Brief Announcement: Colorless Tasks and Extension-Based Proofs

Yusong Shi
Department of Computer Science and Technology, Tsinghua University, Beijing, China

Weidong Liu
Department of Computer Science and Technology, Tsinghua University, Beijing, China
Zhongguancun Laboratory, Beijing, China

Abstract

The concept of extension-based proofs models the idea of a valency argument, which is widely used in distributed computing. Extension-based proofs are limited in power: it has been shown that there is no extension-based proof of the impossibility of a wait-free protocol for (n, k)-set agreement among $n > k \geq 2$ processes. There are only a few tasks that have been proven to have no extension-based proof of the impossibility, since the techniques in these works are closely related to the specific task.

We give a necessary and sufficient condition for colorless tasks to have no extension-based proofs of the impossibility of wait-free protocols in the NIIS model. We introduce a general adversarial strategy decoupled from any concrete task specification. In this strategy, some properties of the chromatic subdivision that is widely used in distributed computing are proved.

2012 ACM Subject Classification Theory of computation → Interactive proof systems; Theory of computation → Distributed algorithms; Theory of computation → Distributed computing models; Theory of computation → Problems, reductions and completeness

Keywords and phrases Colorless tasks, Impossibility proofs, Extension-based proof

Digital Object Identifier 10.4230/LIPIcs.DISC.2024.54

Related Version *Extended Version*: `https://arxiv.org/abs/2303.14769` [12]

Acknowledgements We would like to thank Faith Ellen and Shihao Liu for helpful discussions and the anonymous reviewers for their comments.

1 Introduction

One of the most important results in distributed computing, due to Fischer, Lynch, and Paterson [7], is that there is no deterministic protocol that solves the consensus task in the asynchronous message passing system. The key idea of their proof is called a valency argument, which proves the existence of an infinite execution in which no process terminates.

The (n, k)-set agreement task, which is a generalization of the consensus task, was first proposed by Chaudhuri [6]. The (n, k)-set agreement task was independently shown to have no wait-free protocol by Borowsky and Gafni [4], Herlihy and Shavit [8], and Saks and Zaharoglou [11]. Topological techniques were used to prove these results.

In [1], Alistarh, Aspnes, Ellen, Gelashvili and Zhu pointed out the differences between valency arguments and combinatorial or topological techniques. In the proof by Fischer, Lynch and Paterson, an infinite execution can be constructed by extending an initial execution infinitely often. In contrast, in those proofs using combinatorial techniques, the existence of a bad execution is proved, but not explicitly constructed. [1] generalized this type of proof and called it an extension-based proof. An extension-based proof is defined as an interaction between a prover and a protocol that claims to solve a task. The prover tries to find out some errors in the protocol by submitting queries to the protocol. If the prover manages to do so, then the prover wins against the protocol. If there exists a prover that can win against any protocol that claims to solve a task, we say that this task has an extension-based impossibility proof. The proof of the impossibility of consensus is an example of an extension-based proof. In the same paper, they showed that there are no extension-based proofs for the impossibility of a wait-free protocol for the (n, k)-set agreement in the non-uniform iterated immediate snapshot (NIIS) model. The same result was proved in the non-uniform iterated snapshot (NIS) model in the journal version [2]. Some tasks [3, 10] that are closely related to the set agreement task and 1-dimensional colorless tasks have also been shown to have no extension-based proofs.

Do other tasks also have no extension-based impossibility proofs? One way to generate new results is to find a condition that characterizes the tasks that have extension-based impossibility proofs. A task is specified by a tuple $(\mathcal{I}, \mathcal{O}, \Delta)$. A protocol solves a task $(\mathcal{I}, \mathcal{O}, \Delta)$ if, starting with any input values in $\mathcal{I}$, processes decide on output values in $\mathcal{O}$ after communicating with each other for some steps according to the protocol, respecting the input/output relation Δ. Both $\mathcal{I}$ and $\mathcal{O}$ are closed under containment, since processes are assumed to be faulty and may crash at any time. We can show that a task $(\mathcal{I}, \mathcal{O}, \Delta)$ has no extension-based proofs if we can design an adversarial strategy that can construct an adaptive protocol that wins against any extension-based prover.

