^r(q, p, t) = \mathsf{false}$ for all r, and that p is appended in $\mathsf{accusationcounter}_s^r(p)$ for tie-breaking purposes only. For every eventually timely f-source p, the implicit forwarding of accusation messages ensures that $\mathsf{accusationcounter}_s^r(p)$ will eventually be the same at every correct process s in the limit $r \to \infty$.

We now define some predicates that require knowledge of the execution σ. Whereas they cannot be computed locally by the processes in the execution, they can be used in the labeling algorithm.

Definition 8.11 (WTL extended state predicates and variables). Given an execution σ, let the *dominant* eventual timely f-source p_σ be the one that leads to the unique smallest

value of $\mathsf{accusationcounter}_s^{\infty}(p_\sigma)$, which is the same at every process $s \in \Pi \setminus F(\sigma)$. With $r_{\mathrm{stab},\sigma} = r_{\mathrm{stab},p_\sigma}$ denoting the stabilization time of the dominant eventual timely f-source in σ and $F(\sigma|_r) \subseteq F(\sigma)$ the set of processes that crashed by time r, we also define

- $\mathsf{minheardof}_s(\sigma, r) = \min_{p \in \Pi \setminus F(\sigma|_r)} \mathsf{heardof}_s^r(p)$,
- $\mathsf{oldenough}(\sigma, r) = \mathsf{true}$ if and only if $\forall s \in \Pi \setminus F(\sigma|_r)$, both (i) $\mathsf{minheardof}_s(\sigma, r) \geq r_{\mathrm{stab},\sigma} + \Theta$ and (ii) $\forall p \in \Pi \setminus p_\sigma : \mathsf{accusationcounter}_s^r(p_\sigma) < \mathsf{accusationcounter}_s^r(p)$.
- $\mathsf{mature}(\sigma, r) = \mathsf{true}$ if and only if $\exists r_0 < r$ such that both (i) $\mathsf{oldenough}(\sigma, r_0) = \mathsf{true}$ and (ii) $\forall s \in \Pi \setminus F(\sigma|_r) : \mathsf{minheardof}_s(\sigma, r) \geq r_0$.

Note that it may occur that another eventual timely f-source $p' \neq p_\sigma$ in σ has a smaller stabilization time $r_{\mathrm{stab},p'} < r_{\mathrm{stab},p_\sigma}$ than the dominant one, which happens if p' causes more accusations than p_σ before stabilization in total.

The following properties are almost immediate from the definitions:

Lemma 8.12 (Properties of oldenough and mature). *The following properties hold for* oldenough*:*

(i) *If* $\mathsf{oldenough}(\sigma, r) = \mathsf{true}$*, then* $\mathsf{accusationcounter}_s^t(p_\sigma) = \mathsf{accusationcounter}_s^r(p_\sigma)$ *for every* s *that did not crash by time* $t \geq r$*.*

(ii) $\mathsf{oldenough}(\sigma, r)$ *is stable, i.e.,* $\mathsf{oldenough}(\sigma, r) = \mathsf{true} \Rightarrow \mathsf{oldenough}(\sigma, t) = \mathsf{true}$ *for* $t \geq r$*.*

(iii) *(i) and (ii) also hold for* $\mathsf{mature}(\sigma, r)$*, and* $\mathsf{mature}(\sigma, r) = \mathsf{true} \Rightarrow \mathsf{oldenough}(\sigma, r) = \mathsf{true}$*.*

Proof. Since $\mathsf{oldenough}(\sigma, r) = \mathsf{true}$ entails that every process $s \in \Pi \setminus F(\sigma|_r)$ has received the accusation messages for all rounds up to $r_{\mathrm{stab},\sigma}$ since $\mathsf{minheardof}_s(\sigma, r) \geq r_{\mathrm{stab},\sigma} + \Theta$ according to Definition 8.11, (i) follows. This also implies (ii), since the accusation counter of every process $p \neq p_\sigma$ can at most increase after time r. That these properties carry over to mature is obvious from the definition. □

The following lemma proves that two executions σ and ρ with indistinguishable prefixes $\sigma|_r \sim_s \rho|_r$, i.e., $(\sigma|_r)^t \sim_s (\rho|_r)^t$ for $0 \leq t \leq r$, cannot both satisfy $\mathsf{oldenough}(\sigma, r)$ resp. $\mathsf{oldenough}(\rho, r)$, and, hence, $\mathsf{mature}(\sigma, r)$ resp. $\mathsf{mature}(\rho, r)$, except when the dominant eventual timely f-source is the same in σ and ρ:

Lemma 8.13. *Consider two executions* σ *and* ρ *with* $\sigma|_r \sim_s \rho|_r$ *for some process* s *that is not faulty by round* r *in both* σ *and* ρ*. Then,*

$$(\mathsf{oldenough}(\sigma, r) = \mathsf{true} \wedge \mathsf{oldenough}(\rho, r) = \mathsf{true}) \Rightarrow p_\sigma = p_\rho.$$

Proof. As $\mathsf{oldenough}(\sigma, r) = \mathsf{true}$, Definition 8.11 implies $\forall p \in \Pi \setminus p_\sigma : \mathsf{accusationcounter}_s^r(p_\sigma) < \mathsf{accusationcounter}_s^r(p)$, and similarly $\forall p \in \Pi \setminus p_\rho : \mathsf{accusationcounter}_s^r(p_\rho) < \mathsf{accusationcounter}_s^r(p)$. Since $\sigma|_r \sim_s \rho|_r$, this is only possible if $p_\sigma = p_\rho$. □

Finally, we need the following technical lemmas:

Lemma 8.14 (Indistinguishability precondition). *Suppose* $\tau|_{r'} \sim_{s'} \sigma|_{r'}$ *is such that* s' *received a message from* $s \neq s'$ *containing its state in the sending round* $r'_0 \leq r' - 1$ *by round* r' *in* $\sigma|_{r'}$ *and hence also in* $\tau|_{r'}$*. Analogously, suppose* $\sigma|_r \sim_s \rho|_r$ *is such that* s *received a message from* s' *containing its state in the sending round* $r_0 \leq r - 1$ *by round* r *in* $\sigma|_r$ *and hence also in* $\rho|_r$*. Then,*

(i) $\tau|_{r'_0} \sim_s \sigma|_{r'_0}$,

(ii) $\tau|_{\min\{r'_0, r\}} \sim_s \rho|_{\min\{r'_0, r\}}$,

(iii) $\sigma|_{r_0} \sim_{s'} \rho|_{r_0}$,

(iv) $\tau|_{\min\{r_0,r'\}} \sim_{s'} \rho|_{\min\{r_0,r'\}}$.

Proof. If (i) would not hold, since s sends a message containing its state in round r'_0 to s' both in $\tau|_{r'}$ and in $\sigma|_{r'}$, these two states would be distinguishable for s, which contradicts our assumption. The analogous argument proves (iii). Statement (ii) follows from combining (i) with $\sigma|_r \sim_s \rho|_r$, (iv) follows from combining (iii) with $\tau|_{r'} \sim_{s'} \sigma|_{r'}$. □

Lemma 8.15 (Heardof inheritance). *Suppose $\sigma|_r \sim_s \rho|_r$ and $\mathsf{minheardof}_s(\rho, r) \geq r_0$ for some $1 \leq r_0 < r$, as it arises in $\mathsf{mature}(\rho, r) = \mathsf{true}$, for example. Then, $\forall p \in \Pi \setminus F(\rho|_r)$, it also holds in $\sigma|_r$ that $\mathsf{heardof}'_s(p) \geq r_0$, but not necessarily $\mathsf{heardof}'_s(p') \geq r_0$ for $p' \in (\Pi \setminus F(\sigma|_r)) \cap F(\rho|_r)$. Consequently, it may happen that $\mathsf{minheardof}_s(\sigma, r) < r_0$.*

Proof. Since the state of s is the same in $\sigma|_r$ and $\rho|_r$, but the sets $F(\rho|_r)$ and $F(\sigma|_r)$ may be different, the lemma follows trivially. □

With the abbreviation $C(\sigma|_r) = \Pi \setminus F(\sigma|_r)$ for all non-faulty processes in $\sigma|_r$, and $\sigma|_r \sim_Q \rho|_r$ for $\forall q \in Q : \ \sigma|_r \sim_q \rho|_r$, we define the short-hand notation $\sigma|_r \sim_{\geq n-f} \rho|_r$ to express indistinguishability for a majority of (correct) processes, defined by $\exists Q \subseteq C(\sigma|_r) \cap C(\rho|_r), |Q| \geq n - f$ such that $\forall q \in Q : \ \sigma|_r \sim_q \rho|_r$.

