One Step Forward, One Step Back: FLP-Style Proofs and the Round-Reduction Technique for Colorless Tasks

Hagit Attiya
Department of Computer Science, Technion, Israel

Pierre Fraigniaud
IRIF – CNRS & Université Paris Cité, France

Ami Paz
LISN – CNRS & Université Paris-Saclay, France

Sergio Rajsbaum
IRIF, École Polytechnique and Instituto de Matemáticas, UNAM, Mexico

Abstract

The paper compares two generic techniques for deriving lower bounds and impossibility results in distributed computing. First, we prove a *speedup theorem* (a-la Brandt, 2019), for wait-free *colorless* algorithms, aiming at capturing the essence of the seminal *round-reduction* proof establishing a lower bound on the number of rounds for 3-coloring a cycle (Linial, 1992), and going by *backward induction*. Second, we consider *FLP-style* proofs, aiming at capturing the essence of the seminal consensus impossibility proof (Fischer, Lynch, and Paterson, 1985) and using *forward induction*.

We show that despite their very different natures, these two forms of proof are tightly connected. In particular, we show that for every colorless task Π, if there is a round-reduction proof establishing the impossibility of solving Π using wait-free colorless algorithms, then there is an FLP-style proof establishing the same impossibility. For 1-dimensional colorless tasks (for an arbitrarily number $n \geq 2$ of processes), we prove that the two proof techniques have exactly the *same* power, and more importantly, both are *complete*: if a 1-dimensional colorless task is *not* wait-free solvable by $n \geq 2$ processes, then the impossibility can be proved by both proof techniques. Moreover, a round-reduction proof can be *automatically* derived, and an FLP-style proof can be automatically generated from it.

Finally, we illustrate the use of these two techniques by establishing the impossibility of solving any colorless *covering* task of arbitrary dimension by wait-free algorithms.

2012 ACM Subject Classification Theory of computation → Distributed algorithms

Keywords and phrases Wait-free computing, lower bounds

Digital Object Identifier 10.4230/LIPIcs.DISC.2023.4

Related Version *Full Version*: http://arxiv.org/abs/2308.04213

Funding *Hagit Attiya*: partially supported by the Israel Science Foundation (grants 380/18 and 22/1425).
Pierre Fraigniaud: partially supported by the ANR projects DUCAT (ANR-20-CE48-0006), FREDDA (ANR-17-CE40-0013), and QuData (ANR-18-CE47-0010).
Sergio Rajsbaum: partially supported by the ANR projects DUCAT (ANR-20-CE48-0006).

Acknowledgements The authors thank Faith Ellen and the referees for helpful comments.

1 Introduction

We analyze the relative power of two generic and versatile techniques for establishing lower bounds and impossibility results in asynchronous distributed computing. We focus on solving *tasks*, defined as triples $\Pi = (\mathcal{I}, \mathcal{O}, \Delta)$, where processes start with initial input values defined by $\mathcal{I}$, and decide irrevocably on output values allowed by $\mathcal{O}$ after communicating with each other for some number of steps, respecting the input/output relation Δ; the sets of processes, input values and output values are all finite.

This paper concentrates on the family of *colorless* tasks, including consensus, set agreement [17], loop agreement [25], and various robot and graph agreement tasks [3, 16]. A colorless task is defined only in terms of input and output values, regardless of the number of processes involved, and regardless of which process has a particular input or output value; accordingly, $\mathcal{I}$ and $\mathcal{O}$ consist of sets of values, without process ids. For instance, in the binary consensus task, $\mathcal{I} = \big\{\{0\}, \{1\}, \{0,1\}\big\}$, meaning that all processes may start with input 0, or all processes may start with input 1, or some processes may start with input 0 while others may start with input 1. In the same task, $\mathcal{O} = \big\{\{0\}, \{1\}\big\}$, meaning that the only valid output configurations are when all processes output 0, or all processes output 1. Finally, consensus specification is captured by $\Delta(\{0\}) = \{0\}$, $\Delta(\{1\}) = \{1\}$, and $\Delta(\{0,1\}) = \big\{\{0\}, \{1\}\big\}$, meaning that, if there was an initial agreement between the input values then the processes must stick to this agreement and output their input values, and otherwise they are allow to output either 0 or 1, as long as they all agree on the same value.