In this paper, we focus on a subset of tasks called *colorless tasks.* A colorless task is defined only in terms of input and output values, without process ids. *All* our discussions use the definition and related consequences of tasks rather than those specified for colorless tasks. So why do we talk about colorless tasks while adopting the form of general tasks? Part of our design needs a property (Property 1) of the input/output relation Δ.

▶ **Property 1.** *In any possible execution, if a process is allowed to output a value v, then any other process that has seen a superset of the values seen by this process is also allowed to output the value v.*

This property is intrinsic for colorless tasks.

2 Model

An *immediate snapshot* (IS) object, introduced by Borowsky and Gafni in [5] consists of an array and supports only one type of operation, called a *writeread* operation, where a process with id i writes a value to the i-th cell of the array and returns a snapshot of the array immediately following the write. The writeread operations performed to some IS object by different processes are said to be concurrent if all snapshots occur after all writes to the array are finished.

The NIIS model assumes an unbounded sequence of IS objects $IS_1, IS_2 \cdots$. $(n + 1)$ sequential threads of control, called *processes*, $\Pi = \{p_0, p_1 \dots p_n\}$, communicate through IS objects to solve decision tasks. A *protocol* is a distributed program to solve a task. In any execution of a protocol in the NIIS model, each process p_i performs a writeread operation on each IS object starting from IS_1. Initially, p_i's state contains its identifier i and its input value. Each time p_i performs a writeread operation on some IS object IS_j using its current state s_i as argument, and sets its current state s_i to its identifier i and the response of its writeread operation. Then p_i consults a map δ to determine whether it should terminate and output a value. If $\delta(s_i) \neq \perp$, p_i outputs $\delta(s_i)$ and terminates. Otherwise, it continues to access this next IS object. Therefore, each *NIIS protocol* is determined by a decision map δ from a local state to output values or $\perp$.

A *configuration C* consists of the contents of each shared object and the state of each process. However, since each process remembers its entire history and only process p_i can write to the i-th component of each IS object, a configuration is fully determined by the states of processes in this configuration. An initial configuration consists of the input values and process ids of all processes. A process is *active* in a configuration if it has not terminated. A configuration is terminated if all processes have terminated.

A *scheduler* repeatedly chooses a set of processes that are poised to perform writeread operations on the same IS object concurrently. A *schedule* α is an ordered sequence of sets of processes chosen by the scheduler. Let C be a reachable configuration in which all active processes have accessed the same number of IS objects. For any set P of processes, a *P-only 1-round* schedule from C is an ordered partition of processes in P that are active in C. A *P-only r-round* schedule from C is a schedule $\alpha_1\alpha_2\cdots\alpha_r$ such that each α_i is a P-only 1-round schedule from $C\alpha_1\cdots\alpha_{i-1}$. A *full r-round schedule* from C is a P-only r-round schedule from C where $P=\Pi$.

An *(abstract) simplex* is the set of all subsets of some finite set. There is a natural geometric interpretation of an (abstract) simplex. In this paper, we use the two definitions interchangeably. An n-simplex S spanned by a set of affinely independent vertices $\{\vec{v}_0,\dots\vec{v}_n\}$ is defined to be the set of all points x such that $x=\sum_{i=0}^{n} t_i\vec{v}_i$ where $\sum_{i=0}^{n} t_i=1$ and $t_i\geq 0$ for all i. Any simplex T spanned by a subset of $\{\vec{v}_0,\dots\vec{v}_n\}$ is called a *face* of S. An *(abstract) simplicial complex* is a finite collection $\mathcal{K}$ of sets that is closed under subset: for any set $S\in\mathcal{K}$, if $S^{'}\subseteq S$, then $S^{'}\in\mathcal{K}$.

For a task, all possible input or output values can be represented by a simplicial complex, called an *input complex* $\mathcal{I}$ or an *output complex* $\mathcal{O}$. Each vertex $\vec{s}$ of these simplices is labeled with a process id and a value that are denoted by $ids(\vec{v})$ and $vals(\vec{s})$, respectively. The *topological task specification* is defined as a carrier map that carries each simplex S of the input complex to a subcomplex of the output complex.