The following lemma guarantees that prefixes that are indistinguishable only for strictly less than $n - f$ processes are eventually distinguishable for all processes:

Lemma 8.16 (Vanishing minority indistinguishability). *Given $\rho|_{r_0}$, there is a round r, $r_0 \leq r < \infty$, such that for every $\sigma|_{r_0}$ with $\rho|_{r_0} \not\sim_{\geq n-f} \sigma|_{r_0}$, it holds that $\rho|_r \not\sim \sigma|_r$.*

Proof. Due to our reliable link assumption, for every process s that does not fail in ρ, there is a round $r > r_0$ where $\mathsf{minheardof}_s(\rho, r) \geq r_0$. Now assume that there is some $\sigma|_{r_0}$ with $\rho|_{r_0} \sim_Q \sigma|_{r_0}$ for a maximal set Q with $1 \leq |Q| < n - f$, but $\rho|_r \sim_s \sigma|_r$ for some process s. Since s receives round-r_0 messages from $|C(\rho|_n)| \geq n - f$ processes in $\rho|_r$, and $\rho|_r \sim_s \sigma|_r$, process s must receive exactly the same messages also in $\sigma|_r$. As at most $|Q| < n - f$ of those messages may be sent by processes that cannot distinguish $\rho|_{r_0} \sim_Q \sigma|_{r_0}$, at least one such message must originate in a process q' with $\rho|_{r_0} \not\sim_{q'} \sigma|_{r_0}$. In this case, Lemma 8.14.(iii) prohibits $\rho|_r \sim_s \sigma|_r$, however, which provides the required contradiction. □

The following lemma finally shows that majority indistinguishability in conjunction with mature prefixes entails strong indistinguishability properties in earlier rounds:

Lemma 8.17 (Majority indistinguishability precondition). *Suppose $\tau|_r \sim_{\geq n-f} \sigma|_r \sim_{\geq n-f} \rho|_r$ and $\mathsf{mature}(\rho, r) = \mathsf{true}$. Then, for the round r_0 imposed by the latter, it holds that $\tau|_{r_0} \sim_{C(\rho|_r)} \sigma|_{r_0} \sim_{C(\rho|_r)} \rho|_{r_0}$, and hence also $\tau|_{r_0} \sim_{C(\rho|_r)} \rho|_{r_0}$.*

Proof. Let S resp. Q be the set of at least $n - f$ processes causing $\sigma|_r \sim_{\geq n-f} \rho|_r$ resp. $\sigma|_r \sim_{\geq n-f} \tau|_r$. Since $Q \cap S \neq \emptyset$ by the pigeonhole principle, let $s \in Q \cap S$. Clearly, $\tau|_r \sim_s \sigma|_r \sim_s \rho|_r$, and hence also $\tau|_r \sim_s \rho|_r$. Since $\mathsf{mature}(\rho, r) = \mathsf{true}$, Lemma 8.14.(i) in conjunction with Lemma 8.15 implies $\rho|_{r_0} \sim_{C(\rho|_r)} \sigma|_{r_0}$, as well as $\rho|_{r_0} \sim_{C(\rho|_r)} \tau|_{r_0}$, and hence also $\sigma|_{r_0} \sim_{C(\rho|_r)} \tau|_{r_0}$ as asserted. □

With these preparations, we can define an explicit labeling algorithm Algorithm 1 for the WTL model, i.e., an algorithm that computes a label $\Delta(\sigma|_r)$ for every r-prefix $\sigma|_r$ of an admissible execution σ in our WTL model. A label can either be $\emptyset$ (still undefined) or else denote a single process p (which will turn out to be a broadcaster), and will be persistent in σ in the sense that $\Delta(\sigma|_r) = p \Rightarrow \Delta(\sigma|_{r+k}) = p$ for every $k \geq 0$. Note that we can hence

uniquely also assign a label $\Delta(\sigma)$ to an infinite execution. Note that, for defining our decision sets, we will assign σ to Σ_{I_p}, where I_p is the initial value of $p = \Delta(\sigma)$ in σ.

Informally, our labeling algorithm works as follows: If there is some unlabeled mature prefix $\rho|_r$, it is labeled either (i) with the label of some already labeled but not yet mature $\sigma|_r$ if the latter got its label early enough, namely, by the round r_0 where $\mathsf{oldenough}(\rho, r_0) = \mathsf{true}$, or else (ii) with its dominant p_ρ.

Algorithm 1: Computing Δ for each r-prefix $\sigma|_r$ in the WTL model.

```
1  Initially, let Δ(σ|_0) = ∅.
2  for r = 1, 2, ... do
3  |  foreach σ|_r do  Δ(σ|_r) ← Δ(σ|_{r−1})
4  |  foreach ρ|_r with Δ(ρ|_r) = ∅ do
5  |  |  if ∃σ|_r ∼_{≥n−f} ρ|_r with Δ(σ|_r) = p ≠ ∅ and mature(σ, r) = true then
6  |  |  |  Δ(ρ|_r) ← p
7  |  foreach ρ|_r with Δ(ρ|_r) = ∅ and mature(ρ, r) = true do
8  |  |  if ∃σ|_r ∼_{≥n−f} ρ|_r with Δ(σ|_{r_0}) = p ≠ ∅ for r_0 satisfying
   |  |   oldenough(ρ, r_0) = true then
9  |  |  |  Δ(ρ|_r) ← p
10 |  foreach ρ|_r with Δ(ρ|_r) = ∅ and mature(ρ, r) = true do
11 |  |  if ∃σ|_r ∼_{≥n−f} ρ|_r with Δ(σ|_r) = p ≠ ∅ and mature(σ, r) = true then
12 |  |  |  Δ(ρ|_r) ← p // Only happens when σ|_r got its label in line 9
13 |  |  else
14 |  |  |  Δ(ρ|_r) ← p_ρ
```

The following Theorem 8.18 in conjunction with Lemma 8.19shows that Algorithm 1 computes labels, which result in strong decision sets that are compatible with the needs of Theorem 7.12. Strong consensus in the WTL model can hence be solved by means of our universal algorithm.

Theorem 8.18 (Strong decision sets for WTL algorithm). *The set* $\Sigma(p) = \{\sigma \mid \Delta(\sigma) = p\}$ *is open in the uniform topology, and so is the strong decision set* $\Sigma_v = \{\sigma \mid (\Delta(\sigma) = p) \wedge (I_p = v)\}$.

Proof. We show that, if σ is assigned to the partition set $\Sigma(p)$, then $B_{2^{-(i+D(\sigma))}}(\sigma) \subseteq \Sigma(p)$, where i is the smallest round where $\mathsf{mature}(\sigma, i) = \mathsf{true}$ and $D(\sigma)$ is the maximum number of rounds required for a minority indistinguishability in σ_i to go away ($D(\sigma) = r - r_0$ in the notation of Lemma 8.16), which implies openness of $\Sigma(p)$. Note that the corresponding property obviously also holds for the decision set $\Sigma_v = \{\sigma \mid (\Delta(\sigma) = p) \wedge (I_p = v)\}$.

First of all, in Algorithm 1, $\Delta(\sigma|_i)$ gets initialized to $\emptyset$ in line 1 and assigns a label $\neq \emptyset$ at the latest when $\mathsf{mature}(\sigma, i) = \mathsf{true}$. Once assigned, this value is never modified again as each assignment, except the one in line 3, may only be performed if the label was still $\emptyset$.

For an unlabeled prefix $\sigma|_i$ that is indistinguishable to a mature labeled prefix $\rho|_i$, there are two possibilities: Either, its indistinguishability is a majority one, in which case $\sigma|_i$ gets its label from $\rho|_i$ in line 6, or else the minority indistinguishability will go away within $D(\sigma)$ rounds. It thus suffices to show that if a label $\Delta(\rho|_r) \leftarrow \{p\}$ is assigned to a round r prefix $\rho|_r$, then every majority-indistinguishable prefix $\sigma|_r \sim_{\geq n-f} \rho|_r$ has either $\Delta(\rho|_r) = \Delta(\sigma|_r)$ or $\Delta(\sigma|_r) = \emptyset$.

We prove this by induction on $r = 0, 1, \ldots$. The base for $r = 0$ follows directly from line 1. For the step from $r-1$ to r, assume by hypothesis that, for all round $r-1$ prefixes that already had $\{p\}$ assigned, all their majority-indistinguishable prefixes have label $\{p\}$ or $\emptyset$. For the purpose of deriving a contradiction, suppose that a label $\Delta(\rho|_r) \neq \emptyset$ is assigned to a round r-prefix $\rho|_r$ in iteration r and there exists some $\sigma|_r$ with $\sigma|_r \sim_{\geq n-f} \rho|_r$ and $\emptyset \neq \Delta(\sigma|_r) \neq \Delta(\rho|_r)$. Let S be the set of involved processes, i.e., $\sigma|_r \sim_s \rho|_r$ for $s \in S$ with $|S| \geq n - f$.

We need to distinguish all the different ways of assigning labels to $\rho|_r$.

Suppose $\sigma|_r$ nor $\rho|_r$ get their labels in round r, but not in line 6. Since both $\mathsf{mature}(\sigma, r) = \mathsf{true}$ and $\mathsf{mature}(\rho, r) = \mathsf{true}$, Lemma 8.12.(iii) in conjunction with Lemma 8.13 reveals that $p_\sigma = p_\rho$ since $\sigma|_r \sim_{\geq n-f} \rho|_r$. In all cases except for the one where both $\rho|_r$ and $\sigma|_r$ get their labels in line 9, we immediately get a contradiction since $\Delta(\rho|_r) = \Delta(\sigma|_r)$ in any case. Finally, if $\rho|_r$ and $\sigma|_r$ get their labels in line 9, there is some $\tau|_r \sim_{\geq n-f} \rho|_r$ with $\mathsf{mature}(\tau, r) = \mathsf{false}$ but $\Delta(\tau|_{r_0}) \neq \emptyset$, where r_0 is such that $\mathsf{oldenough}(\rho, r_0) = \mathsf{true}$, and some $\omega_r \sim_{\geq n-f} \sigma|_r$ with the analogous properties in round r'_0. Let Q' resp. Q'' be the sets of at least $n-f$ processes involved in $\tau|_r \sim_{\geq n-f} \rho|_r$ resp. $\omega_r \sim_{\geq n-f} \sigma|_r$. Since $\mathsf{mature}(\rho, r) = \mathsf{true}$, Lemma 8.17 implies $\rho|_{r_0} \sim_{C(\rho|_r)} \sigma|_{r_0} \sim_{C(\rho|_r)} \tau|_{r_0}$ and also $\rho|_{r'_0} \sim_{C(\rho|_r)} \sigma|_{r'_0} \sim_{C(\rho|_r)} \omega_{r_0}$, which establishes $\omega_{r_0} \sim_{C(\rho|_r)} \tau|_{r_0}$. Since, by the induction hypothesis, $\Delta(\omega_{r_0}) = \Delta(\tau|_{r_0})$, we again end up with $\Delta(\rho|_r) = \Delta(\sigma|_r)$, which provides the required contradiction.