Colorless tasks are simpler to analyze than general tasks, such as symmetry breaking tasks [15], but they are still undecidable [24]. They have an elegant computability characterization [23, 28], essentially stating that a colorless task is wait-free solvable if and only if there is a continuous map from the geometric realization of $\mathcal{I}$ to that of $\mathcal{O}$ that respects Δ. One major interest of colorless tasks is that, whenever solvable, they can be solved by simple algorithms, referred to as *colorless algorithms* [28] (see also [23, Ch. 4]). Roughly speaking, such algorithms ignore multiplicities of input values and process states, and manipulate only *sets* of values; in contrast, general algorithms manipulate *vectors* of values and take into account which processes possess which value.

The two lower-bound or impossibility techniques at the core of our work are *FLP-style proofs*, named after Fischer, Lynch and Paterson [19], and *round-reduction proofs*, whose first occurrence might be attributed to Linial [33]. The former offers a form of *forward* induction technique, while the latter offers a form of *backward* induction, and they both present some form of *locality*. We recall these two techniques hereafter.

1.1 Stepping Forward: FLP-Style Impossibility Proofs

The celebrated FLP proof technique [19] is perhaps the most used technique for proving impossibility results in distributed computing. It has been used to prove the impossibility of solving consensus and several other problems [6, 18, 34], as well as to derive lower bounds [1, 6, 31]. To prove that a task $\Pi = (\mathcal{I}, \mathcal{O}, \Delta)$ is not wait-free solvable, the FLP technique considers a hypothetical algorithm ALG solving the task, and constructs an infinite sequence $\sigma_0, \sigma_1, \ldots$ of system configurations such that, for every $i \geq 0$,

- σ_{i+1} is a successor of σ_i, and
- ALG cannot output in σ_i.

To initiate this sequence, the prover is allowed to ask the *valencies* of all initial configurations $\sigma \in \mathcal{I}$, where the valency of a configuration σ is the set of values that are output by ALG in all executions starting from σ. Based on these valencies, the prover selects an initial

configuration $\sigma_0 \in \mathcal{I}$. Then, given a configuration σ_i, the prover asks the algorithm for the valencies of some successor configurations of σ_i, and one of these configurations is selected to be the next configuration σ_{i+1} in the sequence, and so on. This is a form of *forward induction*, starting with σ_0 and constructing a sequence of configurations one after the other.

If ALG actually solves the task Π, then the prover will fail to construct an infinite sequence, merely because ALG can reveal the actual valencies that it produces for each given configuration. On the other hand, if for every algorithm ALG hypothetically solving Π the prover successfully constructs an infinite sequence, then this establishes that Π is not solvable because the sequence is constructed in such a way that ALG cannot output in σ_i. For instance, the FLP impossibility proof for consensus [19] constructs such an infinite sequence $\sigma_0, \sigma_1, \ldots$ for every algorithm ALG by establishing that (1) there exists an initial *bivalent* configuration σ_0 (i.e., a configuration from which an execution of ALG leads all processes to output 0, and another execution of ALG leads all processes to output 1), and (2) for every bivalent configuration σ_i, there exists a bivalent one-step successor σ_{i+1} of σ_i that is bivalent.

An FLP-style impossibility proof is a simple case of extension-based impossibility proofs [2] and *local proofs* [5], hence if such a proof exists for a task Π then there are also extension-based and local impossibility proofs for it. All these techniques use types of *valency-arguments* [6, Chapter 7], in the sense that they consider output values in executions starting from a given configuration, and then decide on the next configuration. All these techniques work for consensus but fail for set agreement (i.e., set agreement is not solvable, but this impossibility cannot be established by valency arguments), and the exact set of tasks for which each of them applies is not known.