Like tasks, protocols can be represented in terms of combinatorial topology. The *i-th protocol complex* consists of all simplices, represents configurations that are reachable from some initial configuration by a i-round schedule. The *i-th execution map* is a carrier map that carries each initial configuration to all configurations reached from it in the i-th protocol complex. A *protocol* is represented by $(\mathcal{I},\mathcal{P},\Xi)$ and a simplicial map $\delta:\mathcal{P}\to\mathcal{O}$ where $\mathcal{I}$ is the input complex, $\mathcal{P}$ is the i-th protocol complex, and Ξ is the i-th execution map, for some non-negative integer i. We say that a protocol $(\mathcal{I},\mathcal{P},\Xi)$ *solves a task* $(\mathcal{I},\mathcal{O},\Delta)$ if $\delta(\Xi(s^k))$ is in $\Delta(s^k)$ for each $s^k\in\mathcal{I}$.

Hoest and Shavit [9] showed that the i-th protocol complex of an NIIS protocol is equal to $\chi^i(\mathcal{I},\delta)$, where χ is the non-uniform chromatic subdivision constructed from the NIIS protocol. Let U be any simplex in $\mathcal{I}$. A *partial protocol* δ_U with respect to U specifies whether a process should output a value(and which output if so) in each configuration reached from an initial configuration that contains U, by a schedule in which $ids(U)$ is the first set of processes. The i-th *protocol complex of a partial protocol δ_U with respect to U* is defined as follows.

- $\mathbb{F}_0(U)$ is the set of all simplices in $\mathcal{I}$ that contain U.
- $\mathbb{F}_1(U)$ is the subcomplex of $\chi(\mathbb{F}_0(U),\delta_U)$ consisting of all simplices representing configurations reachable via 1-round schedules in which the processes in $ids(U)$ have the input values $vals(U)$ and $ids(U)$ is the first set of processes to take a step.
- For $i\geq 1$, $\mathbb{F}_{i+1}(U)=\chi(\mathbb{F}_i(U),\delta_U)$ consists of all simplices representing configurations reachable via $(i+1)$-round schedules in which the processes in $ids(U)$ have the input values $vals(U)$ and $ids(U)$ is the first set of processes to take a step.

Similarly, a partial protocol with respect to U can be represented topologically as $(\mathbb{F}_0(U),\mathbb{F}_i(U),\Xi)$. The i-th *execution map* Ξ is a carrier map that carries each initial configuration in $\mathbb{F}_0(U)$ to all configurations reached from it in $\mathbb{F}_i(U)$. We say that a partial protocol δ_U with respect to U satisfies the task specification Δ if $\delta_U(\Xi(s^k))$ is in $\Delta(s^k)$ for each $s^k\in\mathcal{I}$ where $\Xi(s^k)$ is not empty.

3 Motivation and summary

In this section, we give a description of the motivation behind our necessary and sufficient condition for a colorless task defined by $(\mathcal{I}, \mathcal{O}, \Delta)$ to have no restricted extension-based proofs.

The (n, k)-set agreement task is the first task that was shown to have no extension-based impossibility proofs. As shown in [1], given any extension-based prover, the adversary will pretend to have a protocol for the (n, k)-set agreement task during phase 1 of the interaction. But after the prover chooses a schedule at the end of phase 1, the adversary can assign a valid output value to each undefined configuration that the prover can reach in the later phases. In other words, the adversary has a partial protocol compatible with the existing assigned values that satisfies the task specification of the (n, k)-set agreement after phase 1. We divide the adversarial strategy into two parts: In this first part, the adversary adaptively defines a protocol in response to any specific prover's queries during the first r phases. In the second part, the adversary designs a partial protocol after the end of phase r so that the prover is doomed to lose.

If the adversary can prevent the prover from finding a problem in the first r phases and construct a partial protocol after phase r, no matter what queries the prover makes during the first r phases and which configurations the prover has chosen at the end of the first r phases, we say that the adversary can *finalize after phase* r. We can show that the adversary can win against any extension-based prover, if and only if the adversary can finalize after phase r for some positive integer r.

In this paper, we introduce the idea behind our necessary and sufficient condition for finalization after phase 1. Most of the techniques used in the proof for larger values of r are introduced in the proof of this case.

We start with a necessary condition assuming that no queries are submitted by the prover during phase 1: there must exist a partial protocol with respect to any possible simplex $U \in \mathcal{I}$. We use the asynchronous computability theorem to give a topological condition for a task to have a partial protocol with respect to each $U \in \mathcal{I}$.