However, we also need to make sure that inconsistent labels cannot be assigned in line 6 and any of the other lines, possibly in different rounds. For a contradiction, we assume a "generic" setting that can be fit to all cases: We assume that $\sigma|_{r'}$ got its label $\Delta(\sigma|_{r'}) = \Delta(\tau|_{r'}) \neq \emptyset$ assigned in iteration $r' \leq r$ in line 6 or line 12, since there was some already labeled $\tau|_{r'} \sim_{\geq n-f} \sigma|_{r'}$ with $\mathsf{mature}(\tau, r') = \mathsf{true}$ but $\mathsf{mature}(\sigma, r') = \mathsf{false}$. Moreover, we assume that $\rho|_r$ gets assigned its label $\Delta(\sigma|_r) \neq \Delta(\rho|_r) = \Delta(\omega_r) \neq \emptyset$ in iteration $r \geq r' > r_{\mathrm{stab},\tau} + \Theta$ also in line 6 or in line 12, since there is some already labeled $\omega_r \sim_{\geq n-f} \rho|_r$ with $\mathsf{mature}(\omega, r) = \mathsf{true}$ but $\mathsf{mature}(\rho, r) = \mathsf{false}$. Note carefully that we can rule out the possibility that there are two different, say, $\sigma|_{r'}$ and $\sigma'|_{r'}$, with inconsistent labels, which both match the condition of line 6 or line 12: This is prohibited by the induction hypothesis, except in the case of $r' = r$, where the above generic scenario applies.

To also cover the cases where $\rho|_r$ gets it label assigned in the other lines, we can set $\rho|_r = \omega_r$ in our considerations below. Note that the induction hypothesis again rules out the possibility that there are two different, say, $\sigma|_{r_0}$ and $\sigma'|_{r_0}$, with inconsistent labels, which both match the condition of line 9 here, since $r_0 < r$.

Let $Q' \subseteq C(\tau|_{r'})$ be the set of at least $n-f$ processes causing $\tau|_{r'} \sim_{\geq n-f} \sigma|_{r'}$, and $Q'' \subseteq C(\omega_r)$ be the set of at least $n-f$ non-faulty processes causing $\omega_r \sim_{\geq n-f} \rho|_r$. Since $\mathsf{mature}(\tau, r') = \mathsf{true}$ and $\mathsf{mature}(\omega, r) = \mathsf{true}$, Lemma 8.17 implies

$$\tau|_{r'_0} \sim_{C(\tau|_{r'})} \sigma|_{r'_0} \sim_{C(\tau|_{r'})} \rho|_{r'_0}$$
$$\sigma|_{r_0} \sim_{C(\omega_r)} \rho|_{r_0} \sim_{C(\omega_r)} \omega_{r_0}$$

We first consider the case $r'_0 \leq r_0 \leq r'$: Since $Q' \subseteq C(\tau|_{r'})$, $\tau|_{r'} \sim_{Q'} \sigma|_{r'}$ also implies $\tau|_{r_0} \sim_{Q'} \sigma|_{r_0}$. As $\mathsf{oldenough}(\tau, r'_0) = \mathsf{true}$, Lemma 8.12.(ii) also ensures $\mathsf{oldenough}(\tau, r_0) = \mathsf{true}$. Moreover, since obviously $Q' \cap C(\omega_r) \neq \emptyset$ as well, we finally observe that actually $\tau|_{r_0} \sim_{Q' \cap C(\omega_r)} \omega_{r_0}$. By Lemma 8.13, we hence find that $p_\omega = p_\tau$. Now there are two possibilities: If actually $\tau|_{r_0} \sim_{\geq n-f} \omega_{r_0}$ holds, line 9 implies that $\Delta(\omega_r) = \Delta(\tau|_{r_0})$. Otherwise, every process will eventually be able to distinguish $\tau|_r$ and ω_r and, hence, $\rho|_r$ and $\sigma|_r$ by

Lemma 8.16. Both are contradictions to one of our assumptions $\Delta(\omega_r) \neq \Delta(\tau|_{r_0})$ and $\rho|_r \sim_{\geq n-f} \sigma|_r$.

To handle the case $r'_0 > r_0$, we note that we can repeat exactly the same arguments as above if we exchange the roles of ω_r and $\tau|_{r'}$ and $\sigma|_r$ and $\rho|_r$. In the only possible case of $r_0 \leq r'_0 \leq r$, since $Q'' \subseteq C(\omega_r)$, $\omega_r \sim_{Q''} \rho|_r$ also implies $\omega_{r'_0} \sim_{Q''} \rho|_{r'_0}$. As $\mathsf{oldenough}(\omega, r_0) = \mathsf{true}$, Lemma 8.12.(ii) also ensures $\mathsf{oldenough}(\omega, r'_0) = \mathsf{true}$. Moreover, since obviously $Q'' \cap C(\tau|_{r'}) \neq \emptyset$ as well, we finally observe that actually $\omega_{r'_0} \sim_{Q'' \cap C(\tau|_{r'})} \tau|_{r'_0}$. By Lemma 8.13, we hence find again that $p_\omega = p_\tau$. The same arguments as used in the previous paragraph establish the required contradictions.

In the remaining case $r'_0 \leq r_0$ but $r_0 > r'$, we have the situation where $\sigma|_{r'}$ has already assigned its label *before* round r_0, where $\mathsf{oldenough}(\rho, r_0) = \mathsf{true}$. In general, every process may be able to distinguish ρ and σ (not to speak of τ and ω) after r_0, and usually $p_\tau \neq p_\omega$, so nothing would prevent $\Delta(\sigma|_r) \neq \Delta(\rho|_r)$ if the labeling algorithm would not have taken special care, namely, in line 9: Rather than just assigning $\Delta(\rho|_r) = \{p_\omega\}$, it uses the label of $\sigma|_{r_0}$ and therefore trivially avoids inconsistent labels. Note carefully that doing this is well-defined: If there were two different eligible $\sigma|_{r_0}$ and $\sigma'|_{r_0}$ available in line 9, (14) reveals that $\sigma|_{r_0} \sim_{\geq n-f} \sigma'|_{r_0}$, such that their labels must be the same by the induction hypothesis.

This completes the proof of our theorem. □

The following Lemma 8.19 finally confirms that a non-empty label p assigned to some prefix $\sigma|_r$ is indeed a broadcaster:

Lemma 8.19. *If $\Delta(\sigma|_r) = \{p\}$ is computed by Algorithm 1, then $(p, 0, I_p(\sigma))$ is contained in the view $V_q(\sigma|_r)$ of every process $q \in \Pi \setminus F(\sigma|_r)$ that has not crashed in $\sigma|_r$.*

Proof. We distinguish the two essential cases where $\rho|_r \in \Sigma_p$ can get its label $\{p\}$: If $\Delta(\rho|_r)$ was assigned via line 14, the dominant p_ρ must indeed have reached all correct processes in the system according to Definition 8.11 of $\mathsf{oldenough}(\rho, r_0)$, which is incorporated in $\mathsf{mature}(\rho, r)$. In all other cases, $\Delta(\rho|_r)$ was assigned since there is some $\sigma|_{r'} \sim_{s'} \rho|_{r'}$, $r' \leq r$, with at least $\mathsf{oldenough}(\sigma, r') = \mathsf{true}$. By the same argument as before, the dominant p_σ must have reached every correct process in $\sigma|_{r'}$ already. As $\mathsf{minheardof}_{s'}(\sigma, r') \geq r_{\mathrm{stab},\sigma} + \Theta$ according to the definition of $\mathsf{oldenough}(\sigma, r')$ implies also $\mathsf{minheardof}_{s'}(\rho, r') \geq r_{\mathrm{stab},\sigma} + \Theta$ since $\sigma|_{r'} \sim_{s'} \rho|_{r'}$, it follows that p_σ has also reached all correct processes in $\rho|_{r'}$ already. □

9 CONCLUSIONS

We provided a complete characterization of both uniform and non-uniform deterministic consensus solvability in distributed systems with benign process and communication failures using point-set topology. Consensus can only be solved when the space of admissible executions can be partitioned into disjoint decision sets that are both closed and open in our topologies. We also showed that 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. The utility and wide applicability of our characterization was demonstrated by applying it to several different distributed computing models.

Part of our future work will be devoted to a generalization of our topological framework to other decision problems. Since the initial publication of our results, this generalized study has been started by Attiya, Castañeda, and Nowak [5]. Another very interesting area of future research is to study the homology of non-compact message adversaries, i.e., a more detailed topological structure of the space of admissible executions.

ACKNOWLEDGMENTS

We gratefully acknowledge the suggestions of the reviewers, which stimulated the inclusion of several additional results and pointed out many ways to improve our paper.

