A Topological View of Partitioning Arguments: Reducing *k*-Set Agreement to Consensus

Hugo Rincon Galeana[1], Kyrill Winkler[2], Ulrich Schmid[2](✉), and Sergio Rajsbaum[1]

[1] Instituto de Matemáticas, UNAM, CDMX, 04510 Mexico, D.F., Mexico
[2] TU Wien, ECS Group (E191-02), Treitlstrasse 1–3, 1040 Vienna, Austria
s@ecs.tuwien.ac.at

Abstract. The objective of this paper is to understand the effect of partitioning in distributed computing models. In spite of being quite similar agreement problems, (deterministic) consensus (1-set agreement) and k-set agreement (for $k > 1$) require surprisingly different techniques for proving impossibilities. There is a widely applicable generic theorem, however, which allows to reduce the impossibility of k-set agreement to consensus in message-passing models that allow some partitioning. In this paper, we provide the topological representation of this theorem, which reveals how partitioning is reflected in the protocol complex: It turns out that this leads to a "color splitting" of the algorithm's decision map, which separates the sub-complexes representing the partitioned processes. We also harvest a general advantage of topological results, which allowed us to carry over our findings to shared memory systems. We first demonstrate the utility of our reduction theorem by proving that d-set agreement cannot be solved in the d-solo asynchronous read-write model even when a single process may crash, not just in the wait-free case. Moreover, our new insights into the structure of protocol complexes gave us the idea for a simple proof of the fact that no partitioning argument can provide a valid impossibility proof for wait-free set agreement in the iterated immediate snapshot model: For any set of partition-compatible runs (which do not contain runs where all processes always have a complete view), we provide a way to construct a simple algorithm that solves set agreement.

Keywords: Algebraic topology · Consensus · Set agreement · Partitioning arguments · Shared memory

1 Introduction

Partitions, i.e., sets of processes that cannot always communicate with each other, are a fundamental combinatorial notion in distributed computability

This work has been supported by the PAPIIT-UNAM grant IN109917 and the Austrian Science Fund (FWF) projects RiSE/SHiNE (S11405) and ADynNet (P28182).

[13,16]. Studying the effect of partitioning on the (im)possibility of agreement problems has been a focal point at least since the realization that asynchronous consensus is solvable with a minority of initially crashed processes [15] and the observation that a shared register cannot be implemented on top of a message passing system with a majority of faulty processes [5]. Since then, partitioning arguments have been applied successfully to many distributed computing problems [14]. Essentially, these arguments exploit the fact that one cannot guarantee agreement among processes of a distributed system that never, neither directly nor indirectly, communicate with each other.

The objective of this paper is to understand the computational power of models where partitioning can occur. For this purpose, we focus on the k-set agreement problem and its special cases consensus and set agreement. In these problems, every process owns a local input value taken from a finite domain $\mathcal{V}$ (to rule out trivial solutions, we assume that $|\mathcal{V}| \geq k$), and must irrevocably assign a local output value (also called decision value) that must be the input value of some process and satisfy certain properties. For consensus, no two processes may decide on, i.e., assign, different values. For set agreement among n processes, the number of different decision values must be at most $n-1$ system-wide. The general case is k-set agreement, which requires that the number of different decision values is at most k. We will focus solely on deterministic algorithms.

Due to the landmark FLP impossibility result [15], which employs (now classic) bivalence proofs, it is well-known that consensus is impossible to solve in asynchronous systems if a single process may crash. The corresponding result for general k-set agreement is the impossibility of solving this problem in asynchronous systems where $f \geq k$ processes may crash. Surprisingly, establishing the latter result requires quite involved techniques based on algebraic topology resp. a variant of Sperner's lemma [11,20,23], which have not been matched by combinatorial proofs so far in their full generality (see the related work for some exceptions for special cases).

Despite this apparent "proof-incompatibility" of consensus and k-set agreement for $k > 1$, Biely, Robinson, and Schmid showed in [9] that, in message-passing models that allow partitioning, impossibility proofs for k-set agreement can be reduced to impossibility proofs for consensus: They provided a theorem (called BRS theorem in the sequel), which uses a partitioning argument as a means for reduction and is generic w.r.t. the underlying system model: Essentially, if failures and asynchrony allow for runs where the system partitions into k parts, the processes must decide on their own in every partition. By choosing distinct proposal values, solving k-set agreement in such runs requires solving consensus in every partition. Consequently, the impossibility of k-set agreement can be proved by showing that it is impossible to reach consensus in at least one of these partitions. Note that the BRS theorem actually works for fairly weak forms of partitioning, where some communication between partitions is still possible.

In [9], the authors applied the BRS theorem to various message-passing models, including purely asynchronous systems with crash failures, synchronous systems with omission failures, dynamic networks with omission failures, and even asynchronous systems with failure detectors.

Main Contributions: In order to understand the impact of partitioning in distributed computing models, we present a topological version/interpretation of the BRS theorem and, on the positive side, show that it can even be applied in the shared memory setting. On the negative side, we prove that one cannot show the impossibility of set agreement in the iterated immediate snapshot model based on a partitioning argument. In more detail:

(1) We provide a topological variant of a slightly generalized version of the BRS theorem and its proof for message-passing systems, which reveals that partitioning is reflected in the protocol complex by a "color splitting" of the algorithm's decision map that separates the sub-complexes representing the partitioned processes. This insight into the structure of the protocol complexes of partitionable models is of independent interest, as the second major contribution of our paper reveals.

(2) We exploit the natural genericity of topological results to translate the BRS theorem to the shared memory model. First, we apply it to d-set agreement in the d-solo asynchronous read-write model, where up to d processes may run solo: We provide a simple and illustrative proof that this problem is not just wait-free impossible, as has been shown in [18] already, but even impossible if just a single process may crash. Second, we used our new insights obtained in (1) to prove that one cannot hope to show the impossibility of wait-free set agreement in the iterated immediate snapshot model using any form of partitioning arguments: For any set of partition-compatible runs (which do not contain runs where all processes always have a complete view), we provide a way to construct a simple algorithm that solves set agreement.

