Why Extension-Based Proofs Fail

Dan Alistarh
IST Austria
Austria
dan.alistarh@ist.ac.at

James Aspnes
Yale University
USA
james.aspnes@gmail.com

Faith Ellen
University of Toronto
Canada
faith@cs.toronto.edu

Rati Gelashvili
University of Toronto
Canada
gelash@cs.toronto.edu

Leqi Zhu
University of Toronto
Canada
lezhu@cs.toronto.edu

ABSTRACT

It is impossible to deterministically solve wait-free consensus in an asynchronous system. The classic proof uses a valency argument, which constructs an infinite execution by repeatedly extending a finite execution. We introduce *extension-based proofs*, a class of impossibility proofs that are modelled as an interaction between a prover and a protocol and that include valency arguments.

Using proofs based on combinatorial topology, it has been shown that it is impossible to deterministically solve k-set agreement among $n > k \geq 2$ processes in a wait-free manner. However, it was unknown whether proofs based on simpler techniques were possible. We show that this impossibility result cannot be obtained by an extension-based proof and, hence, extension-based proofs are limited in power.

CCS CONCEPTS

• **Theory of computation** → **Interactive proof systems**; **Complexity theory and logic**; **Distributed algorithms**; *Distributed computing models*.

KEYWORDS

set agreement, impossibility proofs, valency arguments, combinatorial topology

ACM Reference Format:
Dan Alistarh, James Aspnes, Faith Ellen, Rati Gelashvili, and Leqi Zhu. 2019. Why Extension-Based Proofs Fail. In *Proceedings of the 51st Annual ACM SIGACT Symposium on the Theory of Computing (STOC '19), June 23–26, 2019, Phoenix, AZ, USA.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3313276.3316407

1 INTRODUCTION

One of the most well-known results in the theory of distributed computing, due to Fischer, Lynch, and Paterson [FLP85], is that there is no deterministic, wait-free protocol solving consensus among $n \geq 2$ processes in an asynchronous message passing system. In fact, they showed that, even if at most one process may crash, it is possible to construct an infinite execution in which no process terminates. Their result has been extended to asynchronous shared memory systems where processes communicate by reading from and writing to shared registers [Abr88, CIL87, Her91, LAA87].

Chaudhuri [Cha93] conjectured that these impossibility results could be generalized to the *k-set agreement problem.* In this problem, there are n processes, each starting with an input in $\{0, 1, \ldots, k\}$, where $1 \leq k < n$. Each process that does not crash must output a value that is the input of some process (*validity*) and, collectively, at most k different values may be output (*agreement*). In particular, *consensus* is 1-set agreement. Chaudhuri's conjecture was eventually proved in three concurrent papers by Borowsky and Gafni [BG93], Herlihy and Shavit [HS99], and Saks and Zaharoglou [SZ00]. These papers all relied on sophisticated machinery from combinatorial topology, which they used to model the space of all reachable configurations of the system. Later on, Attiya and Castañeda [AC11] and Attiya and Paz [AP12] showed how to obtain the same results using combinatorial techniques, without explicitly using topology. A common feature of these impossibility proofs is that they assume the existence of a deterministic, wait-free protocol, argue that it has only finitely many executions, and then show that at least $k+1$ different values are output in one of the executions. This implies that any deterministic protocol for k-set agreement among $n > k$ processes in an asynchronous system using only registers has an infinite execution. However, these proofs do not construct such an execution.

In contrast, impossibility proofs for deterministic, wait-free consensus in asynchronous systems explicitly construct an infinite execution by repeatedly extending a finite execution by the steps of some processes. Fischer, Lynch, and Paterson introduced *valency arguments* to show that such extensions are possible in the case of consensus. A natural question arises: is there a proof of the impossibility of k-set agreement that explicitly constructs an infinite execution by repeated extensions?

Our contributions. In this paper, we formally define the class of *extension-based proofs*, which model impossibility proofs that explicitly construct an infinite execution by repeated extensions. We also prove that that there is no extension-based proof of the impossibility of a deterministic, wait-free protocol solving k-set agreement among $n > k \geq 2$ processes in an asynchronous system using only registers.

We model a proof of the impossibility of solving a task as an interaction between a *prover* and a protocol that claims to solve the task. The prover has to refute this claim. To do so, it repeatedly queries the protocol about the states of processes in configurations that can be reached in a small number of steps from configurations it already knows about. The goal of the prover is to construct a *bad* execution, i.e. an execution in which some processes take infinitely many steps without terminating, or output values that do not satisfy the specifications of the task. For purposes of exposition, a restricted version of extension-based proofs is presented in Section 3, together with an example of such a proof (the impossibility of a deterministic, wait-free solution to consensus between 2 processes in an asynchronous shared memory system). The full version of extension-based proofs is presented in Section 5.

In Section 4, we prove that restricted extension-based proofs cannot show the impossibility of deterministic, wait-free set agreement. A key observation is that, from the results of its queries, many protocols are indistinguishable to the prover. It must construct a single execution that is bad for all these protocols. To prove that no prover can construct a bad execution for every k-set agreement protocol, we show how an adversary can adaptively define a protocol in response to any specific prover's queries. In this *adversarial protocol,* all processes eventually terminate and output correct values in executions consistent with the results of the prover's queries.

We use combinatorial topology to represent reachable configurations of a protocol. This is described in Section 2. More information can be found in the book by Herlihy, Kozlov, and Rajsbaum [HKR13]. In this view, when an extension-based prover makes queries, it is essentially performing local search on the configuration space of the protocol. Because the prover obtains incomplete information about the protocol, the adversary has some flexibility when specifying the protocol's behaviour in configurations not yet queried by the prover.

The class of extension-based proofs has an additional type of query, which makes the prover stronger and simplifies the modelling of valency arguments as extension-based proofs. In Section 5, we also extend the proof in Section 4 to handle this additional type of query.

Finally, possible extensions to extension-based proofs are discussed in Section 6.

2 PRELIMINARIES

2.1 NIIS Model

We consider the *non-uniform iterated immediate snapshot* (NIIS) model with $n \geq 2$ processes, introduced by Hoest and Shavit [HS06]. For deterministic, wait-free computation, it is known that the NIIS model is equivalent to the standard asynchronous shared memory model, in which processes communicate by reading from and writing to shared registers. Specifically, any task that has a deterministic, wait-free solution in one of these models has a deterministic, wait-free solution in the other [HS06].

In the NIIS model, n processes, $p_0, \ldots, p_{n-1}$, communicate using an infinite sequence, $S_1, S_2, \ldots$, of shared *single-writer atomic snapshot* objects. Each snapshot object has n components. The initial value of each component is $\bot$. A snapshot object supports two operations, update(v) and scan(). An update(v) operation by process p_i performed on a snapshot object updates component i of the object to have value v, where v is an element of an arbitrarily large set that does not contain $\bot$. A scan() operation returns a vector containing the current value of each component of the object.

Each process p_i accesses each snapshot object at most twice, starting with S_1. Initially, p_i's *state* consists of its identifier, i, and its input. The first time p_i accesses each snapshot object, it performs an update to set the i'th component of the object to its current state. At its next step, it performs a scan of the same object. Its new state, s_i, consists of i and the result of the scan. Note that process p_i remembers its entire history, because its previous state is the i'th component of the result of the scan. Next, p_i consults a map, Δ, to determine whether it should output a value. If $\Delta(s_i) \neq \bot$, then p_i outputs $\Delta(s_i)$ and terminates. If $\Delta(s_i) = \bot$, then, at its next step, p_i accesses the next snapshot object. A protocol in the NIIS model is completely specified by the map Δ.

A *configuration* consists of the contents of each shared object and the state of each process. A process is *active* in a configuration if it has not terminated. A configuration is *terminal* if it has no active processes. An *initial* configuration is specified by the input of each process.

From any configuration C, a *scheduler* decides the order in which the processes take steps. It repeatedly selects a set of processes that are all poised to perform updates on the same snapshot object. Each of the processes in the set, in order of their identifiers, performs its update. Then, each of these processes performs its next scan, in the same order. Note that the scheduler never selects processes that have terminated. The sequence of subsets of processes selected by the scheduler is called a *schedule from* C. Given a finite schedule α from C, we use $C\alpha$ to denote the resulting configuration. For example, if p_0 and p_1 are poised to access the same snapshot object in C, then $(\{p_0, p_1\})$ is a schedule from C. If p_1 is active in $C' = C(\{p_0, p_1\})$, then $(\{p_1\})$ is a schedule from C' and $(\{p_0, p_1\}, \{p_1\})$ is a schedule from C. After this schedule, p_1 has updated and scanned one more snapshot object than p_0. Each finite schedule from an initial configuration results in a *reachable* configuration. A protocol is *wait-free* if there is no infinite schedule from any initial configuration.

Since each process remembers its entire history and only process p_i can update the i'th component of each snapshot object, the contents of the snapshot objects in a configuration are fully determined by the states of the processes in that configuration.

Observation 1. *A reachable configuration is fully specified by the set of states of all processes in the configuration (including the processes that have terminated).*

Two configurations C and C' are *indistinguishable* to a set of processes P if every process in P has the same state in C and C'. Two finite schedules α and β from C are *indistinguishable* to a set of processes P if the resulting configurations $C\alpha$ and $C\beta$ are indistinguishable to P.

Observation 2. *Suppose C and C' are two reachable configurations that are indistinguishable to P, every active process in P is poised to* update *S_t in C, each snapshot object S_r has the same contents in C and C' for all $r \geq t$, and α is a finite schedule from C containing only processes in P. Then α is a schedule from C' and the configurations $C\alpha$ and $C'\alpha$ are indistinguishable to P.*

Suppose C is a reachable configuration in which all active processes are poised to update the same snapshot object. For any set of processes P, a *P-only 1-round schedule* from C is an ordered partition of the processes in P that are active in C. If none of the processes in P are active in C, then the empty schedule is the only 1-round schedule. A *full* 1-round schedule from C is a P-only 1-round schedule where P is the set of all processes. Observe that, in the NIIS model, if α is a P-only 1-round schedule from C and β is any full 1-round schedule from C such that $\beta = \alpha\alpha'$, then α and β are indistinguishable to the processes in P.

For each $t > 1$, a *P-only t-round schedule* from C is a schedule $\alpha_1 \cdots \alpha_t$ such that, for each $1 \leq i \leq t$, α_i is a P-only 1-round schedule from $C\alpha_1 \cdots \alpha_{i-1}$. Notice that some processes in P may have terminated during $\alpha_1 \cdots \alpha_{i-1}$. These processes are not included in α_i. A *full t-round schedule* from C is a P-only t-round schedule from C where P is the set of all processes. Observe that, if $\alpha_1 \cdots \alpha_t$ is a P-only t-round schedule from C, $\beta_1 \cdots \beta_t$ is a full t-round schedule from C, and α_i is a prefix of β_i for all $1 \leq i \leq t$, then the configurations $C\alpha_1 \cdots \alpha_t$ and $C\beta_1 \cdots \beta_t$ are indistinguishable to the processes in P.