REFERENCES

[1] Yehuda Afek, Hagit Attiya, Danny Dolev, Eli Gafni, Michael Merritt, and Nir Shavit. 1993. Atomic snapshots of shared memory. *J. ACM* 40, 4 (1993), 873—-890. https://doi.org/10.1145/153724.153741

[2] Yehuda Afek and Eli Gafni. 2013. Asynchrony from Synchrony. In *Proceedings of the 14th International Conference on Distributed Computing and Networking (ICDCN 2013)*, Davide Frey, Michel Raynal, Saswati Sarkar, Rundrapatna K. Shyamasundar, and Prasun Sinha (Eds.). Lecture Notes in Computer Science, Vol. 7730. Springer, Heidelberg, 225–239. https://doi.org/10.1007/978-3-642-35668-1_16

[3] Marcos K. Aguilera, Carole Delporte-Gallet, Hugues Fauconnier, and Sam Toueg. 2004. Communication-efficient Leader Election and Consensus with Limited Link Synchrony. In *Proceedings of the 23th ACM Symposium on Principles of Distributed Computing (PODC 2004)*, Shay Kutten (Ed.). ACM Press, New York, 328–337. https://doi.org/10.1145/1011767.1011816

[4] Bowen Alpern and Fred B. Schneider. 1985. Defining liveness. *Inform. Process. Lett.* 21, 4 (1985), 181–185. https://doi.org/10.1016/0020-0190(85)90056-0

[5] Hagit Attiya, Armando Castañeda, and Thomas Nowak. 2023. Topological Characterization of Task Solvability in General Models of Computation. In *Proceedings of the 37th International Symposium on Distributed Computing (DISC 2023)*, Rotem Oshman (Ed.). Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl, 24:1–24:23. https://doi.org/10.4230/LIPIcs.DISC.2023.5

[6] Hagit Attiya, Armando Castañeda, and Sergio Rajsbaum. 2020. Locally Solvable Tasks and the Limitations of Valency Arguments. In *Proceedings of the 24th International Conference on Principles of Distributed Systems (OPODIS 2020)*, Quentin Bramas, Rotem Oshman, and Paolo Romano (Eds.). Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 18:1–18:16. https://doi.org/10.4230/LIPIcs.OPODIS.2020.18

[7] Hagit Attiya and Jennifer Welch. 2004. *Distributed Computing* (2nd ed.). John Wiley & Sons, Hoboken.

[8] Ido Ben-Zvi and Yoram Moses. 2014. Beyond Lamport's happened-before: on time bounds and the ordering of events in distributed systems. *J. ACM* 61, 2 (2014), 13:1–13:26. https://doi.org/10.1145/2542181

[9] Martin Biely and Peter Robinson. 2019. On the Hardness of the Strongly Dependent Decision Problem. In *Proceedings of the 20th International Conference on Distributed Computing and Networking (ICDCN 2019)*. ACM Press, New York, 120–123. https://doi.org/10.1145/3288599.3288614

[10] Martin Biely, Peter Robinson, Ulrich Schmid, Manfred Schwarz, and Kyrill Winkler. 2018. Gracefully degrading consensus and k-set agreement in directed dynamic networks. *Theor. Comput. Sci.* 726 (2018), 41–77. https://doi.org/10.1016/j.tcs.2018.02.019

[11] Armando Castañeda, Pierre Fraigniaud, Ami Paz, Sergio Rajsbaum, Matthieu Roy, and Corentin Travers. 2019. Synchronous t-Resilient Consensus in Arbitrary Graphs. In *Proceedings of the 21st Symposium on Stabilization, Safety, and Security of Distributed Systems (SSS 2019)*, Mohsen Ghaffari, Mikhail Nesterenko, Sébastien Tixeuil, Sara Tucci, and Yukiko Yamauchi (Eds.). Springer, Heidelberg, 53–68. https://doi.org/10.1007/978-3-030-34992-9_5

[12] Tushar Deepak Chandra and Sam Toueg. 1996. Unreliable failure detectors for reliable distributed systems. *J. ACM* 43, 2 (March 1996), 225–267. https://doi.org/10.1145/226643.226647

[13] Bernadette Charron-Bost and André Schiper. 2009. The Heard-Of model: computing in distributed systems with benign faults. *Distrib. Comput.* 22, 1 (April 2009), 49–71. https://doi.org/10.1007/s00446-009-0084-6

[14] Étienne Coulouma, Emmanuel Godard, and Joseph G. Peters. 2015. A characterization of oblivious message adversaries for which Consensus is solvable. *Theor. Comput. Sci.* 584 (2015), 80–90. https://doi.org/10.1016/j.tcs.2015.01.024

[15] Danny Dolev, Cynthia Dwork, and Larry Stockmeyer. 1987. On the minimal synchronism needed for distributed consensus. *J. ACM* 34, 1 (1987), 77–97. https://doi.org/10.1145/7531.7533

[16] Cynthia Dwork, Nancy Lynch, and Larry Stockmeyer. 1988. Consensus in the presence of partial synchrony. *J. ACM* 35, 2 (1988), 288–323. https://doi.org/10.1145/42282.42283

[17] Tristan Fevat and Emmanuel Godard. 2011. Minimal Obstructions for the Coordinated Attack Problem and Beyond. In *Proceedings of the 25th IEEE International Symposium on Parallel and Distributed Processing, (IPDPS 2011)*. 1001–1011. https://doi.org/10.1109/IPDPS.2011.96

[18] Faith Fich and Eric Ruppert. 2003. Hundreds of impossibility results for distributed computing. *Distributed Computing* 16 (2003), 121–163. https://doi.org/10.1007/s00446-003-0091-y
[19] Michael J. Fischer, Nancy A. Lynch, and Michael S. Paterson. 1985. Impossibility of distributed consensus with one faulty process. *J. ACM* 32, 2 (1985), 374–382. https://doi.org/10.1145/3149.214121
[20] Eli Gafni, Petr Kuznetsov, and Ciprian Manolescu. 2014. A Generalized Asynchronous Computability Theorem. In *Proceedings of the 33rd ACM Symposium on Principles of Distributed Computing (PODC 2014)*, Shlomi Dolev (Ed.). ACM Press, New York, 222—231. https://doi.org/10.1145/2611462.2611477
[21] Hugo Rincon Galeana, Sergio Rajsbaum, and Ulrich Schmid. 2022. Continuous tasks and the asynchronous computability theorem. In *Proceedings of the 13th Innovations in Theoretical Computer Science Conference (ITCS 2022)*, Mark Braverman (Ed.). Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 73:1–73:27. https://doi.org/10.4230/LIPIcs.ITCS.2022.73
[22] Hugo Rincon Galeana, Ulrich Schmid, Kyrill Winkler, Ami Paz, and Stefan Schmid. 2023. Topological Characterization of Consensus Solvability in Directed Dynamic Networks. http://arxiv.org/abs/2304.02316
[23] Emmanuel Godard and Eloi Perdereau. 2020. Back to the Coordinated Attack Problem. *Math. Struct. Comput. Sci.* 30, 10 (2020), 1089–1113. https://doi.org/10.1017/S0960129521000037
[24] Maurice Herlihy, Dmitry N. Kozlov, and Sergio Rajsbaum. 2013. *Distributed Computing Through Combinatorial Topology*. Morgan Kaufmann. https://store.elsevier.com/product.jsp?isbn=9780124045781
[25] Maurice Herlihy and Nir Shavit. 1999. The topological structure of asynchronous computability. *J. ACM* 46, 6 (1999), 858–923. https://doi.org/10.1145/331524.331529
[26] Martin Hutle, Dahlia Malkhi, Ulrich Schmid, and Lidong Zhou. 2009. Chasing the weakest system model for implementing Omega and consensus. *IEEE T. Depend. Secure* 6, 4 (2009), 269–281. https://doi.org/10.1109/TDSC.2008.24
[27] Fabian Kuhn and Rotem Oshman. 2011. Dynamic networks: models and algorithms. *SIGACT News* 42(1) (2011), 82–96. https://doi.org/10.1145/1959045.1959064
[28] Petr Kuznetsov, Thibault Rieutord, and Yuan He. 2018. An Asynchronous Computability Theorem for Fair Adversaries. In *Proceedings of the 37th ACM Symposium on Principles of Distributed Computing (PODC 2018)*, Idit Keidar (Ed.). ACM Press, New York, 387–396. https://doi.org/10.1145/3212734.3212765
[29] Leslie Lamport. 1978. Time, clocks, and the ordering of events in a distributed system. *Commun. ACM* 21, 7 (1978), 558–565. https://doi.org/10.1145/359545.359563
[30] Leslie Lamport, Robert Shostak, and Marshall Pease. 1982. The Byzantine generals problem. *ACM T. Progr. Lang. Sys.* 4, 3 (1982), 382–401. https://doi.org/10.1145/357172.357176
[31] Ronit Lubitch and Shlomo Moran. 1995. Closed Schedulers: A Novel Technique for Analyzing Asynchronous Protocols. *Distrib. Comput.* 8, 4 (June 1995), 203–210. https://doi.org/10.1007/BF02242738
[32] Friedemann Mattern. 1989. Virtual time and global states of distributed systems. In *Proceedings of the International Workshop on Parallel and Distributed Algorithms*, Michel Cosnard, Yves Rober, Patrice Quinton, and Michel Raynal (Eds.). North Holland, Amsterdam, 215–226.
[33] Yoram Moses and Sergio Rajsbaum. 2002. A layered analysis of consensus. *SIAM J. Comput.* 31, 4 (2002), 989–1021. https://doi.org/10.1137/S0097539799364006
[34] Achour Mostefaoui, Sergio Rajsbaum, and Michel Raynal. 2003. Conditions on input vectors for consensus solvability in asynchronous distributed systems. *J. ACM* 50, 6 (2003), 922—-954. https://doi.org/10.1145/950620.950624
[35] Achour Mostéfaoui and Michel Raynal. 2001. Leader-based consensus. *Parallel Process. Lett.* 11, 1 (2001), 95–107. https://doi.org/10.1142/S0129626401000452
[36] James Munkres. 2000. *Topology* (2nd ed.). Prentice Hall, Hoboken.
[37] Thomas Nowak. 2010. *Topology in Distributed Computing*. Master's thesis. Embedded Computing Systems Group, Technische Universität Wien.
[38] Thomas Nowak, Ulrich Schmid, and Kyrill Winkler. 2019. Topological Characterization of Consensus under General Message Adversaries. In *Proceedings of the 28th ACM Symposium on Principles of Distributed Computing (PODC 2019)*, Faith Ellen (Ed.). ACM Press, New York, 218–227. https://doi.org/10.1145/3293611.3331624
[39] P. R. Parvedy and M. Raynal. 2003. Uniform agreement despite process omission failures. In *Proceedings of the 17th International Parallel and Distributed Processing Symposium (IPDPS 2003)*, Jack Dongarra (Ed.). IEEE Press, New York, 22–26. https://doi.org/10.1109/IPDPS.2003.1213388
[40] Kenneth J. Perry and Sam Toueg. 1986. Distributed agreement in the presence of processor and communication faults. *IEEE T. Software Eng.* SE-12, 3 (1986), 477–482. https://doi.org/10.1109/TSE.