In a nutshell, the results of our paper reveal how the partitioning argument implemented by means of the BRS theorem actually works: It is the "color-split" structure of the resulting protocol complex, which effectively allows to avoid a complex global topological analysis and to perform a simple reduction to the solvability of consensus in some partition instead. In the case of set agreement, the existence of any such splitting already guarantees a solution algorithm. We conjecture that we will be able to come up with similar statements also for other problems, in particular, for general k-set agreement. It is important to note, though, that our results apply only to systems where some partitioning can occur. They cannot hence replace results like [11,20,23] in general.

Related Work: We are not aware of much research that considers non-trivial reductions of k-set agreement to consensus. However, quite some papers, like [21], prove the impossibility of k-set agreement by partitioning the system into more than k sets of processes that decide independently.

For message-passing systems, besides [9], Biely et al. have employed reduction already in [8] to show that consensus is impossible in certain partially synchronous models, and to prove the tightness of the generalized loneliness failure detector $L(k)$ for k-set agreement. Similar reduction arguments were employed in [12] and, in particular, in [10], where certain k-set agreement runs with disjoint participants are pasted together in order to prove the necessity of the generalized quorum failure detector Σ_k for solving k-set agreement. In [4], a reduction to asynchronous set agreement is used to derive a lower bound on the minimum size of a "synchronous window" that is necessary for k-set agreement.

For shared memory systems, [1] shows the wait-free equivalence of k-set agreement and k-simultaneous consensus using read/write atomic registers. The latter problem allows processes to participate, with the same input, in k independent consensus instances that run simultaneously, but requires a decision only in one of those. Whereas this can be seen as an explicit form of partitioning, it is obviously much less general than the partitioning allowed by the BRS theorem.

Regarding different proof techniques for consensus and k-set agreement, the only alternatives to the celebrated impossibility proofs for k-set agreement [11,20,23] known to us, which are all based on algebraic topology resp. different proofs of Sperner's lemma, are the combinatorial impossibility proof for k-set agreement in wait-free environments provided in [6] and the counting-based impossibility of general wait-free colored and colorless tasks in [7]. The latter two results do not generalize to $k = f < n - 1$ crash failures, however.

In [2,3], Alistarh et al. described an approach for proving general impossibilities for wait-free tasks in the iterated immediate snapshot model by introducing *extension-based proofs*. The idea is to consider a game between a prover (constructing a schedule) and the protocol (specified by its decision map Δ, unknown to the prover). By constructing a protocol on the fly, via an adversarial strategy w.r.t. the prover, the authors could prove that the wait-free k-set agreement impossibility cannot be established by an extension-based proof. Part (2) of our work differs from [2] in that we restrict our attention to partitioning arguments, i.e., do not aim at general combinatorial proofs (not to speak of bivalence arguments) as does the latter. This restriction greatly reduces the effort needed to show that partitioning arguments are not sufficient for showing the wait-free set agreement impossibility: Rather than adversarially constructing a protocol following the prover's strategy, the construction of our set agreement protocol just instantiates a simple generic algorithm and is hence relatively easy.

Paper Organization: After a short introduction to topological modeling of distributed systems in Sect. 2, we translate the definitions and concepts underlying the original BRS theorem and prove[1] some basic lemmas in Sect. 3. In Sect. 4, we develop our topological version of the BRS theorem, in Sect. 5, we prove that set agreement can be solved for any set of partition-compatible runs. We conclude in Sect. 6 with some open questions.

2 Topological Modeling of Distributed Systems

We now briefly describe the basics of modeling distributed systems using algebraic topology, see e.g. [17] for a comprehensive introduction. Whereas this powerful approach became particularly popular for asynchronous shared memory systems [20], it is well-suited for message passing systems as well [19].

[1] Lacking space forced us to relegate all proofs into the full version [22] of our paper.

We consider a set $\Pi = \{p_1, \ldots, p_n\}$ of processes, each with its own unique identifier, which may suffer from crash or omission failures, i.e., may also lose messages. Except in Sect. 5, where we use the standard iterated snapshot asynchronous shared memory model, we will primarily deal with message passing protocols, where each process $p_i \in \Pi$ has an individual *message buffer* m_j for received messages from every potential sender process $p_j \in \Pi$ in its local state $(p_i, m_1, \ldots, m_n)$, where $(m_1, \ldots, m_n)$ is called p_i's *view*. Note that m_i is assumed to contain p_i's local variables. Valid messages are taken from a possibly infinite set M. Typically, we will consider deterministic full information protocols, where processes send messages that consist of the entire history of local states. We represent "p_2 receives message $m \in M$ from p_1" by appending m to the message buffer of p_2 that corresponds to p_1. We will assume that the initial input is given as a message from a process p_i to itself, and that the decision value is also appended to the message buffer reserved for itself. A global state of the system is a vector of local states, one for each $p_i \in \Pi$.

We define a run of a given protocol as a valid infinite sequence of global states, in which eventually each non-crashing process reaches a final decision state, i.e., a state where it has decided. Note carefully that, since we consider full-history protocols, the final decision state of a process contains its view of the entire run. Clearly, the decision of the process is determined by its local view at the first time it reached a decision state. We call such local views minimal final views. Note that we will assume that processes may still send messages after they have decided; messages that reach a process after a final decision state do not change the process' decision, however. Observe that if run α and run β have the same minimal final views for each process, then they have the same decisions. Therefore, we can restrict our attention to the equivalence classes of runs where two runs are equivalent if all processes have the same minimal final view.