1.2 Stepping Back: Round-Reduction Impossibility Proofs

The round-reduction proof technique in distributed computing can be traced back to the seminal work of Linial [33] establishing a lower bound for coloring the n-node cycle C_n using a (failure-free) synchronous algorithm. In a nutshell, he proved that for every $t \geq 1$, if there exists a t-round algorithm ALG producing a proper k-coloring of C_n, then there exists a $(t-1)$-round algorithm ALG$'$ producing a proper 2^{2^k}-coloring of C_n. Repeating this argument for roughly $\frac{1}{2}\log^* n$ times implies that if there is a $(\frac{1}{2}\log^* n)$-round algorithm for 3-coloring C_n then there exists a 0-round algorithm for $(n-1)$-coloring it, which is impossible. Here, $\log^\star n$ denotes the number of times one should apply the $\log_2$ function to get from n to a value smaller than 1.

This technique was generalized as a *speedup theorem* by Brandt [13]. Such a theorem is based on establishing the existence of a map F transforming any task Π in some class $\mathcal{T}$ of tasks into another task $\Pi' \in \mathcal{T}$ such that, for every $t \geq 1$, if Π is solvable in t rounds by an algorithm ALG from some class $\mathcal{A}$ of algorithms, then $F(\Pi)$ is solvable in $t-1$ rounds by an algorithm ALG$' \in \mathcal{A}$. Whenever such a theorem can be established, we get that for every task $\Pi \in \mathcal{T}$, and for every $t \geq 0$, if the task $F^{(t)}(\Pi)$ obtained by iterating t times F on Π is not solvable in zero rounds by an algorithm in $\mathcal{A}$, then Π is not solvable in t rounds by an algorithm in $\mathcal{A}$, which provides a lower bound on the complexity of Π. In particular, if $F^{(t)}(\Pi)$ is not solvable in zero rounds by an algorithm in $\mathcal{A}$ for all $t \geq 0$, then Π cannot be solved by an algorithm in $\mathcal{A}$. We refer to this technique as a *round-reduction* impossibility or and lower bound proof. In contrast with the forward induction approach of FLP-style proofs, round-reduction is an a-posteriori technique, starting by assuming an algorithm solves a problem in t rounds, claiming another problem is solvable in $t-1$ rounds, and repeating this argument down to 0 rounds; we hence refer to it as backward induction.

A speedup theorem has been established in [13] for solving locally-checkable labeling (LCL) tasks [36] using algorithms running in the anonymous LOCAL model [37]. A speedup theorem has also been established in [20], but for general (colored) tasks and wait-free algorithms running in the *iterated immediate snapshot* (IIS) model [7]. The transformations F_{LOCAL} and F_{IIS} used in [13] and [20] respectively, are of very different natures. Nevertheless, both enabled to establish lower bounds for various tasks, including sink-less orientation and maximal independent set (MIS) in the LOCAL model (see [8,13]), and approximate agreement in the IIS model, and even when the IIS model is enhanced with powerful objects like test&set (see [20]).

Our first contribution is a speedup theorem for wait-free colorless algorithms solving colorless tasks in the IIS model. It is important to note that, although the set of colorless tasks is a subset of the set of general tasks, the speedup theorem in [20] does not imply our speedup theorem, since the transformation F_{IIS} used in [20] does not apply to colorless *algorithms*. Specifically, if Π is solvable in t rounds by n processes running a wait-free colorless algorithm, then $F_{\mathrm{IIS}}(\Pi)$ is indeed solvable in $t-1$ rounds by n processes, but running a wait-free algorithm that *may not be colorless*. As a consequence, the theorem cannot be iterated, which ruins the ability to design a round-reduction proof for colorless tasks. The speedup theorem for colorless tasks we present here uses a transformation that preserves solvability by colorless algorithms. The transformation CI we define (Definition 4) has the following properties, as shown in Theorem 6 and Theorem 11. Applications of this theorem to approximate agreement and covering tasks can be found in Section 7.