Then we allow the prover to submit queries in phase 1. In the protocol complex of a partial protocol, the output values of some configurations may already be determined during the interaction of phase 1. For two simplices U_1 and U_2 in $\mathcal{I}$ and each simplex S in $\mathbb{F}_1(U_1) \cap \mathbb{F}_1(U_2)$, we consider the configuration, denoted by $CEN(S)$, reached from S via a schedule that repeats the set of processes $ids(S)$ until all processes in $ids(S)$ terminate. We say that two partial protocols δ_{U_1} and δ_{U_2} are *compatible* if the output values of $CEN(S)$ are the same under δ_{U_1} and δ_{U_2} for each possible S. A set of partial protocols is *compatible* if any two partial protocols are compatible. We show that an enhanced necessary condition for finalization after phase 1 is that the set of partial protocols is compatible.

Then we prove that a task $(\mathcal{I}, \mathcal{O}, \Delta)$ has a set of compatible partial protocols $\{\delta_U | U \in \mathcal{I}\}$ then the adversary can always finalize after phase 1, by showing how the adversary can construct an adaptive protocol to win against any prover using this set of compatible partial protocols which can be assumed to terminate after r_m rounds.

The adversary uses an infinite sequence of complexes $S^0, S^1...$ and an integer t (current complex) to represent the adaptive protocol, in which $S^0 = \mathcal{I}$. Our adversary maintains three invariants in the interaction with an extension-based prover. For each $0 \le r < t$ and each vertex $v \in S^r$, $\delta(v)$ is defined. If v is a vertex in S^t, then $\delta(v)$ is undefined or $\delta(v) \neq \perp$. If a vertex s represents the state of a process in a configuration that the prover has reached, then $\delta(v)$ is defined. The second invariant is about the safety of the adaptive protocol. The output values defined by the adaptive protocol will not violate the task specification. To achieve this, the adversary defines the δ values using the output values obtained from the set of partial protocols. The third invariant is that the active distance between a configuration terminated with output values given by δ_{U_1} and a configuration terminated with output values given by δ_{U_2}, where $U_2 \neq U_1$ is at least 3.

The adversary sets $\delta(v) = \perp$ for each vertex in S^r where $r \leq r_m$. The only question is to decide δ for a vertex in S^r where $r > r_m$. Each terminated vertex has a simplex U of $\mathcal{I}$ as its label, indicating which partial protocol its δ value is from. If v is reached from some n-simplex s^n in $\mathbb{F}_{r_m}(U)$ and has a label U, the adversary can use the value of $\delta_U(v^{'})$ where $s^n \in \mathbb{F}_{r_m}(U)$ and $v^{'}$ is the vertex of s^n with the same process id as v. A problem here is that sometimes the adversary has to terminate v with a different label $U^{'}$ to avoid an infinite execution.

If an n-simplex s^n in $Ch^{r_m}(\mathcal{I})$ is not in $\mathbb{F}_{r_m}(U^{'})$, but shares a simplex s_s with $\mathbb{F}_{r_m}(U^{'})$, we define an n-simplex in $\mathbb{F}_{r_m}(U^{'})$ as the canonical neighbor of s^n with the label $U^{'}$. If v has the label $U^{'} \neq U$, the adversary can use the value of $\delta_{U'}(N(s^n, U^{'}))$ where $s^n \in \mathbb{F}_{r_m}(U)$ and $v^{'}$ is the vertex of s^n with the same process id as v. An implementation of canonical neighbors is provided such that this assignment of output values does not violate the carrier map.