We define a task $T_\Pi = \langle \mathcal{I}, \mathcal{O}, \Delta \rangle$ as a tuple, where $\mathcal{I}$ and $\mathcal{O}$ are chromatic simplicial complexes, which model the valid inputs and outputs for a set Π of processes, and $\Delta : \mathcal{I} \to 2^{\mathcal{O}}$ is a valid decision function that maps valid input configurations to sets of valid output configurations. Both complexes are chromatic, with coloring χ (formally, a simplicial map from the complex to a simplex of matching dimension, i.e., one that maps simplices to simplices) that attaches a unique label (in fact, a process id from Π) to every vertex such that no two neighbors in any 1-simplex have the same label. Note that we will usually write T instead of T_Π for brevity.

More formally, the input complex $\mathcal{I} = \langle V(\mathcal{I}), F(\mathcal{I}) \rangle$ is given by its set of vertices $V(\mathcal{I})$ and its set of faces $F(\mathcal{I})$: $V(\mathcal{I}) = \{(p_i, v_i) \mid p_i \in \Pi,\ v_i \in V_i\}$ where V_i is the set of valid inputs for p_i, and $F(\mathcal{I}) = \{\sigma \subseteq V(\mathcal{I}) \mid \sigma \text{ is part of a valid input cfg.}\}$.

The output complex $\mathcal{O} = \langle V(\mathcal{O}), F(\mathcal{O}) \rangle$ is given by its set of vertices $V(\mathcal{O})$ and its set of faces $F(\mathcal{O})$: $V(\mathcal{O}) = \{(p_i, v_i) \mid p_i \in \Pi,\ v_i \in \hat{V}_i\}$ where $\hat{V}_i$ is the set of valid outputs for p_i, $F(\mathcal{O}) = \{\sigma \subseteq V(\mathcal{O}) \mid \sigma \text{ is part of a valid output cfg.}\}$.

The decision function $\Delta : F(\mathcal{I}) \to 2^{F(\mathcal{O})}$ is a function with the property that, for every $\sigma \subseteq \rho$ and $\rho' \in \Delta(\rho)$, there is some $\sigma' \subseteq \rho'$ with $\sigma' \in \Delta(\sigma)$. Moreover,

Δ is a chromatic map, i.e., $\chi(\Delta(\sigma)) \subseteq \chi(\sigma)$, where $\chi(\sigma)$ gives the set of colors of the vertices in σ and $\chi(S) = \bigcup_{\sigma \in S} \chi(\sigma)$ for every set S of simplices.

We define the protocol complex $\mathcal{P}_{\mathcal{M}} = \langle V(\mathcal{P}_{\mathcal{M}}), F(\mathcal{P}_{\mathcal{M}}) \rangle$ for a given protocol P and model $\mathcal{M}$ as $V(\mathcal{P}_{\mathcal{M}}) = \{(p_i, v_i) \mid p_i \in \Pi,\ v_i \in \overline{V}_i\}$, where $\overline{V}_i$ is the set of valid minimal final views for p_i in protocol P under a given model $\mathcal{M}$. The set of faces is $F(\mathcal{P}_{\mathcal{M}}) = \{\sigma \subseteq V(\mathcal{P}_{\mathcal{M}})\}$, where σ corresponds to a valid configuration of minimal final views of a run. The chromatic function $\chi : V(\mathcal{P}_{\mathcal{M}}) \to \Pi$ is given by the id of each process, that is $\chi(p_i, v_i) = p_i$. The decision map for a protocol $\mu : V(\mathcal{P}_{\mathcal{M}}) \to 2^{V(\mathcal{O})}$ is a chromatic vertex map that maps final views of a process to valid outputs for a task.

Since the initial input values are self-messages in our model, each run is produced by a unique configuration of initial input values. Therefore, for each task T, there exists a chromatic simplicial map $i_T : F(\mathcal{P}_{\mathcal{M}}) \to F(\mathcal{I})$ that maps simplices to simplices of matching dimension, such that $i_T(\sigma)$ is the initial input configuration for each process in σ. A protocol P solves a task T in model $\mathcal{M}$ if and only if the decision map for the protocol is a simplicial map that carries Δ, i.e., ensures $\mu(\sigma) \subseteq \Delta(i_T(\sigma))$. Note that, since the decision map μ needs to be chromatic, it is determined by the mapping values at the facets, i.e., the maximal faces. Therefore, the facet decision map $\hat{\mu} : \hat{F}(\mathcal{P}_{\mathcal{M}}) \to 2^{V(\mathcal{O})}$, which is just the restriction of μ to the facets $\hat{F}(\mathcal{P}_{\mathcal{M}})$ in $F(\mathcal{P}_{\mathcal{M}})$, fully determines μ.

3 BRS Basic Definitions

In this section, we recast the foundations of the BRS theorem introduced in [9] in our topological framework. Most of the concepts introduced here allow to relate the runs of different algorithms in different models. After all, our purpose is to reduce a run of a k-set agreement algorithm A in some model $\mathcal{M}$ to a run of a consensus algorithm B in some model $\mathcal{M}'$, with a different set of processes and possibly different synchrony assumptions and failure models. The restricted model only requires that algorithm A is computationally compatible with $\mathcal{M}'$, i.e, that A can be executed in $\mathcal{M}'$. Since it is primarily the number of processes in $\mathcal{M}'$ and $\mathcal{M}'$ that matter here, we will follow [9]and sloppyly write $\mathcal{M} = \langle \Pi \rangle$ and $\mathcal{M}' = \langle D \rangle$ in the sequel. Note carefully, however, that the runs in $\mathcal{M}'$ do not necessarily correspond to runs in $\mathcal{M}$ and vice versa, due to $|\langle D \rangle| \neq |\langle \Pi \rangle|$ and the usually different synchrony assumptions and failure models.

A pivotal concept here is a restricted algorithm.