2.2 Topological Representation of a Protocol

An *(abstract) simplex* is the set of all subsets of some finite set. An *(abstract) simplicial complex* is a finite collection of sets, $\mathbb{S}$, that is closed under subset: for any set $\sigma \in \mathbb{S}$, if $\tau \subseteq \sigma$, then $\tau \in \mathbb{S}$. In other words, $\mathbb{S}$ is the union of a finite number of simplices. Each set $\sigma \in \mathbb{S}$ is called a *face*. If $|\sigma| = 1$, then σ is called a *vertex*. If $|\sigma| = 2$, then σ is called an *edge*. A *subcomplex* of $\mathbb{S}$ is a subset of $\mathbb{S}$ that is also a simplicial complex.

In the *topological* view of a protocol in the NIIS model (specified by a map Δ), every reachable configuration, C, of the protocol is represented by a simplex, σ, which is the set of all subsets of states of processes in C. In particular, each vertex of σ is a set containing the state one process in C. Since the state of a process includes its identifier, no two processes have the same state in C and, hence, σ has n vertices. By Observation 1, the set of vertices of σ fully specifies C.

For each $t \geq 0$, let $\mathbb{S}^t$ denote the simplicial complex consisting of all simplices representing configurations reachable from initial configurations by full t-round schedules. In particular, the *input complex* of the protocol, $\mathbb{S}^0$, represents all possible initial configurations.

Hoest and Shavit [HS06] introduced the *non-uniform chromatic subdivision* operation, χ, and proved that $\mathbb{S}^{t+1} = \chi(\mathbb{S}^t, \Delta)$, i.e. $\mathbb{S}^{t+1}$ is the non-uniform chromatic subdivision of $\mathbb{S}^t$. In general, the *non-uniform chromatic subdivision operation* χ maps every subcomplex $\mathbb{A}$ of $\mathbb{S}^t$ to a subcomplex $\chi(\mathbb{A}, \Delta)$ of $\mathbb{S}^{t+1}$. It has the property that $\chi(\mathbb{A}, \Delta)$ is the union of $\chi(\sigma, \Delta)$ over all simplices $\sigma \subseteq \mathbb{A}$. The formal definition of this operation is fairly technical and appears in Section 2.3.

Suppose σ is an n-vertex simplex in $\mathbb{S}^t$, which represents a reachable configuration C. A special case of Hoest and Shavit's result is that every n-vertex simplex in $\chi(\sigma, \Delta) \subseteq \mathbb{S}^{t+1}$ represents a configuration reachable from C by a full 1-round schedule. More generally, if P is a subset of the processes and $\tau \subseteq \sigma$ is a simplex that consists of vertices representing the states of these processes in configuration C, then each simplex in $\chi(\tau, \Delta)$ consists of vertices representing the states of these processes in a configuration reachable from C by a P-only 1-round schedule. If τ is also a subset of an n-vertex simplex σ' that represents another configuration C', then, by Observation 2, for each P-only 1-round schedule α, the states of the processes in P are the same in $C\alpha$ and $C'\alpha$.

There is a natural geometric interpretation of an (abstract) simplicial complex and subdivision. A *geometric simplex* σ is the set of convex combinations of a finite number of affinely independent points (each of which is a *vertex* of σ) in some Euclidean space [HKR13]. A *face* of σ is the set of convex combinations of a subset of the affinely independent points. A *geometric simplicial complex* $\mathcal{K}$ is a finite collection of geometric simplices such that each face of $\sigma \in \mathcal{K}$ is a simplex in $\mathcal{K}$ and, for any two simplices $\sigma, \tau \in \mathcal{K}$, $\sigma \cap \tau \in \mathcal{K}$. The *geometric realization* of $\mathcal{K}$ is the union of the simplices in $\mathcal{K}$ (in Euclidean space). A geometric simplicial complex $\mathcal{B}$ is a *subdivision* of $\mathcal{A}$ if their geometric realizations are the same and each simplex in $\mathcal{A}$ is the union of finitely many simplices in $\mathcal{B}$. One of the main contributions of Hoest and Shavit [HS06] is showing that the non-uniform chromatic subdivision of a simplicial complex is a subdivision of the simplicial complex in the geometric setting.

Figure 1 contains an example of the non-uniform chromatic subdivision of a (geometric) simplicial complex $\mathbb{S}$ with 3 processes, p_0, p_1, and p_2, in Euclidean space. In the configuration represented by the left triangle of $\mathbb{S}$, p_0, p_1, and p_2 have states x, y, and z, respectively, none of which have terminated. In the configuration represented by the right triangle, p_0 and p_2 have the same state, but p_1 has state y', in which it terminates and outputs $\Delta(y')$. We also illustrate two subcomplexes $\mathbb{A}$ and $\mathbb{B}$ of $\mathbb{S}$ and their subdivisions. Note that all vertices in $\mathbb{S}$ and $\chi(\mathbb{S}, \Delta)$ representing states of the same process have the same colour. For readability, process identifiers are omitted from the states in $\chi(\mathbb{S}, \Delta)$.

2.3 Non-uniform Chromatic Subdivision

In this section, we define the non-uniform chromatic subdivision of a simplex and then extend the definition to a simplicial complex. Hoest and Shavit [HS06] discuss these definitions in more detail.

Let $t \geq 0$ and let $\sigma \subseteq \mathbb{S}^t$ be an n-vertex simplex that represents a reachable configuration C. For each vertex $v \in \sigma$, let *state*(v) be the state represented by v. For each simplex $\tau \subseteq \sigma$, let P_τ denote the set of processes whose states appear in τ and let *Id*(τ) be the set of identifiers of the processes in P_τ. Let $\vec{\tau}$ be the n-component vector such that $\vec{\tau}_i$ is *state*(v), if there is a vertex $v \in \tau$ that represents the state of process p_i, and $\perp$ otherwise. Then $\vec{\tau}$ is the result of each process in P_τ's last scan in the configuration CP_τ.

For any vertex v, we let $\Delta(v)$ denote $\Delta(\mathit{state}(v))$. We say that vertex v *is active* if $\Delta(v) = \perp$ and it *has terminated* if $\Delta(v) \neq \perp$. The non-uniform chromatic subdivision $\chi(\sigma, \Delta)$ of σ can only be defined when $\Delta(v)$ is defined for every vertex $v \in \sigma$. The non-uniform chromatic subdivision $\chi(\mathbb{A}, \Delta) \subseteq \mathbb{S}^{t+1}$ of $\mathbb{A}$ is the union of $\chi(\sigma, \Delta)$ for all simplices $\sigma \subseteq \mathbb{A}$. To define $\chi(\sigma, \Delta)$, we consider two cases.

Case 1: *Every vertex $v \in \sigma$ is active.* Then $\chi(\sigma, \Delta)$ is called the *standard chromatic subdivision* of σ. The vertices of $\chi(\sigma, \Delta)$ are of

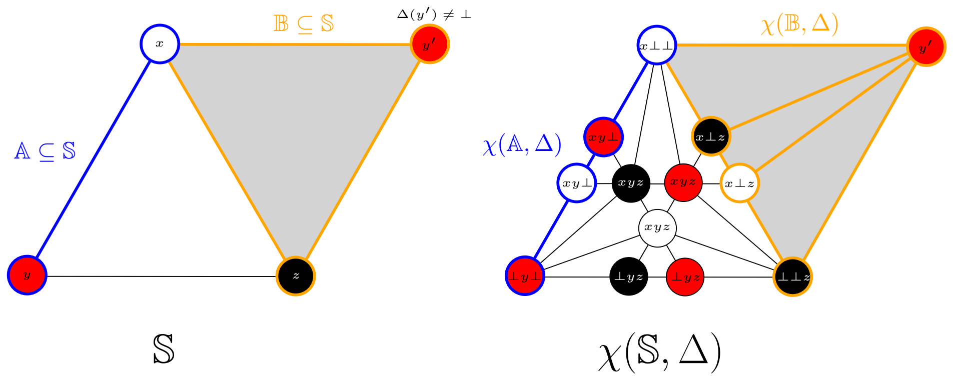
$\Delta(y') \neq \perp$
$\mathbb{B} \subseteq \mathbb{S}$
$\mathbb{A} \subseteq \mathbb{S}$
x
y'
y
z
$\mathbb{S}$
$\chi(\mathbb{B}, \Delta)$
$\chi(\mathbb{A}, \Delta)$
$x\perp\perp$
$xy\perp$
$x\perp z$
xyz
$\perp y \perp$
$\perp yz$
$\perp\perp z$
$\chi(\mathbb{S}, \Delta)$

Figure 1: A non-uniform chromatic subdivision.

the form $\{(i, \vec{\tau}_i)\}$, where $i \in Id(\sigma)$ is an identifier and $\tau_i \subseteq \sigma$ is a simplex such that $i \in Id(\tau_i)$. Suppose that $I \subseteq Id(\sigma)$ is a set of identifiers and, for each $i \in I$, $\tau_i \subseteq \sigma$ is a simplex such that $i \in Id(\tau_i)$. Then $\{(i, \vec{\tau}_i) : i \in I\} \in \chi(\sigma, \Delta)$ if and only if there is an ordering $\preceq$ on I such that $i \preceq j$ implies that $\tau_i \subseteq \tau_j$ and, for each $i, j \in I$, if $i \in Id(\tau_j)$, then $\tau_i \subseteq \tau_j$.

Case 2: *Some vertex $v \in \sigma$ has terminated.* Let T be the set of terminated vertices in σ and let $\tau \subseteq \sigma$ be the simplex consisting of all subsets of σ that only contain active vertices. The *non-uniform chromatic subdivision* of σ is the abstract simplicial complex $\chi(\sigma, \Delta)$ whose vertices are the vertices in T and the vertices in the standard chromatic subdivision $\chi(\tau, \Delta)$ of τ. Each set in $\chi(\tau, \Delta)$ is a set in $\chi(\sigma, \Delta)$. In addition, if $T' \subseteq T$ and $\tau' \in \chi(\tau, \Delta)$, then $T' \cup \tau' \in \chi(\sigma, \Delta)$.

We note the following property for terminated vertices:

Proposition 2.1. *Suppose vertex v has terminated in $\mathbb{S}^t$. Let $\mathbb{A}$ be the subcomplex of $\mathbb{S}^t$ consisting of all subsets in $\mathbb{S}^t$ that only contain vertices adjacent to v in $\mathbb{S}^t$. Then v is adjacent to each vertex in $\chi(\mathbb{A}, \Delta)$ in $\mathbb{S}^{t+1}$.*

2.4 Distances between Subcomplexes

Let $\mathbb{S}$ be a simplicial complex and let $\mathbb{A}$ and $\mathbb{B}$ be non-empty subcomplexes of $\mathbb{S}$. A *path* between $\mathbb{A}$ and $\mathbb{B}$ in $\mathbb{S}$ of *length* ℓ is a sequence of vertices $v_0, v_1, \ldots, v_\ell$ such that $v_0 \in \mathbb{A}$, $v_\ell \in \mathbb{B}$, and, for $0 \leq j < \ell$, $\{v_j, v_{j+1}\}$ is an edge in $\mathbb{S}$. Notice that a vertex may appear more than once in a path and, if $\mathbb{A}$ and $\mathbb{B}$ both consist of a single vertex, then we have the standard definition of a (non-simple) path in an undirected graph. $\mathbb{S}$ is *connected* if for any two vertices $u, v \in \mathbb{S}$, there is path between u and v in $\mathbb{S}$. If $\mathbb{S}$ is connected, then the *distance* between $\mathbb{A}$ and $\mathbb{B}$ in $\mathbb{S}$, denoted $\text{dist}_{\mathbb{S}}(\mathbb{A}, \mathbb{B})$, is the minimum $\ell \geq 0$ such that there is a path between $\mathbb{A}$ and $\mathbb{B}$ in $\mathbb{S}$ of length ℓ.