▶ **Theorem A.** *For every $n \geq 2$ and every $t \geq 1$, the transformation* CI *maps any colorless task Π to a colorless task* CI(Π) *such that, when considering n processes running a colorless wait-free algorithm in the* IIS *model:*

- *If Π is 1-dimensional, then Π is solvable in t rounds if and only if* CI(Π) *is solvable in $t-1$ rounds;*
- *Regardless of the dimension of Π, if Π is solvable in t rounds then* CI(Π) *is solvable in $t-1$ rounds.*

A round-reduction impossibility or lower bound proof derived from Theorem A essentially proceeds by computing $\mathsf{CI}^{(t)}(\Pi)$, and checking whether $\mathsf{CI}^{(t)}(\Pi)$ is solvable in zero rounds. If the answer is negative for some $t \geq 1$, the proof successfully shows that Π cannot be solved in t rounds, and if it is negative for all $t \geq 1$, the proof shows Π is not solvable; otherwise, the proof fails. While CI is not the only possible round-reduction operator for colorless tasks, we focus solely on it in this work.

In the full version of this paper, we illustrate the fact that round-reduction does not extend in a straightforward manner to all classes of algorithms. To this end, we consider *comparison-based* algorithms, an important class of algorithms used for studying tasks such as renaming and weak symmetry-breaking. We show that a closure operator similar to the ones considered here and in [20] but restricted to comparison-based algorithms, does not suffice for deriving a speedup theorem for comparison-based algorithms.

1.3 Round-Reduction vs. FLP-Style Impossibility Proofs

Interestingly, as for extension-based proofs, the round-reduction proofs derived from the speedup theorem in [20] work for consensus but fail for set-agreement and the same holds for our transformation CI. We next show why the fact that both transformations succeed for binary consensus but fail for set-agreement should not come as a surprise, in light of prior results about FLP-style proofs.

Our second contribution shows that, although round-reduction proofs and FLP-style proofs may appear very different in nature, their power in term of establishing impossibility results can be compared. We actually show that the FLP-style proof technique is at least as strong as the round-reduction proof technique.

▶ **Theorem B** (Theorem 13). *For every colorless task* Π *and* $n \geq 2$*, if there is a round-reduction proof establishing the impossibility of solving* Π *by* n *processes running a wait-free colorless algorithm, then there is an FLP-style proof establishing the same impossibility.*

So, in particular, the fact that there is no round-reduction proof for the impossibility of solving set-agreement wait-free should not come as a surprise, since it is known that set-agreement has no extension-based impossibility proof [2, 5, 14], which is a proof technique at least as strong as FLP-style proofs. Yet, the situation is a bit subtle here: the results in [2, 5] rule out the existence of an FLP-style impossibility proof for solving set-agreement using general algorithms, and [14] proves a similar result for anonymous algorithms (where processes see a multi-set of the values stored in the memory and not a vector), but these works do not necessarily rule out the existence of an FLP-style impossibility proof for solving set agreement using colorless algorithms. The restriction to colorless algorithm allows the claimed algorithm ALG less flexibility regarding the valencies it announces to the prover, so an FLP-style impossibility proof when restricting ALG to be colorless is not ruled out by the previous results.

Nevertheless, we were able to show that, for *1-dimensional* colorless tasks, round-reduction proofs and FLP-style proofs have exactly the same power (Corollary 14). Recall that, in 1-dimensional colorless tasks, $\mathcal{I}$ and $\mathcal{O}$ are graphs, i.e., the n processes can start with at most two different input values and output at most two different values. This family includes binary consensus, approximate agreement with binary inputs, and the colorless version of wait-free checkable tasks [21], but not set-agreement. 1-dimensional colorless tasks are well studied, and they are known to be wait-free solvable if and only if they are 1-resilient solvable, both in the read/write model (with at least two processes) and in the message-passing model (with at least three processes) [9, 12, 27]. In fact, we not only show equivalence between round-reduction and FLP-style proof techniques for 1-dimensional colorless tasks, but we show that both are complete for these tasks (Corollary 12 for the round-reduction technique and Corollary 15 for FLP-style proofs). Hence, if a 1-dimensional colorless task is not wait-free solvable by $n \geq 2$ processes, then this impossibility can be proved by both proof techniques.