Definition 1 (Restricted algorithm). *Let A be an algorithm for a model $\mathcal{M} = \langle \Pi \rangle$ that consists of the set of processes Π, and $D \subseteq \Pi$ a nonempty set of processes. Consider a restricted model $\mathcal{M}' = \langle D \rangle$. To restrict algorithm A for model $\mathcal{M}$ to an algorithm for model $\mathcal{M}'$, we just drop all messages sent from D to the outside. We call the restricted algorithm $A_{|D} = B$.*

Restricted algorithms induce protocol complexes with specific properties. However, even if protocol B corresponds to a restriction of A to D, since message buffers for processes not in D are not present in $\mathcal{B}$ (the protocol complex of B),

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

Definition 2 (Protocol complexes of restricted algorithms). *Given an algorithm A for $\mathcal{M}$ and a restricted algorithm B for $\mathcal{M}'$, let $\mathcal{B}$ be the protocol complex for B executed in $\mathcal{M}'$. We define the extended complex of B with respect to Π, $\mathcal{A}_D$, as follows: $V(\mathcal{A}_D) = \{(p, w, \perp, \dots, \perp) \mid (p, w) \in V(\mathcal{B})\}$ and $\perp$ represents empty message buffers for processes in $\Pi \backslash D$, $F(\mathcal{A}_D) = \{\sigma \subseteq V(\mathcal{A}_D) \mid \exists \hat{\sigma} \in F(\mathcal{B}), (p, w, \perp, \dots, \perp) \in \sigma \Rightarrow (p, w) \in \hat{\sigma}\}$.*

The following Lemma 1 shows that $\mathcal{A}_D$ is isomorphic to $\mathcal{B}$, i.e., $\mathcal{A}_D$ is an "extended view copy" of $\mathcal{B}$. This isomorphic copy $\mathcal{A}_D$ of the protocol complex $\mathcal{B}$ will turn out to be essential, since it corresponds to a subcomplex of $\mathcal{A}$ under certain conditions (Definition 4). Note carefully that, per se, this is not necessarily the case as, e.g., the synchrony model for $\mathcal{M}$ may forbid empty message buffers.

Lemma 1 (Isomorphic complex). *Let $\mathcal{A}_D$ and $\mathcal{B}$ be as defined above. Then there exists a chromatic bijective simplicial map $\mu : \mathcal{A}_D \to \mathcal{B}$.*

The following definition captures the notion of indistinguishability of runs:

Definition 3 (Indistinguishability of runs). *Runs α and β are indistinguishable for a process p if p has the same sequence of states in α and β until p decides. For a non-empty set D of processes, we say that $\alpha \overset{D}{\sim} \beta$ if α is indistinguishable from β until decision for all $p \in D$.*

Note that since p has the same sequence of states until decision for runs α and β, then the minimal final view for p (that contains the full history) is the same for both runs α and β. This means that the simplices σ_α and σ_β that correspond to runs α and β share vertex $(p, s_w, m_1, \dots, m_k)$, where $(s_w, m_1, \dots, m_k)$ corresponds to the minimal final view of p at both runs α and β. This translates naturally to $D\text{-skel}(\sigma_\alpha) = D\text{-skel}(\sigma_\beta)$, where $D\text{-skel}(\sigma_\alpha) = \{(p, w) \in \sigma_\alpha \ : \ p \in D\}$.

Definition 4 (Compatibility of Runs). *Let $\mathcal{R}$ and $\mathcal{R}'$ be sets of runs, possibly from system models with different synchrony assumptions and failure models. Runs $\mathcal{R}'$ are compatible with runs $\mathcal{R}$ for processes in D, denoted by $\mathcal{R}' \preceq_D \mathcal{R}$, if $\forall \alpha \in \mathcal{R}' \exists \beta \in \mathcal{R} : a \overset{D}{\sim} \beta$.*

If $\mathcal{R}$ and $\mathcal{R}'$ are sets of runs for the same protocol A, and in the same model $\mathcal{M}$, then both induce subcomplexes of a common protocol complex $\mathcal{A}$. We will call those subcomplexes $\overline{\mathcal{R}}$ and $\overline{\mathcal{R}'}$ respectively. We define $D\text{-skel}(\overline{\mathcal{R}'})$ as the subcomplex of $\overline{\mathcal{R}'}$ where all vertices correspond to processes of D with views from $\overline{\mathcal{R}'}$. Given these definitions, it is clear that $\mathcal{R}' \preceq_D \mathcal{R}$ if and only if $D\text{-skel}(\overline{\mathcal{R}'}) \subseteq D\text{-skel}(\overline{\mathcal{R}})$. Note that the applicability of Definition 3 is limited, as it only works for sets of runs from the same protocol in the same model. However, we will give a more general definition below, which provides some correspondence between runs of *different* protocols in *different* models: Herein, $\mathcal{M}$ and $\mathcal{M}'$ only need to be computationally compatible, but may otherwise differ in the number of processes, synchrony assumptions, failure models, etc.

Definition 5 (D-View embedding). *Let A and B be protocols with a non-empty set of common processes S, and $D \subseteq S$ with $s = |S|$, and let $\mathcal{A}$ and $\mathcal{B}$ be the protocol complexes corresponding to A's runs in model $\mathcal{M}$ (with $s+r$ processes, $r \geq 0$) and B's runs in model $\mathcal{M}'$ (with $s+k$ processes, $k \geq 0$), respectively. Consider sets of runs $\mathcal{R}$ and $\mathcal{R}'$ from protocol A in model $\mathcal{M}$ and B in model $\mathcal{M}'$ respectively, and the corresponding subcomplexes $\overline{\mathcal{R}} \subseteq \mathcal{A}$ and $\overline{\mathcal{R}'} \subseteq \mathcal{B}$. We say that $\overline{\mathcal{R}'}$ is D-view embedded in $\overline{\mathcal{R}}$, if for every $(p, w_s, m_1, \ldots, m_k) \in D$-skel$(\overline{\mathcal{R}'})$ with w_s denoting the message buffers for the processes in S the following holds for every $1 \leq i \leq k$: (i) $p_i \notin S \Rightarrow m_i = \perp$; (ii) There exists $(p, w_s, m'_1, \ldots m'_r) \in V(\overline{\mathcal{R}})$ such that*

$$m'_j = \begin{cases} m_j & \text{if } p_j \in S, \\ \perp & \text{if } p_j \notin S; \end{cases}$$

(iii) $\mu : D$-skel$(\overline{\mathcal{R}'}) \to \overline{\mathcal{R}}$ defined by $(p, w_s, m_1, \ldots, m_k) \mapsto (p, w_s, m'_1, \ldots, m'_r)$ is a simplicial map. Note that μ is an embedding of D-skel$(\overline{\mathcal{R}'})$, i.e. an injective simplicial map.