The following proposition describes some basic properties of the non-uniform chromatic subdivision operation, which all follow from the fact that the non-uniform chromatic subdivision of a simplicial complex is a subdivision of the simplicial complex in the geometric setting. Since the input complex, $\mathbb{S}^0$, is connected, the proposition implies $\mathbb{S}^t$ is connected for all $t \geq 1$. Hence, the distance between subcomplexes in $\mathbb{S}^t$ is well-defined.

Proposition 2.2. *The* non-uniform chromatic subdivision operation *has the following properties:*

1. *If $\mathbb{S}^0$ is connected, then, for all $t \geq 1$, $\mathbb{S}^t$ is connected.*
2. *$\mathbb{A}$ and $\mathbb{B}$ are disjoint subcomplexes of $\mathbb{S}^t$ if and only if $\chi(\mathbb{A}, \Delta)$ and $\chi(\mathbb{B}, \Delta)$ are disjoint subcomplexes of $\mathbb{S}^{t+1}$.*
3. *If every path between subcomplexes $\mathbb{A}$ and $\mathbb{B}$ in $\mathbb{S}^t$ passes through a subcomplex $\mathbb{C}$, then every path between $\chi(\mathbb{A}, \Delta)$ and $\chi(\mathbb{B}, \Delta)$ in $\mathbb{S}^{t+1}$ passes through $\chi(\mathbb{C}, \Delta)$.*
4. *If $\mathbb{C} \subseteq \mathbb{S}^t$ is a subcomplex containing only active vertices and $\mathbb{C}_1$ and $\mathbb{C}_2$ are disjoint nonempty subcomplexes of $\mathbb{C}$, then $\text{dist}_{\mathbb{S}^{t+1}}(\chi(\mathbb{C}_1, \Delta), \chi(\mathbb{C}_2, \Delta)) \geq 2$.*

We now prove one of the main technical tools used in this paper. It allows us to relate the distance between two subcomplexes in $\mathbb{S}^t$ to the distance between the nonuniform chromatic subdivisions of these subcomplexes in $\mathbb{S}^{t+1}$.

Lemma 2.3. *Let $\mathbb{A}$ and $\mathbb{B}$ be nonempty subcomplexes of $\mathbb{S}^t$, let $\mathbb{A}' = \chi(\mathbb{A}, \Delta)$, and let $\mathbb{B}' = \chi(\mathbb{B}, \Delta)$. Then $\text{dist}_{\mathbb{S}^{t+1}}(\mathbb{A}', \mathbb{B}') \geq \text{dist}_{\mathbb{S}^t}(\mathbb{A}, \mathbb{B})$. If every path between $\mathbb{A}$ and $\mathbb{B}$ in $\mathbb{S}^t$ contains at least one edge between active vertices, then $\text{dist}_{\mathbb{S}^{t+1}}(\mathbb{A}', \mathbb{B}') \geq \text{dist}_{\mathbb{S}^t}(\mathbb{A}, \mathbb{B}) + 1$.*

Proof. We will prove the first claim by induction on $\text{dist}_{\mathbb{S}^t}(\mathbb{A}, \mathbb{B})$. The base case is when $\text{dist}_{\mathbb{S}^t}(\mathbb{A}, \mathbb{B}) = 0$. Since distances are always non-negative, $\text{dist}_{\mathbb{S}^{t+1}}(\mathbb{A}', \mathbb{B}') \geq 0 = \text{dist}_{\mathbb{S}^t}(\mathbb{A}, \mathbb{B})$.

So, suppose $d = \text{dist}_{\mathbb{S}^t}(\mathbb{A}, \mathbb{B}) > 0$ and the first claim holds for all non-empty subcomplexes $\hat{\mathbb{A}}$ and $\hat{\mathbb{B}}$ of $\mathbb{S}^t$ where $\text{dist}_{\mathbb{S}^t}(\hat{\mathbb{A}}, \hat{\mathbb{B}}) < d$. Let $\mathbb{C}$ be the subcomplex of $\mathbb{S}^t$ consisting of all subsets in $\mathbb{S}^t$ that only contain vertices v for which $\text{dist}_{\mathbb{S}^t}(v, \mathbb{A}) = 1$ and let $\mathbb{C}' = \chi(\mathbb{C}, \Delta)$. Since $\text{dist}_{\mathbb{S}^t}(\mathbb{A}, \mathbb{B}) > 0$, $\mathbb{C}$ is non-empty and $\text{dist}_{\mathbb{S}^t}(\mathbb{C}, \mathbb{B}) = d - 1$. Moreover, since $\mathbb{A}$ and $\mathbb{C}$ are disjoint, by Proposition 2.2(2), $\mathbb{A}'$ and $\mathbb{C}'$ are disjoint, i.e. $\text{dist}_{\mathbb{S}^{t+1}}(\mathbb{A}', \mathbb{C}') \geq 1$. By the induction hypothesis applied to $\mathbb{C}$ and $\mathbb{B}$, $\text{dist}_{\mathbb{S}^{t+1}}(\mathbb{C}', \mathbb{B}') \geq \text{dist}_{\mathbb{S}^t}(\mathbb{C}, \mathbb{B}) = d - 1$. Since every path between $\mathbb{A}$ and $\mathbb{B}$ in $\mathbb{S}^t$ passes through $\mathbb{C}$, by Proposition 2.2(3), every path between $\mathbb{A}'$ and $\mathbb{B}'$ in $\mathbb{S}^{t+1}$ passes through $\mathbb{C}'$. Therefore, $\text{dist}_{\mathbb{S}^{t+1}}(\mathbb{A}', \mathbb{B}') \geq \text{dist}_{\mathbb{S}^{t+1}}(\mathbb{A}', \mathbb{C}') + \text{dist}_{\mathbb{S}^{t+1}}(\mathbb{C}', \mathbb{B}') \geq d$ and the first claim holds for $\mathbb{A}$ and $\mathbb{B}$.

Now suppose that every path between $\mathbb{A}$ and $\mathbb{B}$ in $\mathbb{S}^t$ contains at least one edge between active vertices. Let $E \neq \emptyset$ be a smallest set of edges between active vertices in $\mathbb{S}^t$ such that every path between $\mathbb{A}$ and $\mathbb{B}$ contains at least one edge in E. Viewing $\mathbb{S}^t$ as a graph, the removal of E from $\mathbb{S}^t$ results in some number of connected components. Let $\hat{A}$ be the set of vertices in the connected components that contain at least one vertex in $\mathbb{A}$ and let $\hat{B}$ be the set of remaining vertices in $\mathbb{S}^t$. Let $\hat{\mathbb{A}}$ and $\hat{\mathbb{B}}$ be the subcomplexes of $\mathbb{S}^t$ consisting of all subsets in $\mathbb{S}^t$ that only contain vertices in $\hat{A}$ and $\hat{B}$, respectively. Observe that $\mathbb{A}$ is a subcomplex of $\hat{\mathbb{A}}$ and $\mathbb{B}$ is a subcomplex of $\hat{\mathbb{B}}$. Let $\hat{E} \subseteq E$ be the set of edges between $\hat{\mathbb{A}}$ and $\hat{\mathbb{B}}$. Note that $\hat{A}$ and $\hat{B}$ partition the vertices in $\mathbb{S}^t$. Hence, every path between $\hat{\mathbb{A}}$ and $\hat{\mathbb{B}}$ contains an edge in $\hat{E}$. In particular, every path between $\mathbb{A} \subseteq \hat{\mathbb{A}}$ and $\mathbb{B} \subseteq \hat{\mathbb{B}}$ contains an edge in $\hat{E}$. By the minimality of E, $E = \hat{E}$.

Let $\mathbb{C}$ be the subcomplex of $\mathbb{S}^t$ consisting of all subsets in $\mathbb{S}^t$ that only contain vertices contained in some edge in E. By definition of E, every vertex in $\mathbb{C}$ is active. Moreover, every path between $\hat{\mathbb{A}}$ and $\hat{\mathbb{B}} \cap \mathbb{C}$ passes through $\hat{\mathbb{A}} \cap \mathbb{C}$ and every path between $\hat{\mathbb{A}} \cap \mathbb{C}$ and $\hat{\mathbb{B}}$ passes through $\hat{\mathbb{B}} \cap \mathbb{C}$.

Let $\hat{\mathbb{A}}' = \chi(\hat{\mathbb{A}}, \Delta)$ and let $\hat{\mathbb{B}}' = \chi(\hat{\mathbb{B}}, \Delta)$. Then, by Proposition 2.2(3), every path between $\hat{\mathbb{A}}'$ and $\chi(\hat{\mathbb{B}} \cap \mathbb{C}, \Delta)$ passes through $\chi(\hat{\mathbb{A}} \cap \mathbb{C}, \Delta)$ and every path between $\chi(\hat{\mathbb{A}} \cap \mathbb{C}, \Delta)$ and $\hat{\mathbb{B}}'$ passes through $\chi(\hat{\mathbb{B}} \cap \mathbb{C}, \Delta)$. It follows that every path between $\mathbb{A}' \subseteq \hat{\mathbb{A}}'$ and $\mathbb{B}' \subseteq \hat{\mathbb{B}}'$ consists of a path between $\mathbb{A}'$ and $\chi(\hat{\mathbb{A}} \cap \mathbb{C}, \Delta)$, a path between $\chi(\hat{\mathbb{A}} \cap \mathbb{C}, \Delta)$ and $\chi(\hat{\mathbb{B}} \cap \mathbb{C}, \Delta)$, and a path between $\chi(\hat{\mathbb{B}} \cap \mathbb{C}, \Delta)$ and $\mathbb{B}'$. Thus

$$\begin{aligned} \mathrm{dist}_{\mathbb{S}^{t+1}}(\mathbb{A}', \mathbb{B}') \geq\ & \mathrm{dist}_{\mathbb{S}^{t+1}}(\mathbb{A}', \chi(\hat{\mathbb{A}} \cap \mathbb{C}, \Delta)) \\ &+ \mathrm{dist}_{\mathbb{S}^{t+1}}(\chi(\hat{\mathbb{A}} \cap \mathbb{C}, \Delta), \chi(\hat{\mathbb{B}} \cap \mathbb{C}, \Delta)) \\ &+ \mathrm{dist}_{\mathbb{S}^{t+1}}(\chi(\hat{\mathbb{B}} \cap \mathbb{C}, \Delta), \mathbb{B}')\,. \end{aligned}$$

By the first claim,

$$\begin{aligned} &\mathrm{dist}_{\mathbb{S}^{t+1}}(\mathbb{A}', \chi(\hat{\mathbb{A}} \cap \mathbb{C}, \Delta)) \geq \mathrm{dist}_{\mathbb{S}^{t}}(\mathbb{A}, \hat{\mathbb{A}} \cap \mathbb{C}) \text{ and} \\ &\mathrm{dist}_{\mathbb{S}^{t+1}}(\mathbb{B}', \chi(\hat{\mathbb{B}} \cap \mathbb{C}, \Delta)) \geq \mathrm{dist}_{\mathbb{S}^{t}}(\mathbb{B}, \hat{\mathbb{B}} \cap \mathbb{C})\,. \end{aligned}$$