1986.6312888
[41] Daniel Pfleger. 2018. *Knowledge and Communication Complexity*. Master's thesis. Embedded Computing Systems Group, Technische Universität Wien.
[42] Michel Raynal and Julien Stainer. 2013. Synchrony Weakened by Message Adversaries vs Asynchrony Restricted by Failure Detectors. In *Proceedings of the 32nd ACM Symposium on Principles of Distributed Computing (PODC 2013)*, Gadi Taubenfeld (Ed.). ACM Press, New York, 166–175. https://doi.org/10.1145/2484239.2484249
[43] Peter Robinson and Ulrich Schmid. 2011. The Asynchronous Bounded-Cycle model. *Theor. Comput. Sci.* 412, 40 (2011), 5580–5601. https://doi.org/10.1016/j.tcs.2010.08.001
[44] Nicola Santoro and Peter Widmayer. 1989. Time is Not a Healer. In *Proceedings of the 6th Annual Symposium on Theoretical Aspects of Computer Science (STACS 1989)*. Springer, Heidelberg, 304–313.
[45] Ulrich Schmid, Bettina Weiss, and Idit Keidar. 2009. Impossibility results and lower bounds for consensus under link failures. *SIAM J. Comput.* 38, 5 (2009), 1912–1951. https://doi.org/10.1137/S009753970443999X
[46] Kyrill Winkler, Ami Paz, Hugo Rincon Galeana, Stefan Schmid, and Ulrich Schmid. 2023. The Time Complexity of Consensus Under Oblivious Message Adversaries. In *Proceedings of the 14th Innovations in Theoretical Computer Science Conference (ITCS 2023)*, Yael Tauman Kalai (Ed.). Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl, 100:1–100:28. https://doi.org/10.4230/LIPIcs.ITCS.2023.100
[47] Kyrill Winkler, Ulrich Schmid, and Yoram Moses. 2019. A Characterization of Consensus Solvability for Closed Message Adversaries. In *Proceedings of the 23rd International Conference on Principles of Distributed Systems (OPODIS 2019)*. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, Dagstuhl, 17:1–17:16. https://doi.org/10.4230/LIPIcs.OPODIS.2019.17
[48] Kyrill Winkler, Ulrich Schmid, and Thomas Nowak. 2021. Valency-Based Consensus Under Message Adversaries Without Limit-Closure. In *Prceedings of the 23rd International Symposium on Fundamentals of Computation Theory (FCT 2021)*, Evripidis Bampis and Aris Pagourtzis (Eds.). Springer, Heidelberg, 457–474. https://doi.org/10.1007/978-3-030-86593-1_32
[49] Kyrill Winkler, Manfred Schwarz, and Ulrich Schmid. 2019. Consensus in directed dynamic networks with short-lived stability. *Distrib. Comput.* 32, 5 (2019), 443–458. https://doi.org/10.1007/s00446-019-00348-0

A PROCESS-TIME GRAPHS

In the main body of our paper, we have formalized our topological results in terms of admissible executions in the generic system model introduced in Section 3. In this section, we will show that they also hold a topological space consisting of other objects, namely, *process-time graphs [8]*. In a nutshell, a process-time graph describes the process scheduling and all communication occurring in a run, along with the set of initial values.

Actually, since we consider deterministic algorithms only, a process-time graph corresponds to a *unique* execution (and vice versa). This equivalence, which actually results from a *transition function* that is continuous in all our topologies (see Lemma A.2), will eventually allow us to use our topological reasoning in either space alike.

In order to define process-time graphs as generic as possible, we will resort to an intermediate *operational system model* that is essentially equivalent to the very flexible general system model from Moses and Rajsbaum [33]. Crucially, it will also instantiate the weak clock functions $\chi_p(C^t)$ stipulated in our generic model in Section 3, which must satisfy $\chi_p(C^t) \leq t$ in every admissible execution $(C^t)_{t\geq 0} \in \Sigma$. Since t represents some global notion of time here (called *global real time* in the sequel), ensuring this property is sometimes not trivial. More concretely, whereas t is inherently known at every process in the case of lock-step synchronous systems like dynamic networks under message adversaries [49], for example, this is not the case for purely asynchronous systems [19].

A.1 Basic operational system model

Following Moses and Rajsbaum [33], we consider message passing or shared memory distributed systems made up of a set Π of $n \geq 2$ processes. We stipulate a global discrete clock with values taken from $\mathbb{N}_0 = \mathbb{N} \cup \{0\}$, which represents global real time in multiples of some arbitrary unit time. Depending on the particular distributed computing model, this global clock may or may not be accessible to the processes.

Processes are modeled as communicating state machines that encode a deterministic distributed algorithm (protocol) $\mathcal{P}$. At every real time time $t \in \mathbb{N}_0$, process p is in some *local state* $L_p^t \in \mathcal{L}_p \cup \{\bot_p\}$, where $\bot_p \notin \mathcal{L}_p$ is a special state representing that process p has failed.[4] Local state transitions of p are caused by local *actions* taken from the set ACT_p, which may be internal bookkeeping operations and/or the initiation of shared memory operations resp. of sending messages; their exact semantics may vary from model to model. Note that a single action may consist of finitely many non-zero time operations, which are initiated simultaneously but may complete at different times. The deterministic protocol $\mathcal{P}_p : \mathcal{L}_p \to \mathsf{ACT}_p$, representing p's part in $\mathcal{P}$, is a function that specifies the local action p is ready to perform when in state $L_p \in \mathcal{L}_p$. We do not restrict the actions p can perform when in state $\bot_p$.

In addition, there is an additional non-deterministic state machine called the *environment* ϵ, which represents the adversary that is responsible for actions outside the sphere of control of the processes' protocols. It controls things like the completion of shared memory operations initiated earlier resp. the delivery of previously sent messages, the occurrence of process and communication failures, and (optionally) the occurrence of *external environment events* that can be used for modeling oracle inputs like failure detectors [12]. Let act_ϵ be the set of all possible combinations of such *environment actions* (also called *events* for conciseness later on). We assume that the environment keeps track of pending shared-memory operations resp. sent messages in its *environment state* $L_\epsilon \in \mathcal{L}_\epsilon$. The environment is also in charge of process *scheduling*, i.e., determines when a process performs a state transition, which will be referred to as *taking a step*. Formally, we assume that the set ACT_ϵ of all possible environment actions consists of all pairs (Sched, e), made up of the set of processes $\mathsf{Sched} \subseteq \Pi$ that take a step and some $e \in \mathsf{act}_\epsilon$ (which may both be empty as well). The non-deterministic *environment protocol* $\mathcal{P}_\epsilon \subseteq \mathcal{G} \times (\mathsf{ACT}_\epsilon \times \mathcal{L}_\epsilon)$ is an arbitrary relation that, given the current global state $G \in \mathcal{G}$ (defined below, which also contains the current environment state $L_\epsilon \in \mathcal{L}_\epsilon$), chooses the next environment action $E = (\mathsf{Sched}, e) \in \mathsf{ACT}_\epsilon$ and the successor environment state $L'_\epsilon \in \mathcal{L}_\epsilon$. Note carefully that we assume that only E is actually chosen non-deterministically by $\mathcal{P}_\epsilon$, whereas L'_ϵ is determined by a transition function $\tau_\epsilon : \mathcal{G} \times \mathsf{ACT}_\epsilon \to \mathcal{L}_\epsilon$ according to $L'_\epsilon = \tau_\epsilon(G, E)$.

Finally, a *global state* of our system (simply called state) is an element of $\mathcal{G} = \mathcal{L}_\epsilon \times \mathcal{L}_1 \times \cdots \times \mathcal{L}_n$. Given a global state $G \in \mathcal{G}$, G_i denotes the local state of process i in G, and G_ϵ denotes the state of the environment in G. Recall that it is G_ϵ that keeps track of in-transit

[4]This failed state $\bot_p$ is the only essential difference to the model of Moses and Rajsbaum [33], where faults are implicitly caused by a deviation from the protocol. This assumption makes sense for constructing "permutation layers", for example, where it is not the environment that crashes a process at will, but rather the layer construction, which implies that some process takes only finitely many steps. Such a process just remains in the local state reached after its last computing step. In our setting, however, the fault state of all processes is solely controlled by the omniscient environment. Hence, we can safely use a failed state $\bot_p$ to gain simplicity without losing expressive power.

(i.e., just sent) messages, pending shared-memory operations etc.[5] We also write $G = (G_\epsilon, C)$, where the vector of the local states $C = (C_1, \ldots, C_n) = (G_1, \ldots, G_n)$ of all the processes is called *configuration*. Given C, the component C_i denotes the local state of process i in C, and the set of all possible configurations is denoted as $\mathcal{C}$. Note carefully that there may be global configurations $G \neq G'$ where the corresponding configurations satisfy $C = C'$, e.g., in the case of different in-transit messages.

A *joint action* is a pair (E, A), where $E = (\mathsf{Sched}, e) \in \mathsf{ACT}_\epsilon$, and A is a vector with index set Sched such that $A_p \in \mathsf{ACT}_p$ for $p \in \mathsf{Sched}$. When the joint action E is applied to global state G where process p is in local state G_p, then $A_p = \mathcal{P}_p(G_p)$ is the action prescribed by p's protocol. Note that some environment actions, like message receptions at process p require $p \in \mathsf{Sched}$, i.e., "wake-up" the process. For example, a joint action (E, A) that causes p to send a message m to q and process r to receive a message m' sent to it by process s earlier, typically works as follows: (i) p is caused to take a step, where its protocol $\mathcal{P}_p$ initiates the sending of m; (ii) the environment adds m to the send buffer of the communication channel from p to q (maintained in the environment state L_ϵ); (iii) the environment moves m' from the send buffer of the communication channel from s to r (maintained in the environment state L_ϵ) to the receive buffer (maintained in the local state of r), and (iv) causes r to take a step. It follows that the local state L_r of process r reflects the content of message m' immediately after the step scheduled along with the message reception.