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

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

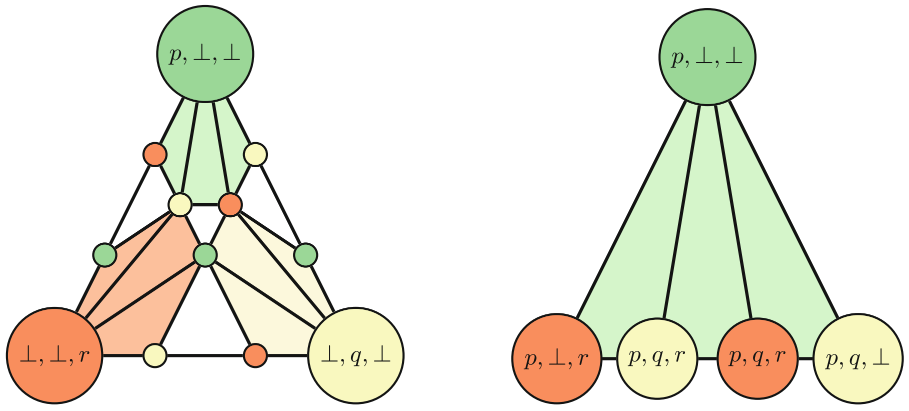
The following Definition 6 is crucial for expressing the consequences of partitioning in our topological setting. Essentially, it says that it is reflected by a splitting of the decision map.

Definition 6 (Decision map split). *Let $\mathcal{A}$ be the protocol complex for a given algorithm A on a model $\mathcal{M} = \langle \Pi \rangle$ and $\mathcal{A}'$ a non-empty subcomplex of $\mathcal{A}$. Let $D \subseteq \Pi$ be a set of processes in $\mathcal{M}$, $\overline{D} = \Pi \backslash D$ and $B = A_{|D}$ the restriction of algorithm A to D in a given model $\mathcal{M}' = \langle D \rangle$ with possibly different synchrony assumptions and failure models, resulting in protocol complex $\mathcal{B}$. We say that D splits the decision map of A at $\mathcal{A}'$ with respect to $\mathcal{M}'$ if $\mathcal{B}$ is D-view embedded in $\mathcal{A}'$ and*

$$\mu_{|\mathcal{A}'}(p, w) = [\mu_D * \mu_{|\overline{D}}](p, w) := \begin{cases} \mu_D(p, w) & \text{if } p \in D, \\ \mu_{|\overline{D}}(p, w) & \text{if } p \in \overline{D}. \end{cases}$$

Herein, $\mu_{|\mathcal{A}'}$ is the decision map μ of $\mathcal{A}$ restricted to $\mathcal{A}'$, μ_D is the decision map for restricted algorithm B at the extended complex $\mathcal{A}_D$, and $\mu_{|\overline{D}}$ is the decision map μ restricted to $\overline{D}$-skel$(\mathcal{A}')$.

An illustration of decision map splitting for a fixed $D = \{r, q\}, \bar{D} = \{p\}$ can be found in Fig. 1b. Note that the subcomplex for D (at the bottom) is the full 1-round subdivided complex for 2 processes, whereas the subcomplex for $\bar{D}$ represents a process that only hears from itself. Figure 1a shows the analogous splitting in the full one-round complex.



(a) Complex for d-solo runs $n = 3, d = 2$. (b) Subcomplex d-solo runs $D = \{r, q\}, \bar{D} = \{p\}$

Fig. 1. 1-round protocol complex of d-solo runs for $n = 3, d = 2$: (a) full complex, (b) zoom-in for $D = \{r, q\}, \bar{D} = \{p\}$. Nodes represent local process states (views), with the color encoding the respective process and with the 3-tuple encoding (m_p, m_q, m_r).

The following Lemma 3 shows that decision map splitting is equivalent to the extended complex of the restricted algorithm being equal to the corresponding D-skeleton:

Lemma 3 (Decision map splitting condition). *Let $\mathcal{A}$ be the protocol complex for a given algorithm A in a model $\mathcal{M} = \langle \Pi \rangle$ and $\mathcal{A}'$ a non-empty subcomplex of $\mathcal{A}$. Let $D \subseteq \Pi$ be a set of processes in $\mathcal{M}$, $B = A_{|D}$ a restriction of algorithm A to D in a model $\mathcal{M}'$ with possibly different synchrony assumptions and failure assumptions, $\mathcal{A}_D$ the extended complex of B with respect to Π and μ be the decision map for $\mathcal{A}$, then D splits μ at $\mathcal{A}'$ with respect to $\mathcal{M}' \Leftrightarrow \mathcal{A}_D = D$-skel$(\mathcal{A}')$.*

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

4 Topological BRS Theorem

Since we have already established an equivalence between topological conditions and run compatibility, we can proceed to state a slightly more general version of the BRS theorem from a topological perspective. The BRS theorem requires that conditions (A)–(D), as stated in Theorem 1 below, hold, in order to guarantee the k-set agreement impossibility. In the following lemmas, we will state topological properties that are slightly weaker than the original conditions (A)–(D).

Let $\mathcal{M} = \langle \Pi \rangle$ be a model and A an algorithm that runs in $\mathcal{M}$. Assume that each $p \in \Pi$ starts with an input value taken from a set of at least k different values.[2] Also assume that there is a distinguished set $D \subseteq \Pi$, and a partition of $\Pi \backslash D = \overline{D}$ given by $D_1, \ldots, D_{k-1}$. Let $\{v_1, \ldots, v_{k-1}\}$ be a fixed set of different values. Let $\mathcal{M}_A$ denote the runs of algorithm A in model $\mathcal{M}$.