Now consider $\mathrm{dist}_{\mathbb{S}^{t+1}}(\chi(\hat{\mathbb{A}} \cap \mathbb{C}, \Delta), \chi(\hat{\mathbb{B}} \cap \mathbb{C}, \Delta))$. Since every vertex in $\mathbb{C}$ is active and $\hat{\mathbb{A}} \cap \mathbb{C}$ and $\hat{\mathbb{B}} \cap \mathbb{C}$ are disjoint non-empty subcomplexes of $\mathbb{C}$, Proposition 2.2(4) implies that

$$\mathrm{dist}_{\mathbb{S}^{t+1}}(\chi(\hat{\mathbb{A}} \cap \mathbb{C}, \Delta), \chi(\hat{\mathbb{B}} \cap \mathbb{C}, \Delta)) \geq 2\,.$$

By definition of $\mathbb{C}$, $\mathrm{dist}_{\mathbb{S}^t}(\hat{\mathbb{A}} \cap \mathbb{C}, \hat{\mathbb{B}} \cap \mathbb{C}) = 1$. Hence,

$$\mathrm{dist}_{\mathbb{S}^t}(\mathbb{A}, \hat{\mathbb{A}} \cap \mathbb{C}) + \mathrm{dist}_{\mathbb{S}^t}(\mathbb{B}, \hat{\mathbb{B}} \cap \mathbb{C}) = \mathrm{dist}_{\mathbb{S}^t}(\mathbb{A}, \mathbb{B}) - 1\,.$$

Finally, combining these equations, it follows that $\mathrm{dist}_{\mathbb{S}^{t+1}}(\mathbb{A}', \mathbb{B}') \geq \mathrm{dist}_{\mathbb{S}^t}(\mathbb{A}, \mathbb{B}) + 1$. □

3 RESTRICTED EXTENSION-BASED PROOFS

In this section, we introduce and formally define the class of *restricted extension-based proofs*, which are used to prove impossibility results. To prove that a task has no wait-free solution, given any protocol that supposedly solves the task, such a proof constructs a schedule from an initial configuration, which witnesses the fact that the protocol is incorrect. Specifically, either the schedule is infinite or one of the specifications of the task is violated in the configuration resulting from the schedule. To learn information about the protocol, the proof queries the protocol about the states of processes in various reachable configurations. In the NIIS model, the only information a proof learns about the state of a process in a reachable configuration is whether that process has output a value and, if so, the value that it output. The rest of the information about its state is the same for all NIIS protocols. In other words, if the protocol is specified by a map Δ from process states to outputs or $\perp$, then the proof is querying the map Δ at various process states.

More formally, a *restricted extension-based* proof in the NIIS model is an interaction between a prover and any protocol defined by a map Δ from process states to outputs or $\perp$. The prover starts with no knowledge about the protocol (except its initial configurations) and makes the protocol reveal information about the states of processes in various configurations by asking *queries*. Each query allows the prover to *reach* some configuration of the protocol. The interaction proceeds in phases, beginning with phase 1.

In each phase $\varphi \geq 1$, the prover starts with a finite schedule, $\alpha(\varphi)$, and a set, $\mathscr{A}(\varphi)$, of configurations that are reached by performing $\alpha(\varphi)$ from initial configurations. These initial configurations only differ from one another in the input values of processes that do not appear in the schedule $\alpha(\varphi)$. The prover also maintains a set, $\mathscr{A}'(\varphi)$, containing the configurations it reaches by non-empty schedules from configurations in $\mathscr{A}(\varphi)$ during phase φ. This set is empty at the start of phase φ. At the start of phase 1, $\alpha(1)$ is the empty schedule and $\mathscr{A}(1)$ is the set of all initial configurations of the protocol.

The prover *queries* the protocol by specifying a configuration $C \in \mathscr{A}(\varphi) \cup \mathscr{A}'(\varphi)$ and a set of processes P that are poised to update the same snapshot object in C. For each process $p_i \in P$, let s_i denote the state of p_i in the configuration C' resulting from scheduling P from C. The protocol replies to this query with $\Delta(i, s_i)$, for each $p_i \in P$. Notice that, by the definition of the NIIS model, this is enough for the prover to know the state of every process and the contents of every component of every snapshot object in configuration C'. Then, the prover adds C' to $\mathscr{A}'(\varphi)$, and we say that the prover has *reached* C'.

If the prover reaches a configuration in which the outputs of the processes do not satisfy the specifications of the task, it has demonstrated that the protocol is incorrect. In this case, the prover *wins* (and the interaction ends).

A *chain of queries* is a (finite or infinite) sequence of queries such that, if (C_i, P_i) and (C_{i+1}, P_{i+1}) are consecutive queries in the chain, then C_{i+1} is the configuration resulting from scheduling P_i from C_i. If the prover constructs an infinite chain of queries, it has demonstrated that the protocol is not wait-free. In this case, the prover also *wins* (and the interaction ends). In particular, the prover wins against the trivial protocol in which no process ever outputs a value, by constructing any infinite chain of queries. After constructing finitely many chains of queries in phase φ without winning, the prover must end the phase by committing to an extension of the schedule $\alpha(\varphi)$.

It is important that the prover is allowed to construct chains of queries instead of just single queries. Whenever a chain of queries results in a terminal configuration, rather than going on forever, the prover learns useful information about the protocol, i.e. the outputs of all processes in that configuration. If the prover cannot construct a chain of queries, then it might be forced to commit to an extension without learning any useful information about the protocol. This is because it is possible to add any finite number of rounds at the beginning of a protocol that are ignored by the processes.

Suppose the prover chooses configuration $C' \in \mathscr{A}'(\varphi)$ at the end of phase φ. Let α' be a (nonempty) schedule such that C' is reached by performing α' starting from some configuration $C \in \mathscr{A}(\varphi)$. Let $\alpha(\varphi + 1)$ denote the schedule $\alpha(\varphi)\alpha'$. Since $C \in \mathscr{A}(\varphi)$, there is an initial configuration I such that C is reached by performing $\alpha(\varphi)$

starting from I. Thus C' is reached by performing $\alpha(\varphi + 1)$ starting from I. Finally, let $\mathscr{A}(\varphi + 1)$ be the set of all configurations that are reached by performing $\alpha(\varphi + 1)$ from the initial configurations that only differ from I by the states of processes that do not appear in this schedule. Then the prover begins phase $\varphi + 1$.

If, in every configuration in $\mathscr{A}(\varphi)$, every process has terminated, then $\mathscr{A}'(\varphi) = \emptyset$, the prover *loses*, and the interaction ends.

If the number of phases in the interaction is infinite, the prover has constructed an infinite schedule in which some processes remain active and, hence, the protocol is not wait-free. This is the third way that the prover can *win*.

To prove that a task is impossible using an extension-based proof, a prover must win against every protocol.

An example. We express the proof of the impossibility of deterministically solving wait-free binary consensus among two or more processes as a restricted extension-based proof.

Theorem 3.1. *Deterministic wait-free consensus among $n \geq 2$ processes is impossible in the NIIS model.*

Proof. Let C_0 denote the initial configuration in which p_0 has input 0 and p_1 has input 1. Then, by validity, the solo-execution by p_0 must decide $a_0 = 0$ and the solo-execution by p_1 must decide $1 - a_0 = 1$. The prover performs the query chain corresponding to the solo execution by p_0 from C_0. The prover wins if this does not terminate or p_0 does not output 0. Similarly, the prover performs the query chain corresponding to the solo execution by p_1 from C_0 and wins if this does not terminate or p_1 does not output 1.

The prover will either construct an infinite query chain in some phase, reach a configuration in which both 0 and 1 have been output, or inductively construct an infinite sequence of configurations $C_1, C_2, \ldots$ and a corresponding sequence of bits $a_1, a_2, \ldots$ such that, for all $i \geq 1$, C_i is reached from C_{i-1} by scheduling one set of processes (either $\{p_0\}$, $\{p_1\}$, or $\{p_0, p_1\}$), the solo-execution by p_0 from C_i outputs a_i, and the solo-execution by p_1 from C_i outputs $1 - a_i$. Let $i \geq 1$ and suppose the claim is true for $i - 1$.

If process p_0 has terminated (and output value a_{i-1}) in configuration C_{i-1}, then the solo execution by p_1 from C_{i-1}, which outputs $1 - a_{i-1}$, results in a configuration in which both 0 and 1 have been output. Similarly, if p_1 has terminated in configuration C_{i-1}, then the prover has reached a configuration in which both 0 and 1 have been output. So, suppose that neither p_0 nor p_1 has terminated in C_{i-1}.

From C_{i-1}, the prover first performs the query chain corresponding to the schedule $\{p_0\}, \{p_1\}, \{p_1\}, \ldots$ where p_0 is scheduled once and then p_1 is scheduled until it outputs a value b_i. If that never happens, then the prover wins. If $b_i = 1 - a_{i-1}$, then the prover ends phase i, chooses $C_i = C_{i-1}\{p_0\}$, and sets $a_i = a_{i-1}$. Note that the solo execution by p_0 from C_i outputs $a_i = a_{i-1}$ and the solo execution by p_1 from C_i outputs $1 - a_i = 1 - a_{i-1}$.

Otherwise, $b_i = a_{i-1}$. In this case, the prover performs the query chain from C_{i-1} corresponding to the schedule $\{p_1\}, \{p_0\}, \{p_0\}, \ldots$, where p_1 is scheduled once and then p_0 is scheduled until it outputs a value d_i. If that never happens, then the prover wins. If $d_i = a_{i-1}$, then the prover ends the round, chooses $C_i = C_{i-1}\{p_1\}$, and sets $a_i = a_{i-1}$. Note that the solo execution by p_0 from C_i outputs $a_i = a_{i-1}$ and the solo execution by p_1 from C_i outputs $1 - a_i = 1 - a_{i-1}$.

Otherwise, $d_i = 1 - a_{i-1}$. Then the prover performs the query $\{p_0, p_1\}$. Note that the configurations $C_{i-1}\{p_0, p_1\}$ and $C_{i-1}\{p_0\}\{p_1\}$ are indistinguishable to p_1, i.e. p_1 has the same state in both configurations. Thus, it outputs b_i in its solo execution from $C_{i-1}\{p_0, p_1\}$. Likewise, the configurations $C_{i-1}\{p_0, p_1\}$ and $C_{i-1}\{p_1\}\{p_0\}$ are indistinguishable to p_0, so it outputs d_i in its solo execution from $C_{i-1}\{p_0, p_1\}$. Finally, the prover ends the phase by choosing $C_i = C_{i-1}\{p_0, p_1\}$, and sets $a_i = d_i$. Note that the solo execution by p_0 from C_i outputs a_i and the solo execution by p_1 from C_i outputs $b_i = a_{i-1} = 1 - d_i = 1 - a_i$.

Thus, in all cases, the claim is true for i. Hence, by induction, the claim is true for all $i \geq 0$. □

4 WHAT CANNOT BE PROVED BY RESTRICTED EXTENSION-BASED PROOFS

In this section, we prove that no restricted extension-based proof can show the impossibility of deterministically solving k-set agreement in a wait-free manner in the NIIS model, for $n > k \geq 2$ processes. Observe that any protocol for $n > k + 1$ processes is also a protocol for $k + 1$ processes, since the remaining processes could crash before taking any steps. Therefore, it suffices to consider $n = k + 1$.