▶ **Theorem C.** *The round-reduction and FLP-style proof techniques are both complete for 1-dimensional colorless tasks and wait-free colorless algorithms.*

Theorem C follows from the fact that, for 1-dimensional colorless tasks (and colorless algorithms), our speedup theorem (Theorem A) also provides a *necessary* condition, that is, for every 1-dimensional colorless task Π, every $n \geq 2$, and every $t \geq 1$, Π is solvable in t rounds by n processes running a wait-free colorless algorithm in the IIS model *if and only if* $\mathsf{CI}(\Pi)$ is solvable in $t-1$ rounds by n processes running a wait-free colorless algorithm in the IIS model. This if-and-only-if condition provides a mechanical way for deciding whether a given 1-dimensional colorless task Π is solvable. Indeed, the transformation CI used in Theorem A has a desirable property: it preserves the number of input and output values (i.e., the vertices in $\mathcal{I}$ and $\mathcal{O}$), and it may just potentially add some combinations of output values that were not legal in Π but become legal in $\mathsf{CI}(\Pi)$. As a consequence, iterating CI starting from Π necessarily leads to a fixed point Π^* for CI, i.e., $\mathsf{CI}(\Pi^*) = \Pi^*$, after a bounded

number of iterations. It follows that a 1-dimensional colorless task Π is wait-free solvable in the IIS model if and only if Π^* is wait-free solvable in zero rounds, which is decidable. The completeness of the FLP-style proof technique follows. Indeed, if a 1-dimensional colored task Π is not wait-free solvable, then Π has a round-reduction impossibility proof (by computing the fixed point Π^*, and showing that Π^* is not solvable in zero rounds), from which it follows, thanks to Theorem B, that Π has an FLP-style impossibility proof.

1.4 Applications

We illustrate the concepts and results introduced in this paper by applying them to the vast class of *covering tasks*, whose colored version was introduced and studied in [21] under the name *locality-preserving* tasks and further studied in [41]. To get the intuition of such a tasks, it is easier to consider 1-dimensional covering tasks, for which $\mathcal{I}$ and $\mathcal{O}$ are graphs. Recall that for two connected simple (i.e., no self-loops nor multiple edges) graphs G and H, and a function $f : V(H) \to V(G)$, the pair (H, f) is a covering of G if f is an homorphism (i.e., it preserves edges) and, for every $v \in V(H)$, the restriction of f to $N_H[v]$ is a one-to-one mapping $f : N_H[v] \to N_G[f(v)]$. For instance, for $C_3 = (v_0, v_1, v_2)$, $C_6 = (u_0, \ldots, u_5)$, and $f : V(C_6) \to V(C_3)$ defined as $f(u_i) = v_{i \bmod 3}$, (C_6, f) is a covering of C_3. A covering $(\mathcal{O}, f)$ of $\mathcal{I}$ induces a task $\Pi = (\mathcal{I}, \mathcal{O}, \Delta)$ where Δ is essentially defined as f^{-1}. For higher dimensional colorless tasks, $\mathcal{I}$ and $\mathcal{O}$ are connected simplicial complexes, and f must be simplicial, but the general idea is the same [39]. A covering $(\mathcal{O}, f)$ of $\mathcal{I}$ is trivial if $\mathcal{I}$ and $\mathcal{O}$ are isomorphic. It is known that no non-trivial covering tasks can be solved wait-free in the IIS model [21]. Here, we consider a colorless version of covering tasks, and study impossibility proofs for solving them using colorless algorithms.