Theorem 1 (Original BRS theorem [9]). *We consider the following runs of algorithm A in model $\mathcal{M}$ and a restricted model $\mathcal{M}' = \langle D \rangle$:*

dec-D *Any $p_j \in D$ receives no messages from any process in $\overline{D}$ until every process in D has decided.*
dec-$\overline{\mathbf{D}}$ *For every D_i, there is some $q \in D_i$ that decides v_i, which was proposed by some $p \in \overline{D}$.*

$\mathcal{R}_{(D)}$ denotes the set of runs from $\mathcal{M}$ where **dec-D** *holds, $\mathcal{R}_{(D,\overline{D})}$ denotes the set of runs from $\mathcal{M}$ where both* **dec-D** *and* **dec-$\overline{\mathbf{D}}$** *hold. If all of the following conditions (A) $\mathcal{R}_{(D)}$ is nonempty, (B) $\mathcal{R}_{(D)} \preceq_D \mathcal{R}_{(D,\overline{D})}$, (C) Consensus is not solvable in $\mathcal{M}'$, (D) $\mathcal{M}'_{A_{|D}} \preceq_D \mathcal{M}_A$, hold, then A does not solve k-set agreement.*

We generalize the statement of the BRS theorem by slightly relaxing conditions (A)–(D):

Lemma 4 (Condition equivalence). *Let $\mathcal{M}$, $\mathcal{M}'$, A, D, $\overline{D}$ and D_i be as defined for the BRS theorem. Then, (A)-(D) imply (A') $\mathcal{R}_{(D)}$ is nonempty, (B') Consensus is not solvable in $\mathcal{M}'$, (C') $\mathcal{M}'_{A_{|D}} \preceq_D \mathcal{R}_{(D,\overline{D})}$.*

The following Lemma 5 is the key technical lemma for the topological equivalence of a slightly stronger version of the BRS theorem established in Theorem 2 below.

Lemma 5 (BRS equivalence). *Let $\mathcal{M}$, $\mathcal{M}'$, A, D, $\overline{D}$ and D_i be as defined for the BRS theorem. Then (A')–(C') are equivalent to the following :*

(1) There exists a non empty subcomplex $\mathcal{A}'$ of $\mathcal{A}$ such that D-skel$(\mathcal{A}') = \mathcal{A}_D$ (the extended complex for $A_{|D}$ with respect to $\mathcal{M}$).
(2) For each D_i, the decision map $\mu_{|\mathcal{A}'}$ maps every view from D_i into a decision configuration that includes v_i as a decision value.

[2] Note that the original BRS theorem actually assumed that every process starts with a unique value.

(3) Each v_i is the input value for some process $p \in \overline{D}$ at subcomplex $\mathcal{A}'$.
(4) Consensus is not solvable in $\mathcal{M}' = \langle D \rangle$.

The following Theorem 2 provides the main result of this section, the topological version of a generalization of the BRS theorem.

Theorem 2 (Decision split theorem). *Let $\mathcal{M} = \langle \Pi \rangle$ be a model and A an algorithm that runs in $\mathcal{M}$. Let $\mathcal{A}$ be the protocol complex of A, $\mathcal{M}' = \langle D \rangle$ be a model with D a subset of Π, and μ the decision map for $\mathcal{A}$. Assume that the following conditions hold:*

(a) There exists a non-empty subcomplex $\mathcal{A}'$, such that D splits μ at $\mathcal{A}'$ with respect to $\mathcal{M}'$.
(b) Consensus is not solvable in $\mathcal{M}'$.
(c) Processes in $\overline{D} = \Pi \backslash D$ always decide at least $k-1$ input values from $\overline{D}$ for runs in $\mathcal{A}'$.

Then, A does not solve k-set agreement.

We conclude this section with some corollaries of Theorem 2.

Corollary 1. *Let $\mathcal{M}$, $\mathcal{M}'$, A, D, $\overline{D}$ and D_i be as defined for the original BRS theorem (Theorem 1). If conditions (1)–(4) given in Lemma 5 hold, then A does not solve k-set agreement.*

Corollary 2. *Let $\mathcal{M}$, $\mathcal{M}'$, A, D, $\overline{D}$ and D_i be as defined for the original BRS theorem (Theorem 1). If conditions (A')–(C') in Lemma 4 hold, then A does not solve k-set agreement.*

5 Partition Compatibility in Shared Memory and Set Agreement

In the previous sections, we developed a topological version of the BRS theorem [9], which allows to reduce the impossibility of k-set agreement to the impossibility of consensus in a wide variety of message-passing systems. In this section, we will transfer some of the resulting insights to the standard *asynchronous shared memory* (ASM) model. We thus consider a set of processes $\Pi = \{p_1, \ldots, p_n\}$ and a shared snapshot memory $M = (e, m_1, \ldots, m_n)$, where e is a buffer for global shared variables [not used in our protocols], and each m_i corresponds to the local memory portion of process p_i in the snapshot memory. To simplify our reasoning, we will assume that the protocols are full information immediate snapshot layered protocols, which does not change the computability power of our algorithms [17,20].

In this model, each process executes a predefined number r of asynchronous rounds, called layers. Each layer i consists of concurrent write-read snapshots. A write-read snapshot at layer i consists of writing the full view (the complete local state) of a process into its corresponding part of the memory, and immediately after writing, taking a snapshot of the views from processes at the same layer i. The initial view of a process consists only of its input value; therefore, during the first round, each process only writes its input value to the shared memory. Note that each process' current view contains the history of previous views, so we need not be concerned with overwriting previous views in the shared memory. The final view of a process in an r-round protocol is its view after round r, and the protocol's decision map μ maps the process's final view to some output value.