We show that using our adversarial strategy, the prover cannot win in phase 1, which means that the prover has chosen some configuration to end phase 1. Let U be the simplex in $\mathcal{I}$ representing the first set of processes in the schedule from some initial configuration to the chosen configuration and their input values. In the subdivision of each n-simplex $s^n \in \mathbb{F}_{r_m}(U)$, the δ values of terminated vertices are obtained from $\delta_U(s^n)$ or $\delta_{U'}(N(s^n, U^{'}))$. Configurations with different labels are separated according to invariant (3). Although δ_U or $\delta_{U'}$ are two different partial protocols, they have the same output values for some configuration $CEN(S)$ since they are compatible by assumption for some shared simplex $S \in \mathbb{F}_1(U) \cap \mathbb{F}_1(U^{'})$. There is a sequence of output assignments from $\delta_{U'}(\tau)$ to $\delta_U(CEN(S))$ and then to $\delta_U(s^n)$ for some shared simplex S such that only one process changes its output values in two adjacent output assignments, where τ is a $dim(CEN(S))$-dimensional subsimplex of $N(s^n, U^{'})$. The colorless condition is used here since the dimension of $CEN(S)$ is less than $n = dim(s^n)$. The adversary terminates the vertices adjacent to the vertices terminated with the label $U^{'}$ using the output assignments of this sequence until the output values of the outermost layer are in $\delta_U(s^n)$. Finally, the adversary can define the δ value of each remaining undefined vertex using $\delta_U(s^n)$. A partial protocol with respect to U is constructed.

▶ **Theorem 1.** *For a colorless task $(\mathcal{I}, \mathcal{O}, \Delta)$, there exists an adversary that can finalize after the first round to win against any restricted extension-based prover if and only if there exists a compatible set of partial protocols, each of which corresponds to a simplex $U \in \mathcal{I}$.*

References

1 Dan Alistarh, James Aspnes, Faith Ellen, Rati Gelashvili, and Leqi Zhu. Why extension-based proofs fail. *Proceedings of the 51'st Annual ACM Symposium on Theory of Computing (STOC)*, pages 986–996, 2019. doi:10.1145/3313276.3316407.

2 Dan Alistarh, James Aspnes, Faith Ellen, Rati Gelashvili, and Leqi Zhu. Why extension-based proofs fail. *SIAM Journal on Computing*, 52(4):913–944, 2023. doi:10.1137/20M1375851.

3 Dan Alistarh, Faith Ellen, and Joel Rybicki. Wait-free approximate agreement on graphs. In *Structural Information and Communication Complexity: 28th International Colloquium, SIROCCO 2021, Wrocław, Poland, June 28 – July 1, 2021, Proceedings*, pages 87–105, Berlin, Heidelberg, 2021. Springer-Verlag. doi:10.1007/978-3-030-79527-6_6.

4 Elizabeth Borowsky and Eli Gafni. Generalized flp impossibility result for t-resilient asynchronous computations. In *Proceedings of the Twenty-Fifth Annual ACM Symposium on Theory of Computing*, STOC '93, pages 91–100, New York, NY, USA, 1993. Association for Computing Machinery. doi:10.1145/167088.167119.

5 Elizabeth Borowsky and Eli Gafni. Immediate atomic snapshots and fast renaming. In *Proceedings of the Twelfth Annual ACM Symposium on Principles of Distributed Computing*, PODC '93, pages 41–51, New York, NY, USA, 1993. Association for Computing Machinery. doi:10.1145/164051.164056.

6 Soma Chaudhuri. More choices allow more faults: Set consensus problems in totally asynchronous systems. *Inf. Comput.*, 105(1):132–158, July 1993. doi:10.1006/inco.1993.1043.

7 Michael J. Fischer, Nancy A. Lynch, and Michael S. Paterson. Impossibility of distributed consensus with one faulty process. *J. ACM*, 32(2):374–382, April 1985. doi:10.1145/3149.214121.

8 Maurice Herlihy and Nir Shavit. The topological structure of asynchronous computability. *J. ACM*, 46(6):858–923, November 1999. doi:10.1145/331524.331529.

9 Gunnar Hoest and Nir Shavit. Toward a topological characterization of asynchronous complexity. *SIAM Journal on Computing*, 36(2):457–497, 2006. doi:10.1137/S0097539701397412.

10 Shihao Liu. The Impossibility of Approximate Agreement on a Larger Class of Graphs. In Eshcar Hillel, Roberto Palmieri, and Etienne Rivière, editors, *26th International Conference on Principles of Distributed Systems (OPODIS 2022)*, volume 253 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 22:1–22:20, Dagstuhl, Germany, 2023. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.OPODIS.2022.22.

11 Michael Saks and Fotios Zaharoglou. Wait-free k-set agreement is impossible: The topology of public knowledge. *SIAM J. Comput.*, 29(5):1449–1483, March 2000. doi:10.1137/S0097539796307698.

12 Yusong Shi and Weidong Liu. Colorless tasks and extension-based proofs, 2023. arXiv:2303.14769.