With ACT denoting the set of all possible joint actions, the *transition function* $\tau : \mathcal{G} \times \mathsf{ACT} \to \mathcal{G}$ describes the evolution of the global state G after application of the joint action (E, A), which results in the successor state $G' = \tau(G, (E, A))$. A *run* of $\mathcal{P}$ is an infinite sequence of global states $G^0, G^1, G^2, \ldots$ generated by an infinite sequence of joint actions. In order to guarantee a stable global state at integer times, we assume for simplicity that the joint actions occur atomically and instantaneously at times $0.5, 1.5, 2.5, \ldots$, i.e., that $G^{t+1} = \tau(G^t, (E^{t.5}, A^{t.5}))$. G^0 is the *initial state* of the run, taken from the set of possible initial states $\mathcal{G}^0$. Finally, Ψ denotes the subset of all *admissible runs* of our system. Ψ is typically used for enforcing liveness conditions like "every message sent to a correct process is eventually delivered" or "every correct process takes infinitely many steps".

Unlike Moses and Rajsbaum [33], we handle process failures explicitly in the state of the processes, i.e., via the transition function: If some joint action $(E^{t.5}, A^{t.5})$ contains $E^{t.5} = (\mathsf{Sched}, e)$, where e requests some process p to fail, this will force $G_p^{t+1} = \bot_p$ in the successor state $G^{t+1} = \tau(G^t, (E^{t.5}, A^{t.5}))$, irrespective of any other operations in e (like the delivery of a message) that would otherwise affect p. All process failures are persistent, that is, we require that all subsequent environment actions $E^{t'.5}$ for $t' \geq t$ also request p to fail. As a convention, we consider every $E^{t'.5}$ where p fails as p taking a step as well. Depending on the type of process failure, failing may cause p to stop its protocol-compliant internal computations, to drop all incoming messages, and/or to stop sending further messages. In the case of crash failures, for example, the process may send a subset of the outgoing messages demanded by $\mathcal{P}_p$ in the very first failing step and does not perform any protocol-compliant actions in future steps. A send omission-faulty process does the same, except that it may send protocol-compliant messages to some processes also in future steps. A receive omission-faulty process may omit to process some of its received messages in every step where it fails, but sends protocol-compliant messages to every receiver. A general omission-faulty process combines the possible behaviors of send and receive omissions. Note that message loss can

[5] A different, but equivalent, conceptual model would be to assume that the state of a processor consists of a visible state and, in the case of message passing, message buffers that hold in-transit messages.

also be modeled in a different way in our setting: Rather than attributing an omission failure to the sender or receiver process, it can also be considered a communication failures caused by the environment. The involved sender process p resp. receiver process q continue to act according to its protocol in this case, i.e., would not enter the fault state $\bot_p$ resp. $\bot_q$ here.

Since we only consider deterministic protocols, a run $G^0, G^1, G^2, \dots$ is uniquely determined by the initial configuration C^0 and the sequence of tuples $(L_\epsilon^0, E^{0.5}), (L_\epsilon^1, E^{1.5}), \dots$ consisting of tuples $(L_\epsilon^t, E^{t.5})$ of environment state and environment actions for $t \geq 0$. Let $\mathcal{G}^\omega$ resp. $\mathcal{C}^\omega$ be the set of all infinite *runs* resp. *executions* (configuration sequences), with $\Psi \subseteq \mathcal{G}^\omega$ resp. $\Sigma \subseteq \mathcal{C}^\omega$ denoting the set of *admissible* runs resp. executions that result from admissible environment action sequences $E^{0.5}, E^{1.5}, \dots$; after all, they may be required to satisfy liveness constraints like fairness that cannot be expressed via the transition function.

Our assumptions on the environment protocol, namely, $L_\epsilon^{t+1} = \tau_\epsilon(G^t, E^{t.5})$, actually imply that a run $G^0, G^1, G^2, \dots$, and thus also the corresponding execution $C^0, C^1, C^2, \dots$, is already uniquely determined by the initial state $G^0 = (L_\epsilon^0, C^0)$ and the sequence of chosen environment actions $E^{0.5}, E^{1.5}, \dots$. Since L_ϵ^0 is fixed and the environment actions abstract away almost all of the internal workings of the protocols and their complex internal states, it should be possible to uniquely describe the evolution of a run/execution just by means of the sequence $E^{0.5}, E^{1.5}, \dots$. In the following, we will show that this is indeed the case.

A.2 Implementing global time satisfying the weak clock property

Our topological framework crucially relies on the ability to distinguish/not distinguish two local states α_p^t and β_p^t in two executions α and β at global real time t. Clearly, this is easy for an omniscent observer who knows the corresponding global states and can thus verify that α_p^t and β_p^t arise from the same global time t. Processes cannot do that in asynchronous systems, however, since t is not available to the processes and hence cannot be included in α_p^t and β_p^t. Consequently, two *different* sequences of environment actions (called *events* in the sequel for conciseness) $E_\alpha^{0.5}, E_\alpha^{1.5}, \dots, E_\alpha^{(t-1).5}$ and $E_\beta^{0.5}, E_\beta^{1.5}, \dots, E_\beta^{(t'-1).5}$, applied to the same initial state, may produce the *same* state $\alpha_p^t = \beta_p^{t'}$. This happens when they are causal shuffles of each other, i.e., reorderings of the steps of the processes that are in accordance with the happens-before relation [29]. Hence, the (in)distinguishability of configurations does not necessarily match the (in)distinguishability of the corresponding event sequences.

Whereas our generic system model does not actually require processes to have a common notion of time, it does require that the weak clock functions χ_p do not progress faster than global real time. We will accomplish this in our operational system model by defining some alternative notion of global time that *is* accessible to the processes. Doing this will also rule out the problem spotted above, i.e., ensure that runs (event sequences) uniquely determine executions (configuration sequences).

There are many conceivable ways for defining global time, including the following possibilities:

(i) In the case of lock-step synchronous distributed systems, like dynamic networks under message adversaries [38, 47, 48], nothing needs to be done here since all processes inherently know global real time t.

(ii) In the case of asynchronous systems with a majority of correct processes, the arguably most popular approach for message-passing systems (see e.g. [3, 26, 35]) is the simulation of *asynchronous communication-closed rounds*: Processes organize rounds $r = 1, 2, \dots$ by locally waiting until $n - f$ messages sent in the current round r have been received. These $n - f$ messages are then processed, which defines both the local state at the beginning of the

next round $r+1$ and the message sent to everybody in this next round. Late messages are discarded, and early messages are buffered locally (in the state of the environment) until the appropriate round is reached. The very same approach can also be used in shared-memory systems with immediate snapshots [1], where a process can safely wait until it sees $n-f$ entries in a snapshot. Just using the round numbers as global time, i.e., choosing $t=r$, is all that is needed for defining global time in such a model.

(iii) In models without communication-closed rounds [19, 43], a suitable notion of global time can be derived from other[6] definitions of *consistent cuts* [32]. We will show how this can be done in our operational system model based on Mattern's *vector clocks*. Our construction will exploit the fact that a local state transition of a process happens only when it takes a step in our model: In between the ℓ^{th} and $(\ell+1)^{\text{th}}$ step of any fixed process p, which happens at time $(t_p(\ell)-1).5$ and $(t_p(\ell+1)-1).5$, respectively, only environment actions (external environment events, message deliveries, shared memory completions), if any, can happen, which change the state of the environment but not the local state of p.

We will start out from the sequence of arbitrary *cuts* [32] $IC^0, IC^1, IC^2, \dots$ (indexed by an integer *index* $k \geq 0$) occurring in a given run $G^0, G^1, G^2, \dots$ (which itself is indexed by the global real time t), where the *frontier* IF^k of IC^k is formed by the local states of the processes after they have taken their k^{th} step, i.e., $IF^0 = IC^0 = C^0$ and $IF^k = (G_1^{t_1(k)}, \dots, G_n^{t_n(k)})$ for $k \geq 1$, with $(t_p(k)-1).5$ being the time when process p takes its k^{th} step. Note that the latter is applied to p's state IF_p^{k-1} in the frontier IF^{k-1} of IC^{k-1} and processes all the external environment events and all the messages received/shared memory operations completed since then. Recall the convention that every environment action where process q fails is also considered as q taking a step.

Clearly, except in lock-step synchronous systems, $t_p(k) \neq t_q(k)$, so $IC^0, IC^1, IC^2, \dots$ can be viewed as the result of applying a trivial "synchronic layering" in terms of Moses and Rajsbaum [33]. Unfortunately, though, any IC^k may be an *inconsistent* cut, as messages sent by a fast process p in its $(k+1)^{\text{th}}$ step may have been received by a slow process q by its k^{th} step. IC^k would violate causality in this case, i.e., would not be left-closed w.r.t. Lamport's happens-before relation [29].

Recall that we restricted our attention to consensus algorithms using full-information protocols, where every message sent contains the entire state transition history of the sender. As a consequence, we do not significantly lose applicability of our results by further restricting the protocol and the supported distributed computing models as follows:

(i) In a single state transition of $\mathcal{P}_p$, process p, can
- actually receive all messages delivered to it since its last step,
- initiate the sending of at most one message to every process, resp.,
- initiate at most one single-writer multiple-reader shared memory operation in the shared memory owned by some other process (but no restriction on operations in its own shared memory portion).