Definition 7 (Views). *The view for a given process p at round k is defined as follows: If $k = 0$, the view consists of a tuple (p, s, v), where p is the process id, s is the initial local state of p and v is the input value for p. If $k > 0$, the view consists of a tuple $(p, s, v_1, \ldots, v_n)$, where each v_j corresponds to either the view of process p_j at the end of round $k-1$ if the write-read execution for round k happened before or at the same time as the write-read execution for process p, or else $\perp$, which represents that p finished its write-read in round k before p_j.*

Note that this definition of the processes' views is extremely useful, since it has a nice combinatorial structure: Herlihy and Shavit showed in [20] that there is an isomorphism between the standard chromatic subdivision of the input complex and the protocol complex for a general 1-layer immediate snapshot protocol. Since each layer i is only determined by the previous layers and the scheduling of layer i, it follows by induction that a $k+1$ layered protocol complex corresponds to the $k+1$-th chromatic subdivision of the input complex.

We start our considerations by using the BRS theorem to show that d-set agreement cannot be implemented in the *d-solo model* introduced in [18] if at most d processes may crash. In the d-solo model for asynchronous shared memory, up to d processes may run solo, i.e., have no *other* process in their view in every round of a run. The wait-free 1-solo model is equivalent to asynchronous read-write shared memory, and any d-solo model can be simulated in the 1-solo model. Since d-set agreement for $d < n$ cannot be implemented in the wait-free read-write model [20], it cannot be implemented in the wait-free d-solo model either. The following Theorem 3 shows that this does not change if one strengthens the model by allowing only up to a single crash.

Theorem 3 (d-set agreement impossibility in d-solo model). *In the d-solo model, it is impossible to solve d-set agreement if just a single process may crash, not even with a colored, i.e., non-anonymous, algorithm.*

The remaining part of this section is devoted to the very different result of translating the insights gained from the topological version of the BRS theorem to the way of how partitioning is reflected in protocol complexes. More specifically, we show that set agreement can always be solved in ASM systems with *partition-compatible runs*, which are runs that allow some processes to hide their information from others. Our key insight is that partition-compatible runs "pierce" a hole at the center of the protocol complex, which allows the border to be retracted continuously to the center, thereby allowing to solve set agreement.

One implication of this result is that partitioning arguments cannot be used to prove the set-agreement impossibility in general: Assume that there is such a proof, which necessarily relies on some set of partition-compatible runs. After all, any partitioning argument rules out runs where every process has a complete view of all other processes in all iterations. Since we can construct a correct set agreement protocol for this set of runs, however, we have a contradiction.

The central idea of partitioning arguments, which exploit limited communication between sets of processes, stimulated the notion of partition-compatible views and (sets of) runs:

Definition 8 (Partition compatibility). *A view $(p_i, s, v_1, \ldots, v_n)$ of a process p_i at the end of some round $k \geq 1$ is called partition-compatible, if p_i did not get information from some p_j during round k, i.e., when $v_j = \perp$. A set of runs S is partition-compatible if, for every run $\alpha \in S$, there exists some participating process p_i and a round k at the end of which p_i's view is partition-compatible.*

Note that it is the presence of the complete-view run, i.e., the presence of the complete-view simplex in the corresponding protocol complex, that makes a set of runs *not* partition compatible. This, once made, obvious observation gave us the idea for the following simple set agreement protocol:

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

Definition 9 (1-layer protocol). *Let $p_i \in \Pi = \{p_0, \ldots, p_n\}$ be a process and $(m_0, \ldots, m_n)$ its view. Since we are considering the iterated immediate snapshot model, the protocol is determined just by the number of communication rounds (1 in this case) and the decision map. We define*

$$\mu(p_i, m_0, \ldots, m_n) = \begin{cases} m_i & \text{if } \perp = m_j \text{ for some } j \in \{1, \ldots, n\}, \\ m_{(i+1) mod\ n+1} & \text{otherwise.} \end{cases}$$

Obviously, since μ always chooses the input value for p_i unless all other input values have been observed, μ satisfies the validity condition. In fact, we can prove the following result:

Lemma 6 (Correctness of the 1-round protocol). *Let S be a set of partition compatible runs for a 1-round immediate snapshot protocol. Then, μ solves set agreement in S.*

An immediate consequence of Lemma 6 is that a partitioning argument cannot be used for showing n-set agreement impossibility for 1-round immediate snapshot protocols. However, in order to show that a partitioning argument cannot be used for a general shared memory protocol, we need to show this result for any number of layers. In order to do so, we define a k-round set agreement protocol for any value of k:

Definition 10 (k-layer protocol). *Consider a general k-layered immediate snapshot protocol. Let $p_i \in \Pi$ be a process, and $(m_0, \ldots, m_n)$ its final view. We* *denote $L^\ell(m_0,\dots,m_n)$ as the view for p_i at layer ℓ. Alternatively, if α is a run of an r-layered protocol and $s < r$ then L^s_α denotes the s-layered protocol run induced by α. We say that a view $v = (m_0,\dots,m_n)$ of a process at a layer s is incomplete if either $\perp \in v$ or if there exists $\ell < s$ and $0 \leq r \leq n$ such that $L^\ell(m_r)$ is an incomplete view; recall that m_r is p_r's view in round $s-1$ or $\perp$. Let*

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

We can prove the following result:

Lemma 7 (Correctness of the k-round protocol). *Let S be a set of partition compatible runs for a k-round immediate snapshot protocol. Then, the decision map μ^k solves n-set agreement in S.*

We can thus state the main result of this section:

Theorem 4. *Let S be any set of partition compatible runs in the iterated immediate snapshot model. Then, there exists a protocol P that solves set agreement for any run in S.*

6 Conclusions

We developed a topological version of a generalization of the BRS theorem. Our findings reveal that partitioning is reflected by a "color splitting" of the algorithm's decision map, which separates the sub-complexes representing the partitioned processes. We used these insights to show that the impossibility of wait-free set agreement in the layered immediate snapshot model cannot be proved using partitioning arguments: For any set of partition compatible runs it is possible to construct a simple protocol that solves set agreement.