To show our result, we define an adversary that is able to win against every restricted extension-based prover. The adversary maintains a *partial* specification of Δ (the protocol it is adaptively constructing) and an integer $t \geq 0$. The integer t represents the number of non-uniform chromatic subdivisions of the input complex, $\mathbb{S}^0$, that it has performed. Once the adversary has defined Δ for each vertex in $\mathbb{S}^t$, then it may perform a non-uniform chromatic subdivision of $\mathbb{S}^t$ (or *subdivide* $\mathbb{S}^t$) and construct $\mathbb{S}^{t+1} = \chi(\mathbb{S}^t, \Delta)$.

For each $0 \leq r \leq t$ and each input value $a \in \{0, 1, \ldots, k\}$, we define $\mathbb{N}_a^r$ to be the subcomplex of $\mathbb{S}^r$ consisting of all subsets in $\mathbb{S}^r$ that only contain vertices representing states of processes which have not seen a (in any scan) and $\mathbb{T}_a^r$ to be the subcomplex of $\mathbb{S}^r$ consisting of all subsets in $\mathbb{S}^r$ that only contain vertices representing states of processes which have output a and terminated, i.e. vertices for which Δ is a. If a vertex v corresponds to the state of a process that has seen input a, then we say v *contains* a. Notice that $\mathbb{N}_a^t$ does not change when the adversary updates Δ, while $\mathbb{T}_a^t$ could possibly change. From these definitions, it follows that non-uniform chromatic subdivisions of these subcomplexes have simple descriptions.

Proposition 4.1. *For $t \geq 1$, $\mathbb{N}_a^t$ is non-empty and $\chi(\mathbb{N}_a^{t-1}, \Delta) = \mathbb{N}_a^t$. If $\mathbb{T}_a^t$ is non-empty, then $\chi(\mathbb{T}_a^{t-1}, \Delta) = \mathbb{T}_a^{t-1}$.*

One difficulty is ensuring that the processes do output incorrect values. To do so, the adversary terminates a process with output value a at a particular state only if the vertex v in $\mathbb{S}^t$ corresponding to its state contains a, i.e. the process has seen a, and v is sufficiently far from any vertex that has terminated with a different value. When the adversary performs a subdivision, the distance increases between terminated vertices that output different values. Hence, the adversary is able to terminate more vertices. This ensures that every chain of queries is finite. Finally, it can be shown that, if the adversary manages to ensure these conditions throughout the first phase, then the prover is doomed to lose. This is because the prover must commit to a non-empty schedule at the end of the first phase, which corresponds to some subcomplex $\mathbb{Q}$ of $\mathbb{S}^r$, for some $1 \leq r \leq t$. Since the outputs are far apart in the current subdivision $\chi^{t-r}(\mathbb{Q}, \Delta)$ of $\mathbb{Q}$, the adversary can terminate all active vertices in $\chi^{t-r}(\mathbb{Q}, \Delta)$. Hence, after a finite number of phases, the prover commits to an extension that results in a configuration (corresponding to a simplex in $\chi^{t-r}(\mathbb{Q}, \Delta)$) in which all processes have terminated.

Adversarial Strategy in Phase 1. We define an adversarial strategy so that, before and after each query made by the prover in phase 1, the adversary is able to maintain the following invariants:

(1) For each $0 \leq r < t$ and each vertex $v \in \mathbb{S}^r$, $\Delta(v)$ is defined. If v is a vertex in $\mathbb{S}^t$, then either $\Delta(v)$ is undefined or $\Delta(v) \neq \perp$. If s is the state of a process in a configuration that was reached by the prover, then $\{s\}$ is a vertex in $\mathbb{S}^r$, for some $0 \leq r \leq t$, and $\Delta(s)$ is defined.
(2) For any input a, if $\mathbb{T}_a^t$ is non-empty, then $\text{dist}_{\mathbb{S}^t}(\mathbb{T}_a^t, \mathbb{N}_a^t) \geq 2$.
(3) For any two inputs $a \neq b$, if $\mathbb{T}_a^t$ and $\mathbb{T}_b^t$ are non-empty, then $\text{dist}_{\mathbb{S}^t}(\mathbb{T}_a^t, \mathbb{T}_b^t) \geq 3$.

We note that there is nothing special about the values 2 and 3. They are simply the smallest values such that the invariant can be maintained and every chain of queries is finite.

The following lemma is a consequence of the invariants. In particular, it says that, after a subdivision, the distance between vertices that output different values increases and the distance between vertices that output a and vertices that do not contain a increases.

Lemma 4.2. *For any two inputs $a \neq b$, if $\mathbb{T}_a^t$ is non-empty, then any path between $\mathbb{T}_a^t$ and $\mathbb{T}_b^t \cup \mathbb{N}_a^t$ in $\mathbb{S}^t$ contains at least one edge between active vertices. Moreover, if the adversary defines $\Delta(v) = \perp$ for each vertex $v \in \mathbb{S}^t$ where $\Delta(v)$ is undefined, subdivides $\mathbb{S}^t$ to construct $\mathbb{S}^{t+1}$, and $\mathbb{T}_b^t$ is non-empty, then $\text{dist}_{\mathbb{S}^{t+1}}(\mathbb{T}_a^{t+1}, \mathbb{T}_b^{t+1}) \geq \text{dist}_{\mathbb{S}^t}(\mathbb{T}_a^t, \mathbb{T}_b^t) + 1$ and $\text{dist}_{\mathbb{S}^{t+1}}(\mathbb{T}_a^{t+1}, \mathbb{N}_a^{t+1}) \geq \text{dist}_{\mathbb{S}^t}(\mathbb{T}_a^t, \mathbb{N}_a^t) + 1$.*

Proof. Consider any path $v_0, v_1, \ldots, v_\ell$ between $\mathbb{T}_a^t$ and $\mathbb{T}_b^t \cup \mathbb{N}_a^t$ in $\mathbb{S}^{t_0}$. Let v_j be the last vertex in $\mathbb{T}_a^t$. Since the invariants hold after each query and $v_\ell \in \mathbb{T}_b^t \cup \mathbb{N}_a^t$, invariants (3) and (2) imply that the distance between v_j and v_ℓ is at least 2. Hence, $\ell \geq j + 2$. Since v_j is the last vertex in $\mathbb{T}_a^t$, v_{j+1} and v_{j+2} are not in $\mathbb{T}_a^t$. Moreover, by invariant (3), v_{j+1} and v_{j+2} are not in $\mathbb{T}_c^t$ for any input $c \neq a$. Hence, $\{v_{j+1}, v_{j+2}\}$ is an edge between active vertices.

Suppose the adversary defines $\Delta(v) = \perp$ for each vertex $v \in \mathbb{S}^t$ where $\Delta(v)$ is undefined and subdivides $\mathbb{S}^t$ to construct $\mathbb{S}^{t+1}$. Then, by Proposition 4.1, for each input a, $\mathbb{T}_a^t = \chi(\mathbb{T}_a^t, \Delta)$ and $\mathbb{N}_a^{t+1} = \chi(\mathbb{N}_a^t, \Delta)$. Since the adversary does not terminate any new vertices, $\mathbb{T}_a^{t+1} = \mathbb{T}_a^t$. Therefore, by the second part of Lemma 2.3, $\text{dist}_{\mathbb{S}^{t+1}}(\mathbb{T}_a^{t+1}, \mathbb{T}_b^{t+1}) \geq \text{dist}_{\mathbb{S}^t}(\mathbb{T}_a^t, \mathbb{T}_b^t)+1$ and $\text{dist}_{\mathbb{S}^{t+1}}(\mathbb{T}_a^{t+1}, \mathbb{N}_a^{t+1}) \geq \text{dist}_{\mathbb{S}^t}(\mathbb{T}_a^t, \mathbb{N}_a^t) + 1$. □

We now describe the adversarial strategy. Initially, the adversary sets $\Delta(v) = \perp$ for each vertex $v \in \mathbb{S}^0$. It then subdivides $\mathbb{S}^0$ to construct $\mathbb{S}^1$ and sets $t = 1$. No vertices in $\mathbb{S}^0$ have terminated, so $\mathbb{T}_a^0$ is empty for each input a. Before the first query, the prover has only reached initial configurations. Hence, the invariants are satisfied.

Now suppose the invariants are satisfied immediately prior to a query (C, P) by the prover, where C is a configuration previously reached by the prover and P is a set of active processes in C poised to access the same snapshot object. Let D be the configuration resulting from scheduling P from C. Since each process in P is poised to access the same snapshot object, by invariant (1), there exists $0 \leq r \leq t$ such that the state of each process in P in configuration C corresponds to a vertex in $\mathbb{S}^r$. Since each process in P is active in C, $\Delta(v) = \perp$ for each such vertex v. Hence, by invariant (1), $r < t$. If $r < t - 1$, then invariant (1) implies that Δ is defined for each vertex corresponding to the state of a process in D. Hence, the adversary does not need to do anything.

So, suppose that $r = t - 1$. Let σ denote the simplex in $\mathbb{S}^t$ whose vertices represent the states of processes in P in D. For each vertex $v \in \sigma$, if $\Delta(v)$ is undefined, the adversary defines $\Delta(v)$ as follows. If there exists an input a such that the distance between v and $\mathbb{N}_a^t$ in $\mathbb{S}^t$ is at least 2 and the distance between v and $\mathbb{T}_b^t$ in $\mathbb{S}^t$ is at least 3, for all inputs $b \neq a$, then the adversary sets $\Delta(v) = a$. Otherwise, the adversary sets $\Delta(v) = \perp$. This ensures that invariants (2) and (3) continue to hold.

If $\Delta(v) \neq \perp$ for every vertex $v \in \sigma$, then invariant (1) holds. Otherwise, the adversary defines $\Delta(v) = \perp$ for each vertex $v \in \mathbb{S}^t$ where $\Delta(v)$ is undefined, subdivides $\mathbb{S}^t$ to construct $\mathbb{S}^{t+1}$, and increments t. Then invariant (1) holds. By Lemma 4.2, invariants (2) and (3) continue to hold.

Therefore, the invariants hold after the prover's query.

The prover does not win in phase 1. Suppose that the invariants all hold before and after each query made by the prover in phase 1. By invariant (3), at most one value is output in any configuration reached by the prover. Moreover, by invariant (2), if a process outputs a value y, then y is a value that it has seen in one of its scans. Hence, y is the input of some process. So, the prover cannot win in phase 1 by showing that the protocol violates agreement or validity. It remains to show that the prover cannot win by constructing an infinite chain of queries in phase 1.

Lemma 4.3. *Every chain of queries in phase 1 is finite.*

Proof. Assume, for a contradiction, that there is an infinite chain of queries, (C_j, P_j), for $j \geq 0$. Let P be the set of processes that are scheduled infinitely often. Then, there exists $j_0 \geq 0$ such that, for all $j \geq j_0$, $P_j \subseteq P$. Let $t_0 \geq 1$ be the value of t held by the adversary immediately prior to query (C_{j_0}, P_{j_0}). By invariant (1), the state of each process in C_{j_0} corresponds to a vertex $v \in \mathbb{S}^r$, for some $0 \leq r \leq t_0$. Hence, no process has accessed S_{t_0+1} in C_{j_0} and, during this chain of queries, only processes in P access S_t for $t > t_0$. Since the processes in P eventually access S_{r+1} for all $r \geq t_0$, and no process in P ever terminates, the adversary eventually defines $\Delta(v) = \perp$ for each vertex $v \in \mathbb{S}^r$ where $\Delta(v)$ is undefined and subdivides $\mathbb{S}^r$ to construct $\mathbb{S}^{r+1}$, for all $r \geq t_0$.