(ii) In addition to (optional) external environment events, the environment protocol only provides
- $\mathsf{fail}(q) \in \mathsf{act}_\epsilon$, which tells process q to fail,
- $\mathsf{delv}(q, p, t_k) \in \mathsf{act}_\epsilon$, which identifies the message m to be delivered to process q (for reception in its next step) by the pair (p, t_k), where p is the sending process and $t_k.5$ is the time when the sending of m has been initiated, resp.,

[6]We note that both synchronous and asynchronous communication-closed rounds, as well as the executions $\mathcal{C}^\omega$ defined in our generic system model in Section 3, are of course also sequences of consistent cuts.

- $\mathsf{done}(q, p, t_\ell, t_k) \in \mathsf{act}_\epsilon$, which identifies the completed shared memory operation (to be processed in its next step), in the shared memory owned by p, as the one initiated by process $q \neq p$ in its step at time $t_\ell.5$; in a read-type operation, it will return to q the shared memory content based on p's state at time t_k, with $t_\ell \leq t_k$.

In such a system, given any cut IC^k, it is possible to determine the unique largest *consistent* cut $CC^k \subseteq IC^k$ [32]. By construction, $CC^0 = IC^0$, and the frontier CF^k of CC^k, $k \geq 1$, consists of the local states of all processes $q \in \Pi$ reached by having taken some $\ell(q)^{\text{th}}$ step, $0 \leq \ell(q) \leq k$, with at least one process p having taken its k^{th} step, i.e., $\ell(p) = k$ and thus $CF_p^k = IF_p^k$, and $CF_q^k = IF_q^{\ell(q)}$ with $0 \leq \ell(q) \leq k$ for all processes q. Note carefully that $\ell(q) < k$ happens when, in IC^k, process q receives some message/data initiated at some step $> k$ at or before its own k^{th} step but after its $\ell(q)^{\text{th}}$ step.

Whereas the environment protocol could of course determine all the consistent cuts $CC^0, CC^1, CC^2, \ldots$ based on the corresponding sequence of global configurations, this is typically not the case for the processes (unless in the special case of a synchronous system). However, in distributed systems adhering to the above constraints, processes can obtain this knowledge (that is to say, their local share of a consistent cut) via *vector clocks* [32]. More specifically, it is possible to implement a vector clock $k_p = (k_p^1, \ldots, k_p^n)$ at process p, where k_p^p counts the number of steps taken by p itself so far, and k_p^q, $q \neq p$, gives the number of steps that p knows that q has taken so far. Vector clocks are maintained as follows: Initially, $k_p = (0, \ldots, 0)$, and every message sent resp. every shared memory operation data written by p gets k_p as piggybacked information (after advancing k_p^p). At every local state transition in p's protocol P_p, k_p^p is advanced by 1. Moreover, when a previously received message/previously read data value (containing the originating process q's vector clock value $\hat{k}_q$) is to be processed in the step, k_p is adjusted to the maximum of its previous value and $\hat{k}_q$ component-wise, i.e., $k_p^q = \max\{k_p^q, \hat{k}_q^q\}$ for $q \neq p$. Obviously, all this can be implemented transparently atop of any protocol $\mathcal{P}$ running in the system.

Now, given the sequence of global states $AC^0, AC^1, AC^2, \ldots$ of the processes running the so augmented protocol in some run $G^0, G^1, G^2, \ldots$, there is a well-known algorithm for computing the maximal consistent cut ACC^k for the non-consistent cut AIC^k formed by the frontier AIF^k of the local states of the processes after every process has taken its k^{th} step: Starting from $\ell := k$, process p searches for the sought $\ell(p)$ by checking the vector clock value $k_p(\ell)$ of the state after its own ℓ^{th} step. It stops searching and sets $\ell(p) := \ell$ iff $k_p(\ell)$ is less or equal to $(k, \ldots, k)$ component-wise. The state $AIF_p^{\ell(p)}$ is then process p's contribution in the frontier ACF^k of the consistent cut ACC^k. Clearly, from $ACC^0, ACC^1, ACC^2, \ldots$, the sought sequence of the consistent cuts $CC^0, CC^1, CC^2, \ldots$ can be obtained trivially by discarding all vector clock information. Therefore, even the processes can compute their share, i.e., their local state, in CC^k for every k.

By construction, the sequence of consistent cuts $CC^0, CC^1, CC^2, \ldots$, and hence the sequence of its frontiers $CF^0, CF^1, CF^2, \ldots$, completely describe the evolution of the local states of the processes in a run $G^0, G^1, G^2, \ldots$. In our operational model, we will hence just use the indices k of CC^k as global time for specifying executions: Starting from the initial state CC^0, we consider CC^k as the result of applying *round* $k \geq 1$ to CC^{k-1} (as we did in the case of lock-step rounds).

A.3 Defining process-time graphs

No matter how consistent cuts, i.e., global time, is implemented, from now on, we just overload the notation used so far and denote by C^k the frontier CF^k in the consistent cut at

global time k. So given an infinite execution α, we again denote by α^t the t^{th} configuration (= the consistent cut with index t) in α.

Clearly, by construction, every C^k is *uniquely* determined by C^0 and all the events that cause the steps leading to C^k. In particular, we can define a vector of events E^k, where E^k_p is a set containing all the events that must be applied to C^{k-1}_p in order to arrive at C^k_p. Note carefully that a process p that does not make a step, i.e., is not scheduled in E^k and thus has the same non-$\perp_p$ state in C^{k-1} and C^k, does not have any event $\mathsf{delv}(p,*,*) \in E^k_p$ (resp. $\mathsf{done}(p,*,*) \in E^k_p$) by construction, i.e., $E^k_p = \emptyset$. Otherwise, E^k_p contains a "make a step" event, all (optional) external environment events, and $\mathsf{delv}(p,*,*)$ for all messages that have been sent to p in steps within C^{k-1} and are delivered to p after its previous step but before or at its k^{th} step (resp. $\mathsf{done}(p,*,*,*)$ for all completed shared memory operation initiated by p in steps within CC^{k-1} and completed after p's previous step but before or at its k^{th} step). Note that E^1_p cannot contain any $\mathsf{delv}(p,*,*)$, as no messages have been sent before (resp. no $\mathsf{done}(p,*,*,*)$, as no shared memory operations have been initiated before).

As a consequence of our construction, the mismatch problem spotted at the beginning of Appendix A.2 no longer exists, and we can reason about executions and the corresponding event sequences alike.

Rather than considering C^0 in conjunction with $E^1, \dots, E^k$, however, we will consider the corresponding *process-time graph k-prefix* PTG^k [8] instead, which we will now define. Since we are only interested in consensus algorithms here, we assume that every process has a dedicated initial state for every possible initial value v, taken from a finite input domain $\mathcal{V}$. For every assignment of initial values $x \in \mathcal{V}^n$ to the n processes in the initial configuration C^0, we inductively construct the following sequence of process-time graph prefixes PTG^t:

Definition A.1 (Process-time graph prefixes). For every $k \geq 0$, the *process-time graph k-prefix* PTG^k of a given run is defined as follows:

- The process-time graph 0-prefix PTG^0 contains the nodes $(p, 0, I_p)$ for all processes $p \in \Pi$, with initial value $I_p \in \mathcal{V}$, and no edges.
- The process-time graph 1-prefix PTG^1 contains the nodes $(p, 0, I_p)$ and $(p, 1, f)$ for all processes $p \in \Pi$, where $f = \perp$ if $\mathsf{fail}(p) \in E^1$ (which models the case of an initially dead process [19]), and $f = *$ otherwise, where $*$ is some encoding (e.g., some failure detector output) of the external environment events $\in E^1$. It contains a (local) edge from $(p, 0, I_p)$ to $(p, 1, f)$ and no other edges.
- The process-time graph k-prefix PTG^k, $k \geq 2$, contains PTG^{k-1} and the nodes (p, k, f) for all processes $p \in \Pi \setminus \{q \mid E^k_q = \emptyset\}$, where $f = \perp$ if $\mathsf{fail}(p) \in E^k$, and $f = *$ otherwise. It contains a (local) edge from (p, ℓ, f_ℓ) to (p, k, f) (if the latter node is present at all, i.e., when $E^k_p \neq \emptyset$), where ℓ is maximal among all nodes $(p, *, *)$ in PTG^{k-1}. For message passing systems, it also contains an edge from (p, s, f_s), $1 \leq s < k$, to (q, k, f) iff $\mathsf{delv}(q, p, s) \in E^k$. For shared memory systems, it contains an edge from (p, ℓ, f_ℓ), $1 \leq \ell < k$, to (q, k, f) if and only if $\mathsf{done}(q, p, s, \ell) \in E^k$; this reflects the fact that the returned data originate from p's step ℓ and not from the step s where q has initiated the shared memory operation.

The *round-ℓ process-time graph* PT^ℓ, for $0 \leq \ell \leq k$, which represents the contribution of round ℓ to PTG^k, is defined as (i) $PT^0 = PTG^0$ and the set of vertices $PT^\ell = PTG^\ell \setminus PTG^{\ell-1}$ along with all their incoming edges (which all originate in $PTG^{\ell-1}$).