Extending the latter to k-set agreement and investigating the applicability of the BRS theorem to alternative shared memory systems remain as open questions.

References

1. Afek, Y., Gafni, E., Rajsbaum, S., Raynal, M., Travers, C.: The k-simultaneous consensus problem. Distrib. Comput. **22**(3), 185–195 (2010). https://doi.org/10.1007/s00446-009-0090-8
2. Alistarh, D., Aspnes, J., Ellen, F., Gelashvili, R., Zhu, L.: Why extension-based proofs fail. CoRR abs/1811.01421 (2018)
3. Alistarh, D., Aspnes, J., Ellen, F., Gelashvili, R., Zhu, L.: Why extension-based proofs fail. In: Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing, STOC 2019, Phoenix, AZ, USA, 23–26 June 2019, pp. 986–996 (2019). https://doi.org/10.1145/3313276.3316407

4. Alistarh, D., Gilbert, S., Guerraoui, R., Travers, C.: Brief announcement: new bounds for partially synchronous set agreement. In: Lynch, N.A., Shvartsman, A.A. (eds.) DISC 2010. LNCS, vol. 6343, pp. 404–405. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-15763-9_40
5. Attiya, H., Bar-Noy, A., Dolev, D.: Sharing memory robustly in message-passing systems. J. ACM **42**(1), 124–142 (1995). https://doi.org/10.1145/200836.200869
6. Attiya, H., Castañeda, A.: A non-topological proof for the impossibility of k-set agreement. Theor. Comput. Sci. **512**, 41–48 (2013)
7. Attiya, H., Paz, A.: Counting-based impossibility proofs for renaming and set agreement. In: Aguilera, M.K. (ed.) DISC 2012. LNCS, vol. 7611, pp. 356–370. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-33651-5_25
8. Biely, M., Robinson, P., Schmid, U.: Weak synchrony models and failure detectors for message passing (k-)set agreement. In: Abdelzaher, T., Raynal, M., Santoro, N. (eds.) OPODIS 2009. LNCS, vol. 5923, pp. 285–299. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-10877-8_23
9. Biely, M., Robinson, P., Schmid, U.: Easy impossibility proofs for k-set agreement in message passing systems. In: Fernàndez Anta, A., Lipari, G., Roy, M. (eds.) OPODIS 2011. LNCS, vol. 7109, pp. 299–312. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-25873-2_21
10. Bonnet, F., Raynal, M.: On the road to the weakest failure detector for k-set agreement in message-passing systems. Theor. Comput. Sci. **412**(33), 4273–4284 (2011). https://doi.org/10.1016/j.tcs.2010.11.007
11. Borowsky, E., Gafni, E.: Generalized FLP impossibility result for t-resilient asynchronous computations. In: STOC 1993: Proceedings of the 25th Annual ACM Symposium on Theory of Computing, pp. 91–100. ACM, New York (1993). https://doi.org/10.1145/167088.167119
12. Bouzid, Z., Travers, C.: (anti–$\Omega^x \times \Sigma_z$)–based k-set agreement algorithms. In: Lu, C., Masuzawa, T., Mosbah, M. (eds.) OPODIS 2010. LNCS, vol. 6490, pp. 189–204. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-17653-1_16
13. Brewer, E.A.: Towards robust distributed systems (abstract). In: Proceedings of the Nineteenth Annual ACM Symposium on Principles of Distributed Computing, PODC 2000. ACM, New York (2000). https://doi.org/10.1145/343477.343502
14. Fich, F., Ruppert, E.: Hundreds of impossibility results for distributed computing. Distrib. Comput. **16**, 121–163 (2003). https://doi.org/10.1007/s00446-003-0091-y
15. Fischer, M.J., Lynch, N.A., Paterson, M.S.: Impossibility of distributed consensus with one faulty process. J. ACM **32**(2), 374–382 (1985)
16. Gilbert, S., Lynch, N.: Brewer's conjecture and the feasibility of consistent, available, partition-tolerant web services. SIGACT News **33**(2), 51–59 (2002). https://doi.org/10.1145/564585.564601
17. Herlihy, M., Kozlov, D.N., Rajsbaum, S.: Distributed Computing Through Combinatorial Topology. Morgan Kaufmann, Burlington (2013). https://store.elsevier.com/product.jsp?isbn=9780124045781
18. Herlihy, M., Rajsbaum, S., Raynal, M., Stainer, J.: From wait-free to arbitrary concurrent solo executions in colorless distributed computing. Theor. Comput. Sci. **683**, 1–21 (2017). https://doi.org/10.1016/j.tcs.2017.04.007
19. Herlihy, M., Rajsbaum, S., Tuttle, M.R.: An overview of synchronous message-passing and topology. Electron. Notes Theor. Comput. Sci. **39**(2), 1–17 (2000). https://doi.org/10.1016/S1571-0661(05)01148-5
20. Herlihy, M., Shavit, N.: The topological structure of asynchronous computability. J. ACM **46**(6), 858–923 (1999). https://doi.org/10.1145/331524.331529

21. de Prisco, R., Malkhi, D., Reiter, M.: On k-set consensus problems in asynchronous systems. IEEE Trans. Parallel Distrib. Syst. **12**(1), 7–21 (2001). https://doi.org/10.1109/71.899936
22. Rincon, H., Winkler, K., Schmid, U., Rajsbaum, S.: A topological view of partitioning arguments: reducing *k*-set agreement to consensus. Technical report TUW-281149, TU Wien (2019). https://publik.tuwien.ac.at/files/publik_281149.pdf
23. Saks, M., Zaharoglou, F.: Wait-free k-set agreement is impossible: the topology of public knowledge. SIAM J. Comput. **29**(5), 1449–1483 (2000). https://doi.org/10.1137/S0097539796307698