Consider the first $j_1 \geq j_0$ such that each process in P_{j_1} is poised to access S_{t_0+2} in C_{j_1}, i.e. P_{j_1} is the first set of processes to access S_{t_0+2} in the chain of queries. By definition of $\mathbb{S}^{t_0+2}$, the set of states of the processes in P_{j_1} in C_{j+1} correspond to a simplex σ_1 in $\mathbb{S}^{t_0+2}$. Since the adversary does not terminate any new processes, $\mathbb{T}_a^t = \mathbb{T}_a^{t_0}$, for any input a and any $t \geq t_0$. Thus, by applying Lemma 4.2 twice, for any inputs $a \neq b$, whenever $\mathbb{T}_a^{t_0}$ and $\mathbb{T}_b^{t_0}$ are non-empty, $\text{dist}_{\mathbb{S}^{t_0+2}}(\mathbb{T}_a^{t_0+2}, \mathbb{T}_b^{t_0+2}) \geq 5$ and $\text{dist}_{\mathbb{S}^{t_0+2}}(\mathbb{T}_a^{t_0+2}, \mathbb{N}_a^{t_0+2}) \geq 4$.

If there is a vertex $v \in \sigma_1$ that has distance at most 2 to some vertex in $\mathbb{T}_a^{t_0+2}$ in $\mathbb{S}^{t_0+2}$, for some input a, then the distance from v

to $\mathbb{N}_a^{t_0+1}$ is at least 2 and the distance from v to non-empty $\mathbb{T}_b^{t_0+1}$ in $\mathbb{S}^{t_0+1}$ is at least 3, for all $b \neq a$. Hence the adversary defines $\Delta(v) \neq \perp$ after query (C_{j_1}, P_{j_2}), i.e. some process in $P_{j_1} \subseteq P$ terminates. This is a contradiction.

So, each vertex in σ_1 has distance at least 3 to $\mathbb{T}_a^{t_0+2}$, for all inputs a where $\mathbb{T}_a^{t_0+2}$ is non-empty. Consider the first $j_2 > j_1$ such that each process in P_{j_2} is poised to access S_{t_0+3} in C_{j_2}. Let P' be the set of processes that have accessed S_{t_0+2} in C_{j_2}. Since each process in $P_{j_1} \cup P_{j_2}$ has already accessed S_{t_0+2}, $P_{j_1} \cup P_{j_2} \subseteq P'$. Hence, the states of P' in C_{j_2} forms a simplex σ_2 in $\mathbb{S}^{t_0+2}$ and $\sigma_1 \subseteq \sigma_2$. Since every vertex in σ_2 has distance at most 1 from every vertex in σ_1, it follows that each vertex in σ_2 has distance at least 2 to $\mathbb{T}_a^{t_0+2}$, for all inputs a where $\mathbb{T}_a^{t_0+2}$ is non-empty.

Let a be the input of any process in P_{j_1}. Since P_{j_1} is the first set of processes to access S_{t_0+2} and each process in P' has accessed S_{t_0+2}, each vertex in σ_2 contains a and $\text{dist}_{\mathbb{S}^{t_0+2}}(\sigma_2, \mathbb{N}_a^{t_0+2}) \geq 1$. Since the distance between σ_2 and any terminated vertex in $\mathbb{S}^{t_0+2}$ is at least 2, any path from a vertex $v \in \sigma_2$ to a vertex in $\mathbb{N}_a^{t_0+1} \cup \bigcup_{b \neq a} \mathbb{T}_b^{t_0+1}$ must contain at least one edge between active vertices (specifically, between v and one of its neighbours). By Lemma 2.3 and Proposition 4.1, it follows that $\text{dist}_{\mathbb{S}^{t_0+3}}(\chi(\sigma_2, \Delta), \mathbb{N}_a^{t_0+3}) \geq 2$ and $\text{dist}_{\mathbb{S}^{t_0+3}}(\chi(\sigma_2, \Delta), \mathbb{T}_b^{t_0+3}) \geq 3$, for any input $b \neq a$ where $\mathbb{T}_b^{t_0+2} = \mathbb{T}_b^{t_0+3}$ is non-empty. The state of each process in P_{j_2} in C_{j_2+1} corresponds to a vertex v in $\chi(\sigma_2, \Delta)$. Hence, the adversary defines $\Delta(v) \neq \perp$ for at least one such vertex v, i.e. some process in $P_{j_2} \subseteq P$ terminates. This is a contradiction. □

The prover loses. Since the prover does not win in phase 1, eventually phase 1 must end. At the end of phase 1, the prover must choose a configuration $C \in \mathscr{A}'(1)$. This determines the set of configurations, $\mathscr{A}(2)$, that the prover can initially query in phase 2. The adversary will update Δ one final time. Afterwards, it can answer all future queries by the prover. The prover will eventually be forced to choose a terminal configuration at the end of some future phase and, consequently, will lose in the next phase.

C is a configuration reached by a non-empty schedule $\alpha(2)$ from an initial configuration $C_0 \in \mathscr{A}(1)$. Let P be the first set of processes in $\alpha(2)$ and let a be the input of some process $p \in P$ at C_0. By invariant (1), the state of each process in P at $C_0 P$ is represented by a vertex in $\mathbb{S}^1$. Since C is a reachable configuration, there is a simplex $\sigma \subseteq \mathbb{S}^1$ with these vertices.

Let $\mathbb{Q}$ be the subcomplex of $\mathbb{S}^1$ consisting of all subsets in $\mathbb{S}^1$ that only contain vertices which are at distance at most 1 from every vertex in σ. Then $\mathbb{Q}$ contains every simplex corresponding to a configuration reachable by a full 1-round schedule α' from some initial configuration C_0' such that $C_0'\alpha'$ and $C_0 P$ are indistinguishable to P. In particular, P is the first set of processes in α' and p has input a in C_0'. It follows that the state of every process contains a in $C_0'\alpha'$. Hence, the distance between $\mathbb{Q}$ and $\mathbb{N}_a^1$ in $\mathbb{S}^1$ is at least 1.

Consider the value of t held by the adversary at the end of phase 1. If $t > 1$, then, applying Lemma 2.3 and Proposition 4.1 $(t-1)$ times, it follows that $\text{dist}_{\mathbb{S}^t}(\chi^{t-1}(\mathbb{Q}, \Delta), \mathbb{N}_a^t) \geq 1$, where $\chi^{t-1}(\mathbb{Q}, \Delta) \subseteq \mathbb{S}^t$ denotes the simplicial complex obtained by performing $t-1$ non-uniform chromatic subdivisions of $\mathbb{Q}$.

By invariant (1), for each vertex $v \in \mathbb{S}^t$, $\Delta(v)$ is either undefined or $\Delta(v) \neq \perp$. By invariant (3), each n-vertex simplex in $\mathbb{S}^t$ represents a configuration in which all processes that have terminated have output the same value. For each vertex $v \in \chi^{t-1}(\mathbb{Q}, \Delta)$ where $\Delta(v)$ is undefined, the adversary sets $\Delta(v) = a$. This is a valid output for v since $\text{dist}_{\mathbb{S}^t}(\chi^{t-1}(\mathbb{Q}, \Delta), \mathbb{N}_a^t) \geq 1$. Then each vertex in $\chi^{t-1}(\mathbb{Q}, \Delta)$ has terminated. Morever, each n-vertex simplex in $\mathbb{S}^t$ represents a reachable configuration in which the processes have output at most $2 \leq k$ different values.

In phases $\varphi \geq 2$, the prover can only query configurations reachable from some configuration in $\mathscr{A}(2)$. By definition, $\mathscr{A}(2)$ is the set of all configurations that are reached by performing $\alpha(2)$ from initial configurations C_0' that only differ from C_0 in the inputs of processes that do not appear in $\alpha(2)$. It follows that, for any process q and any extension α' of $\alpha(2)$ from $C' \in \mathscr{A}(2)$, q appears at most t times in $\alpha(2)\alpha'$ before its state is represented by a vertex in $\chi^{t-1}(\mathbb{Q}, \Delta)$. Note that, by construction, every vertex in $\chi^{t-1}(\mathbb{Q}, \Delta)$ has terminated. Thus, eventually, the prover chooses a configuration at the end of some phase in which every process has terminated. The prover loses in the next phase.

5 EXTENSION-BASED PROOFS

In this section, we extend the definition of a restricted extension-based proof to include output queries, explain how the adversarial protocol can respond to these queries, and extend the proof in Section 4. Roughly speaking, output queries allow the prover to perform a valency argument.

An *output* query in phase φ is specified by a configuration $C \in \mathscr{A}(\varphi) \cup \mathscr{A}'(\varphi)$, a set of active processes P in C that are poised to access the same snapshot object, and a value $y \in \{0, 1, \ldots, k\}$. If there is a schedule from C involving only processes in P, i.e. a P*-only* schedule, that results in a configuration in which some process in P outputs y, then the protocol returns some such schedule. Otherwise, the protocol returns NONE. In each phase, the prover is allowed to make finitely many output queries in addition to finitely many chains of queries.

For example, if P is the set of all processes, then the sequence of output queries $(C, P, 0), (C, P, 1), \ldots, (C, P, k)$ enables the prover to determine which values can be output by the processes when they are scheduled starting from C. In particular, the prover can determine if it is only possible to output one value starting from C, i.e. if C is *univalent*.

Responding to output queries in phase 1. Suppose that the invariants hold prior to an output query (C, P, y) in phase 1. We show that the adversary can answer the output query so that it never conflicts with the result of any future query made in phase 1, while still maintaining the invariants.

By definition, each process in P is poised to access the same snapshot object S_r in C, for some $r \geq 0$. By invariant (1), $r \leq t$, where t is the value held by the adversary immediately prior to the query. Let $\mathbb{V}$ be the subcomplex of $\mathbb{S}^t$ consisting of all subsets in $\mathbb{S}^t$ that only contain vertices representing the state of a process in P in a configuration C' reachable from C by a P-only $(t-r)$-round schedule from C. In particular, $\mathbb{V}$ contains the possible states of each process in P after it has been selected t times (or it has terminated) in some configuration C' reachable from C by a P-only schedule.

If some vertex $v \in \mathbb{V}$ has terminated with output y, then the adversary returns a P-only $(t-r)$-round schedule from C that leads to a configuration C' such that v represents the state of a process in C'. If every vertex in $\mathbb{V}$ has terminated and none have output y, then the adversary returns NONE. Since the adversary never changes $\Delta(v)$ once it has been set, these responses do not conflict with the result of any future query. In both cases, the invariants continue to hold.

Now suppose that no vertex in $\mathbb{V}$ has terminated with output y and $\mathbb{V}$ contains at least one vertex that has not terminated. Let $\mathbb{U} \neq \emptyset$ be the subcomplex of $\mathbb{V}$ consisting of all subsets in $\mathbb{V}$ that only contain vertices that have not terminated (i.e. Δ is currently undefined for these vertices). For each simplex $\sigma \subseteq \mathbb{U}$, let $\mathbb{A}_\sigma$ be the subcomplex of $\mathbb{S}^t$ consisting of all subsets in $\mathbb{S}^t$ that only contain vertices at distance at most 1 to each vertex in σ. We consider a number of cases.