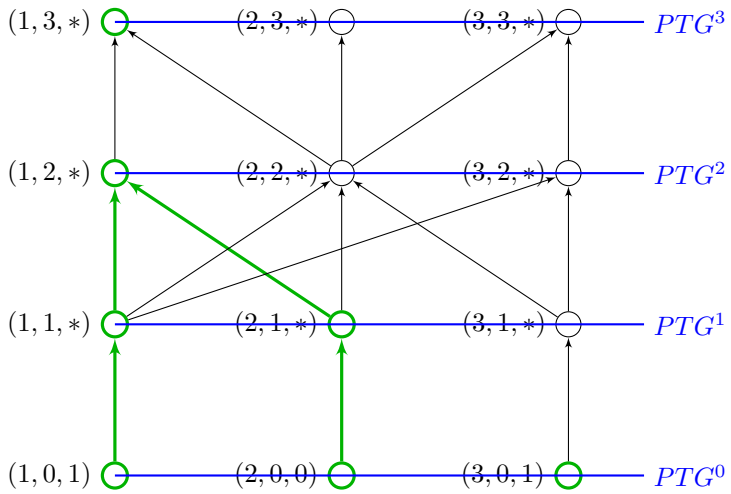

Fig. 5. Example of a process-time graph prefix PTG^3 of a lock-step execution at time $t = 3$, for $n = 3$ processes and initial values $x = (1, 0, 1)$. Process 1's view $V_1(PT^2)$ is highlighted in bold green.

Figure 5 shows an example of a process-time graph prefix occuring in a run with lock-step synchronous or asynchronous rounds. The nodes are horizontally aligned according to global time, progressing along the vertical axis.

Figure 6 shows an example of a process-time graph prefix occuring in a run with processes that do not execute in lock-step rounds and may crash. Nodes are again horizontally aligned according to global time, progressing along the vertical axis. The frontier C^k of the k^{th} consistent cut, reached at the end of round k, is made up of $C_p^k = \{(p, \ell_p(k), *) \in PTG^k \mid 0 \leq \ell_p(k) \leq k \text{ is maximal}\}$. That is, starting from the (possibly inconsistent) cut made up of the nodes $(p, k, *)$ of all processes, one has to go down for process p until the first node is reached where no edge originating in a node with time $> k$ has been received.

Let $\mathcal{PT}^t$ be the set of all possible process-time graph t-prefixes, and $\mathcal{PT}^\omega$ be the set of all posible infinite process-time graphs, for all possible runs of our system. Note carefully that $\mathcal{PT}^t$, as well every set $\mathcal{P}^\ell$ of round-ℓ process-time graphs for finite ℓ, is necessarily *finite* (provided the encoding $(*)$ for external environment events has a finite domain, which we assume). Clearly, $\mathcal{PT}^t$ resp. $\mathcal{PT}^\omega$ can be expressed as a finite resp. infinite sequence $(P^0, \dots, P^t) \in \mathcal{P}^0 \times \mathcal{P}^1 \times \cdots \times \mathcal{P}^t = \mathcal{PT}^t$ resp. $(P^0, P^1, \dots) \in \mathcal{P}^0 \times \mathcal{P}^1 \times \cdots = \mathcal{PT}^\omega$ of round-ℓ process time graphs.[7]

We will denote by $PS \subseteq \mathcal{PT}^\omega$ the set of all admissible process-time graphs in the given model, and by $\Sigma \subseteq \mathcal{C}^\omega$ the corresponding set of admissible executions. Note carefully that process-time graphs are *independent* of the (decision function of the) consensus algorithm, albeit they do depend on the initial values.

Due to the one-to-one of process-time graphs and executions established before, the topological machinery developed in Section 4–Section 5 for $\Sigma \subseteq \mathcal{C}^\omega$ can also be applied to $PS \subseteq \mathcal{PT}^\omega$. Since, in sharp contrast to the set of configurations $\mathcal{C}$, the set of process-time

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

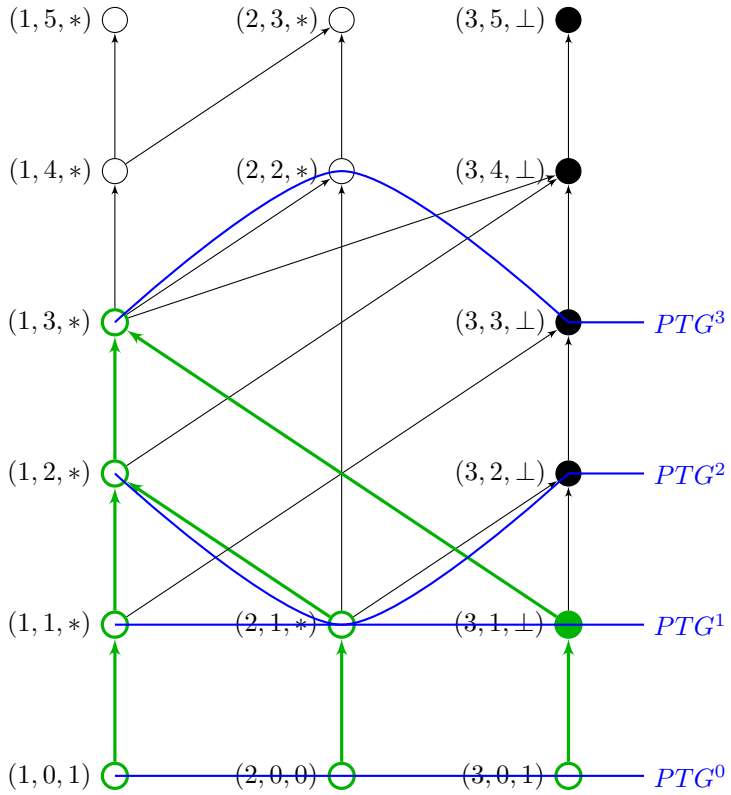

Fig. 6. Example of a process-time graph prefix in a non-lockstep execution of a system of $n = 3$ processes with initial values $x = (2, 0, 1)$, where process p_3 crashes in its step at time 1, in round 1. The vertical axis is the global time axis, and nodes at the same horizontal level occur at the same global time. The length of the edges represent end-to-end delay of a message resp. the access latency of a shared memory operation. Process 1's local view $V_1(PTG^3)$ in PTG^3 is highlighted in bold green.

graphs $\mathcal{PT}^t$ is finite for any time t and hence compact in the discrete topology, Tychonoff's theorem[8] implies compactness of the p-view topology on $\mathcal{PT}^\omega$.

Whereas this is not necessarily the case for $\mathcal{C}^\omega$, we can prove compactness of the image of $\mathcal{PT}^\omega$ under an appropriately defined operational transition function: Given the original transition function $\tau_\epsilon : \mathcal{G} \times \mathsf{ACT}_\epsilon \to \mathcal{L}_\epsilon$, it is possible to define a PTG transition function $\hat{\tau} : \mathcal{PT}^\omega \to \mathcal{C}^\omega$ that just provides the (unique) execution for a given process-time graph. The following Lemma A.2 shows that $\hat{\tau}$ is continuous in any of our topologies.

Lemma A.2 (Continuity of $\hat{\tau}$). *For every $p \in \Pi$, the PTG transition function $\hat{\tau} : \mathcal{PT}^\omega \to \mathcal{C}^\omega$ is continuous when both $\mathcal{PT}^\omega$ and $\mathcal{C}^\omega$ are endowed with any of d_p, $p \in \Pi$, d_{u}, d_{nu}.*

Proof. Let $U \subseteq \mathcal{C}^\omega$ be open with respect to d_p, and let $a \in \hat{\tau}^{-1}[U]$. Since U is open and $\hat{\tau}(a) \in U$, there exists some $\varepsilon > 0$ such that $B_\varepsilon\big(\hat{\tau}(a)\big) \subseteq U$. Let $t \in \mathbb{N}$ such that $2^{-t} \leq \varepsilon$. We

[8]Tychonoff's theorem states that any product of compact spaces is compact (with respect to the product topology).

will show that $B_{2^{-t}}(a) \subseteq \hat{\tau}^{-1}[U]$. For this, it suffices to show that $\hat{\tau}\big[B_{2^{-t}}(a)\big] \subseteq U$. By the equivalence of process-time graph prefixes and the corresponding consistent cuts, which is ensured by construction, it follows for the views of process p that $V_p(a^t) = V_p(b^t)$ implies $V_p(\hat{\tau}(a)^t) = V_p(\hat{\tau}(b)^t)$. Using this in Section 4.1 implies

$$\hat{\tau}\big[B_{2^{-t}}(a)\big] \subseteq B_{2^{-t}}(\hat{\tau}(a)) \subseteq B_{\varepsilon}(\hat{\tau}(a)) \subseteq U \ ,$$

which proves that $\hat{\tau}^{-1}[U]$ is open as needed.

The proof for d_{u} resp. d_{nu} is analogous, except that Section 4.1 must be replaced by Section 4.2 resp. Section 4.3. □

Since the image of a compact space under a continuous function is compact, it hence follows that the set $\hat{\tau}[\mathcal{PT}^{\omega}] \subseteq \mathcal{C}^{\omega}$ of admissible executions is a compact subspace of $\mathcal{C}^{\omega}$. The common structure of $\mathcal{PT}^{\omega}$ and its image under the PTG transition function $\hat{\tau}$, implied by the continuity of τ, hence allows us to reason in either of these spaces. In particular, with Definition A.3, the analog of Theorem 5.2 and Theorem 5.3 read as follows:

Definition A.3 (v-valent process-time graph). We call a process-time graph z_v, for $v \in \mathcal{V}$, v-valent, if it starts from an initial configuration where all processes $p \in \Pi$ have the same initial value $I_p = v$.

Theorem A.4 (Characterization of uniform consensus). *Uniform consensus is solvable if and only if there exists a partition of the set PS of admissible process-time graphs into sets PS_v, $v \in \mathcal{V}$, such that the following holds:*

(1) *Every PS_v is an open set in PS with respect to the uniform topology induced by d_{u}.*
(2) *If process-time graph $a \in PS$ is v-valent, then $a \in PS_v$.*

Theorem A.5 (Characterization of non-uniform consensus). *Non-uniform consensus is solvable if and only if there exists a partition of the set PS of admissible process-time graphs into sets PS_v, $v \in \mathcal{V}$, such that the following holds:*

(1) *Every PS_v is an open set in PS with respect to the non-uniform topology induced by d_{nu}.*
(2) *If process-time graph $a \in PS$ is v-valent, then $a \in PS_v$.*