Case 1: *There is a simplex $\sigma \subseteq \mathbb{U}$ such that $\sigma \not\subseteq \mathbb{N}_y^t$ and no vertex in $\mathbb{A}_\sigma$ has terminated.* Then the adversary subdivides $\mathbb{S}^t$ twice, i.e. it defines $\Delta(v) = \bot$ for each $v \in \mathbb{S}^t$ where $\Delta(v)$ is undefined, subdivides $\mathbb{S}^t$ to construct $\mathbb{S}^{t+1}$, defines $\Delta(v) = \bot$ for each $v \in \mathbb{S}^{t+1}$ where $\Delta(v)$ is undefined, and subdivides $\mathbb{S}^{t+1}$ to construct $\mathbb{S}^{t+2}$.

Consider the simplex $\rho \subseteq \chi(\sigma, \Delta)$ consisting of all subsets of $\{(i, \vec{\sigma}) : i \in Id(\sigma)\}$. Since $\sigma \not\subseteq \mathbb{N}_y^t$, some vertex in σ contains y, so each vertex in ρ contains y. Hence, $\text{dist}_{\mathbb{S}^{t+1}}(\rho, \mathbb{N}_y^{t+1}) \geq 1$. Since each vertex in $\mathbb{A}_\sigma$ is active, $\chi(\mathbb{A}_\sigma, \Delta)$ is the standard chromatic subdivision of $\mathbb{A}_\sigma$. It follows that every path between ρ and $\mathbb{T}_a^{t+1} \cup \mathbb{N}_y^{t+1}$ in $\mathbb{S}^{t+1}$, for inputs $a \neq y$, contains at least one edge between a vertex in ρ and one of its neighbours in $\chi(\mathbb{A}_\sigma, \Delta)$, which is an edge between active vertices. Since no vertex in $\mathbb{T}_a^{t+1}$ is active, $\text{dist}_{\mathbb{S}^{t+1}}(\rho, \mathbb{T}_a^{t+1}) \geq 2$, for all inputs a. By Lemma 2.3 and Proposition 4.1, it follows that $\text{dist}_{\mathbb{S}^{t+2}}(\chi(\rho, \Delta), \mathbb{N}_y^{t+2}) \geq 2$ and $\text{dist}_{\mathbb{S}^{t+2}}(\chi(\rho, \Delta), \mathbb{T}_a^{t+2}) \geq 3$, for all inputs a.

The adversary defines $\Delta(v) = y$, for one vertex $v \in \chi(\rho, \Delta)$, returns a P-only $(t + 2 - r)$-round schedule from C that leads to a configuration C' such that v represents the state of a process in C', and sets t to $t + 2$. By Lemma 4.2, invariants (2) and (3) are not violated by the subdivisions and increments of t. Since $\Delta(v) \neq \bot$, invariant (1) continues to hold and, since $\text{dist}_{\mathbb{S}^t}(v, \mathbb{N}_y^t) \geq 2$ and $\text{dist}_{\mathbb{S}^t}(v, \mathbb{T}_a^t) \geq 3$, for all inputs $a \neq y$, invariants (2) and (3) continue to hold.

Case 2: *There is a simplex $\sigma \subseteq \mathbb{U}$ such that $\sigma \not\subseteq \mathbb{N}_y^t$ and there is a vertex $w \in \mathbb{A}_\sigma$ such that w has terminated with output y.* Then the adversary subdivides $\mathbb{S}^t$ once, i.e. it defines $\Delta(v) = \bot$ for each vertex $v \in \mathbb{S}^t$ where $\Delta(v)$ is undefined and subdivides $\mathbb{S}^t$ to construct $\mathbb{S}^{t+1}$. Note that $w \in \mathbb{T}_y^t = \mathbb{T}_y^{t+1}$, so $\mathbb{T}_y^t$ is non-empty. By invariant (2) and Lemma 4.2, $\text{dist}_{\mathbb{S}^{t+1}}(\mathbb{T}_y^{t+1}, \mathbb{N}_y^{t+1}) \geq 3$. By invariant (3) and Lemma 4.2, $\text{dist}_{\mathbb{S}^{t+1}}(\mathbb{T}_y^{t+1}, \mathbb{T}_a^{t+1}) \geq 4$ for all inputs $a \neq y$ such that $\mathbb{T}_a^t$ is non-empty. Since w has terminated and $w \in \mathbb{A}_\sigma$, by Proposition 2.1, w is adjacent to every vertex in $\chi(\sigma, \Delta) \subseteq \mathbb{S}^{t+1}$. It follows that $\text{dist}_{\mathbb{S}^{t+1}}(\chi(\sigma, \Delta), \mathbb{N}_y^{t+1}) \geq 2$ and $\text{dist}_{\mathbb{S}^{t+1}}(\chi(\sigma, \Delta), \mathbb{T}_a^{t+1}) \geq 3$, for all inputs $a \neq y$ such that $\mathbb{T}_a^{t+1} = \mathbb{T}_a^t$ is non-empty.

The adversary defines $\Delta(v) = y$, for a vertex $v \in \chi(\sigma, \Delta)$, returns a P-only $(t + 1 - r)$-round schedule from C that leads to a configuration C' such that v represents the state of a process in C', and increments t. By Lemma 4.2, invariants (2) and (3) are not violated by the subdivision and increment of t. Since $\Delta(v) \neq \bot$, invariant (1) continues to hold and, since $\text{dist}_{\mathbb{S}^t}(v, \mathbb{N}_y^t) \geq 2$ and $\text{dist}_{\mathbb{S}^t}(v, \mathbb{T}_a^t) \geq 3$, for all inputs $a \neq y$ such that $\mathbb{T}_a^t$ is non-empty, invariants (2) and (3) continue to hold.

Case 3. *For every simplex $\sigma \subseteq \mathbb{U}$, either $\sigma \subseteq \mathbb{N}_y^t$ or some vertex $w \in \mathbb{A}_\sigma$ has terminated with an output $a \neq y$.* In this case, the adversary returns NONE. If $\sigma \subseteq \mathbb{N}_y^t$, then no vertex in σ contains y and, by Proposition 4.1, no vertex in any subdivision of σ contains y. Hence, by invariant (2), the adversary never terminates any vertex in σ (or a future subdivision of σ) with output y as the result of a future query (in any phase). If some vertex $w \in \mathbb{A}_\sigma$ has terminated with output $a \neq y$, then, since w is adjacent to each vertex in σ, w is adjacent to each vertex in any subdivision of σ (Proposition 2.1). Since invariant (3) holds, the adversary never terminates such vertices with any output other than a as the result of a future query in phase 1. In both cases, the invariants continue to hold.

The prover still loses. Since the adversary is able to maintain the invariants before and after each query, as in Section 4, the prover does not win in phase 1 and must eventually choose a configuration $C \in \mathscr{A}'(1)$ at the end of phase 1. The adversary will update Δ one final time. Afterwards, it can answer all future queries by the prover. The prover will eventually be forced to choose a terminal configuration at the end of some future phase and, consequently, will lose in the next phase.

C is a configuration reached by a non-empty schedule $\alpha(2)$ from an initial configuration $C_0 \in \mathscr{A}(1)$. Let P be the first set of processes in $\alpha(2)$ and let a be the input of some process $p \in P$ at C_0. By invariant (1), the state of each process in P at C_0P is represented by a vertex in $\mathbb{S}^1$. Since C is a reachable configuration, there is a simplex $\sigma \subseteq \mathbb{S}^1$ with these vertices.

Let $\mathbb{Q}$ be the subcomplex of $\mathbb{S}^1$ consisting of all subsets in $\mathbb{S}^1$ that only contain vertices which are at distance at most 1 from every vertex in σ. Then $\mathbb{Q}$ contains every simplex corresponding to a configuration reachable by a full 1-round schedule α' from some initial configuration C_0' such that $C_0'\alpha'$ and C_0P are indistinguishable to P. In particular, P is the first set of processes in α' and p has input a in C_0'. It follows that the state of every process contains a in $C_0'\alpha'$. Hence, the distance between $\mathbb{Q}$ and $\mathbb{N}_a^1$ in $\mathbb{S}^1$ is at least 1.

Consider the value of t held by the adversary at the end of phase 1. If $t > 1$, then, applying Lemma 2.3 and Proposition 4.1 $(t-1)$ times, it follows that $\text{dist}_{\mathbb{S}^t}(\chi^{t-1}(\mathbb{Q}, \Delta), \mathbb{N}_a^t) \geq 1$, where $\chi^{t-1}(\mathbb{Q}, \Delta) \subseteq \mathbb{S}^t$ denotes the simplicial complex obtained by performing $t-1$ non-uniform chromatic subdivisions of $\mathbb{Q}$.

To ensure that the prover loses, we have to modify the adversary's strategy at the end of phase 1 in Section 4 slightly. In particular, when the adversary defines $\Delta(v)$ for each vertex in $\chi^{t-1}(\mathbb{Q}, \Delta)$, it cannot simply set $\Delta(v) = a$, where a is an input value contained in each vertex of $\chi^{t-1}(\mathbb{Q}, \Delta)$. The problem is that the adversary may have answered NONE to an output query (C, P, a) where not all vertices have terminated. This can only occur in Case 3. Instead, the adversary does the following.

(1) It subdivides $\mathbb{S}^t$ twice to construct $\mathbb{S}^{t+2}$ and sets t to $t + 2$.
(2) Let T be the set of all terminated vertices in $\mathbb{S}^t$. For each terminated vertex $w \in T$ and each vertex $v \in \chi^{t-1}(\mathbb{Q}, \Delta) \subseteq \mathbb{S}^t$ that is adjacent to w, if $\Delta(v)$ is undefined, then it sets $\Delta(v) = \Delta(w)$.

(3) For each vertex $v \in \chi^{t-1}(\mathbb{Q}, \Delta)$, if $\Delta(v)$ is undefined, then it sets $\Delta(v) = a$.

Immediately before step (1), invariants (3) and (2) imply that $\text{dist}_{\mathbb{S}^t}(\mathbb{T}_b^t, \mathbb{T}_d^t) \geq 3$ and $\text{dist}_{\mathbb{S}^t}(\mathbb{T}_b^t, \mathbb{N}_b^t) \geq 2$ for any inputs $b \neq d$ such that $\mathbb{T}_b^t$ and $\mathbb{T}_d^t$ are non-empty. By Lemma 4.2, immediately after step (1), $\text{dist}_{\mathbb{S}^t}(\mathbb{T}_b^t, \mathbb{T}_d^t) \geq 5$ and $\text{dist}_{\mathbb{S}^t}(\mathbb{T}_b^t, \mathbb{N}_b^t) \geq 4$ for any inputs $b \neq d$ such that $\mathbb{T}_b^t$ and $\mathbb{T}_d^t$ are non-empty. Consequently, immediately after step (1), if a vertex $v \in \mathbb{S}^t$ is adjacent to a vertex $w \in \mathbb{T}_b^t$, then it is not adjacent to any vertex in $\mathbb{T}_d^t$, for $d \neq b$.

Since invariant (1) holds immediately before step (1) and the adversary does not set $\Delta(v) = \perp$ for any vertex $v \in \mathbb{S}^t$ during step (2), invariant (1) continues to hold immediately after step (2).

Consider any vertex $u \in \chi^{t-1}(\mathbb{Q}, \Delta)$ such that the adversary sets $\Delta(u) = b$ in step (2). Then u is adjacent to a vertex $w \in \mathbb{T}_b^t \cap T$. Since $\text{dist}_{\mathbb{S}^t}(w, \mathbb{N}_b^t) \geq 4$, it follows that $\text{dist}_{\mathbb{S}^t}(u, \mathbb{N}_b^t) \geq 3$ and b is a valid output for u. Thus invariant (2) holds immediately after step (2).

Now consider any vertex $v \in \mathbb{T}_d^t$, where $d \neq b$. If $v \notin T$, then v is adjacent to a vertex $w' \in \mathbb{T}_d^t \cap T$. Since $\text{dist}_{\mathbb{S}^t}(w, w') \geq 5$, it follows that $\text{dist}_{\mathbb{S}^t}(u, v) \geq 3$. If $v \in T$, then $\text{dist}_{\mathbb{S}^t}(w, v) \geq 5$ and, hence, $\text{dist}_{\mathbb{S}^t}(u, v) \geq 4$. Thus invariant (3) holds immediately after step (2).

Next, we show that the changes to Δ do not violate the response to any output query made in phase 1.

Lemma 5.1. *No output queries made in phase 1 are ever violated.*

Proof. Let (C, P, y) be an output query made in phase 1. By definition, each process in P is poised to access the same snapshot object S_r in C, for some $r \geq 1$. By invariant (1), $r \leq t$, where t is the value held by the adversary immediately before the output query. Let $\mathbb{V}$ be the subcomplex of $\mathbb{S}^t$ consisting of all subsets in $\mathbb{S}^t$ that only contain vertices which represent the state of a process in P in a configuration C' reachable from C by a P-only $(t-r)$-round schedule from C. Let $\mathbb{U}$ be the subcomplex of $\mathbb{V}$ consisting of all subsets in $\mathbb{V}$ that only contain vertices that have not terminated.

If the adversary answers with a P-only schedule α, then there is a vertex $v \in \mathbb{S}^{t'}$ that represents the state of a process in $C\alpha$ such that $\Delta(v) = y$, where $t' \geq t$ is the value held by the adversary immediately after the output query. Since the adversary never changes $\Delta(v)$ once it has been defined, the adversary will never violate its response to the output query.

If every vertex in $\mathbb{V}$ has terminated and none have output y, then the adversary answers none. In this case, it will never violate its response to the output query, because it never changes $\Delta(v)$ for any vertex v once it has been defined.

The remaining case is when the adversary answers none and for every simplex σ in $\mathbb{U}$, either σ is contained in $\mathbb{N}_y^t$ or there is a vertex in $\mathbb{A}_\sigma$ that has terminated with an output $b \neq y$, where $\mathbb{A}_\sigma$ is the subcomplex of $\mathbb{S}^t$ consisting of all subsets in $\mathbb{S}^t$ that only contain vertices at distance at most 1 from every vertex in σ. Consider any simplex $\sigma \subseteq \mathbb{U}$ and its subdivision σ' in the complex constructed by the adversary at the end of phase 1. We show that the adversary never terminates any vertex in σ' with output y. Hence it does not violate its response to the output query (C, P, y).

Case 1. $\sigma \subseteq \mathbb{N}_y^t$. Then no vertex in σ contains y and, hence, no vertex $v \in \sigma'$ contains y. If the adversary set $\Delta(v) \neq \perp$ prior to step (3) at the end of phase 1, then, by invariant (2), $\Delta(v) \neq y$. If the adversary sets $\Delta(v) = a$ in step (3), then a is contained in v (by construction) and, hence, $\Delta(v) \neq y$. Note the adversary never modifies Δ after step (3).

Case 2. $\sigma \not\subseteq \mathbb{N}_y^t$. Then $\mathbb{A}_\sigma$ contains a vertex w that has terminated with $\Delta(w) \neq y$. Since w has terminated and is adjacent to every vertex in σ, it follows by Proposition 2.1 that w is adjacent to every vertex $v \in \sigma'$. If the adversary set $\Delta(v) \neq \perp$ prior to step (3) at the end of phase 1, then, by invariant (3), $\Delta(v) = \Delta(w) \neq y$. Notice that, since v is adjacent to $w \in \mathbb{T}_y^t$, if $\Delta(v)$ is not set in step (2), then it is not set in step (3). □

Since no output query is violated and the invariants hold prior to step (3), the rest of the argument in Section 4 is unchanged. In particular, the prover must commit to a terminal configuration in $\chi^{t-1}(\mathbb{Q}, \Delta)$ at the end of some phase and, hence, loses in the next phase.

Thus, we have shown that there is no extension-based proof of the impossibility of deterministically solving k-set agreement in a wait-free manner in the NIIS model for $n > k \geq 2$ processes. Since there is a deterministic, wait-free protocol for a task in the NIIS model if and only if there is a deterministic, wait-free protocol for the task using only registers, we have proved our main result.

Theorem 5.2. *No extension-based proof can show the impossibility of a deterministic, wait-free protocol for k-set agreement among $n > k \geq 2$ processes using only registers.*

6 FUTURE WORK

We have shown the limitation of valency arguments for proving the impossibility of deterministic, wait-free solutions to set-agreement in the NIIS model. Although we have restricted attention to the proof of impossibility of one problem in one model, our approach can be applied to other problems and other models.

The definition of an extension-based proof can be modified to handle other termination conditions, such as obstruction-freedom [HLM03]. It suffices for the prover to construct a schedule that violates this condition.

The NIIS model is computationally equivalent to an asynchronous shared memory model in which processes communicate by reading from and writing to shared registers. However, these two models are not equivalent in terms of space and step complexities. A covering argument [BL93] is a standard approach for proving a lower bound on the number of registers needed to solve a problem in an asynchronous system. We have a definition for extension-based proofs in this model that includes covering arguments.

There is a recent proof that any obstruction-free protocol for k-set agreement among $n > k \geq 2$ processes requires $\lceil n/k \rceil$ registers [EGZ18], but it is not extension-based. In fact, some of our early work about extension-based proofs motivated the approach in [EGZ18] We conjecture that it impossible to prove a non-constant lower bound on the number of registers needed by any obstruction-free protocol for k-set agreement using an extension-based proof.

Combinatorial topology has also been successfully applied to show that it is impossible to deterministically solve $(2n-2)$-renaming among n processes using only registers, whenever n is a prime power, i.e. $n = p^{\ell}$, for some prime p and integer ℓ [CR10]. In this problem, processes begin with identifiers from an arbitrarily large domain and must compute identifiers in the range $\{0, 1, \ldots, 2n-3\}$. There are no extension-based proofs of this result. We conjecture that extension-based proofs cannot prove this impossibility result.

We have considered allowing the prover to perform a number of other types of queries and can extend our adversarial protocol so that it can answer them. For example, if a prover asks the same output query (C, P, y) multiple times, the protocol could be required to return different schedules each time, until it has returned all possible P-only schedules from C that output y.

We cannot allow certain queries, such as asking for an upper bound on the length of any schedule. If the prover is given such an upper bound, then it can perform a finite number of chain queries to examine all reachable configurations, thereby fixing the protocol. However, we can allow the prover to use this information in a restricted way and still construct an adversarial k-set agreement protocol. For example, we might require that the prover does not use this information to decide which queries to perform or what extensions to construct, but can use this information to win when it has constructed a schedule that is longer than this upper bound.

7 ACKNOWLEDGMENTS

We would like to thank Toniann Pitassi and Michael Saks for helpful discussions and the anonymous reviewers for their comments. Support is gratefully acknowledged from the Natural Science and Engineering Research Council of Canada, a University of Toronto Postdoctoral Fellowship, the Simons Institute, the National Science Foundation under grants CCF-1217921, CCF-1301926, CCF-1637385, CCF-1650596, and IIS-1447786, the Department of Energy under grant ER26116/DE-SC0008923, and the Oracle and Intel corporations.

REFERENCES

[Abr88] Karl Abrahamson. On achieving consensus using a shared memory. In *Proceedings of the 7th Annual ACM Symposium on Principles of Distributed Computing*, (PODC), pages 291–302, 1988.

[AC11] Hagit Attiya and Armando Castañeda. A non-topological proof for the impossibility of k-set agreement. In *Proceedings of the 13th Symposium on Self-Stabilizing Systems*, (SSS), pages 108–119, 2011.

[AP12] Hagit Attiya and Ami Paz. Counting-based impossibility proofs for renaming and set agreement. In *Proceedings of the 26th International Symposium on Distributed Computing*, (DISC), pages 356–370. Springer, 2012.

[BG93] Elizabeth Borowsky and Eli Gafni. Generalized flp impossibility result for t-resilient asynchronous computations. In *Proceedings of the 25th ACM Symposium on Theory of Computing*, (STOC), pages 91–100, 1993.

[BL93] James E Burns and Nancy A Lynch. Bounds on shared memory for mutual exclusion. *Information and Computation*, 107(2):171–184, 1993.

[Cha93] Soma Chaudhuri. More choices allow more faults: Set consensus problems in totally asynchronous systems. *Information and Computation*, 105(1):132–158, 1993.

[CIL87] Benny Chor, Amos Israeli, and Ming Li. On processor coordination using asynchronous hardware. In *Proceedings of the 6th Annual ACM Symposium on Principles of Distributed Computing*, (PODC), pages 86–97, 1987.

[CR10] Armando Castaneda and Sergio Rajsbaum. New combinatorial topology bounds for renaming: the lower bound. *Distributed Computing*, 22(5-6):287–301, 2010.

[EGZ18] Faith Ellen, Rati Gelashvili, and Leqi Zhu. Revisionist simulations: A new approach to proving space lower bounds. In *Proceedings of the 37th ACM Symposium on Principles of Distributed Computing*, (PODC), pages 61–70, 2018.

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

[Her91] Maurice Herlihy. Wait-free synchronization. *ACM Transactions on Programming Languages and Systems (TOPLAS)*, 13(1):124–149, 1991.

[HKR13] Maurice Herlihy, Dmitry N. Kozlov, and Sergio Rajsbaum. *Distributed Computing Through Combinatorial Topology*. Morgan Kaufmann, 2013.

[HLM03] Maurice Herlihy, Victor Luchangco, and Mark Moir. Obstruction-free synchronization: Double-ended queues as an example. In *23rd International Conference on Distributed Computing Systems, 2003. Proceedings.*, pages 522–529. IEEE, 2003.

[HS99] Maurice Herlihy and Nir Shavit. The topological structure of asynchronous computability. *J. ACM*, 46(6):858–923, 1999.

[HS06] Gunnar Hoest and Nir Shavit. Toward a topological characterization of asynchronous complexity. *SIAM J. Comput.*, 36(2):457–497, 2006.

[LAA87] M. C. Loui and H. H. Abu-Amara. Memory requirements for agreement among unreliable asynchronous processes. In F. P. Preparata, editor, *Advances in Computing Research*, volume 4, pages 163–183. JAI Press, 1987.

[SZ00] Michael Saks and Fotios Zaharoglou. Wait-free k-set agreement is impossible: The topology of public knowledge. *SIAM J. Comput.*, 29(5):1449–1483, 2000.