▶ **Theorem D.** *Every non-trivial covering task admits an FLP-style impossibility proof for $n \geq 2$ processes running wait-free colorless algorithms.*

The proof is based on showing that, for every covering task Π, the transformation F used in our speedup theorem satisfies $F(\Pi) = \Pi$, i.e., Π is itself a fixed point for F. As a consequence, since Π is not solvable in zero rounds (unless $\mathcal{I}$ and $\mathcal{O}$ are isomorphic, i.e., the task Π is trivial), there is a round-reduction impossibility proof for Π (Theorem 19), and the existence of an FLP-style impossibility proof for Π then follows from Theorem B. This last fact is of particular interest, as it shows that FLP-style proofs are not limited to cases where $\mathcal{O}$ is disconnected or to problems that are unsolvable even when restricted to a single input simplex and its faces (as is the case for consensus and approximate agreement).

2 Model and Definitions

We consider the wait-free iterated immediate snapshots model (IIS). This model and its variants have been frequently used e.g. [11,32,38] due to its simplicity, while being equivalent to the usual wait-free read/write shared memory model for task solvability [11, 22]. Furthermore, it is known that as far as colorless task solvability is concerned, one can assume *colorless computation* without loss of generality [23, 28], by which we mean that in each round of computation processes do no consider which process wrote a value, nor by how many processes it was written. We next provide a brief overview of the model.

Processes communicate through a sequence of shared memory objects, and the computation is split into rounds. In the i-th round, process p writes a value to the p-th location of the i-th object, and then takes a snapshot of all the i-th object. The *view* of a process at the end of a round is the set of values it read from the object, without the information of which

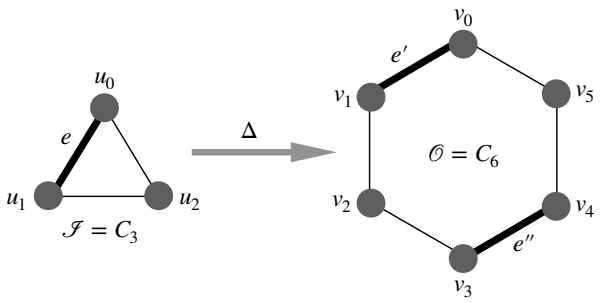

Figure 1 The Hexagone Task. In particular, $\Delta(e)$ is the complex $e' \cup e''$.

process wrote which value, nor by how many processes it was written to the memory. As common in lower bound proofs, we only consider full information protocols: in each round, each process writes its entire state (or equivalently, the view read from the previous object) to the memory.

2.1 Colorless Tasks

In this paper we consider colorless tasks [35]. See [23, Chapter 4] and [28] for an overview.

▶ **Definition 1.** *A* colorless *task* $\Pi = (\mathcal{I}, \mathcal{O}, \Delta)$ *is defined by an input simplicial complex* $\mathcal{I}$*, an output simplicial complex* $\mathcal{O}$*, and an input-output specification* $\Delta : \mathcal{I} \to 2^{\mathcal{O}}$ *mapping every simplex* $\sigma \in \mathcal{I}$ *to a sub-complex* $\Delta(\sigma)$ *of* $\mathcal{O}$ *with dimension at most* $\dim(\sigma)$*.*

The semantics of a colorless task $\Pi = (\mathcal{I}, \mathcal{O}, \Delta)$ is that every vertex of $\mathcal{I}$ is an input value, and every vertex of $\mathcal{O}$ is an output value. A set of processes may start with different input values in $V(\mathcal{I})$, as long as the set σ of these input values belongs to $\mathcal{I}$. To solve the task, it is required that any collection of processes starting with input values forming a set $\sigma \in \mathcal{I}$ outputs only output values in the vertices $V(\mathcal{O})$, as long as the set τ of these output values forms a simplex in $\Delta(\sigma)$.

The next two colorless tasks will serve as running examples in the rest of the paper.

The Set Agreement Task. Set agreement is a well studied relaxation of the classical consensus task. Let $k \geq 2$. *Set-agreement* with input set $[k] = \{1, \ldots, k\}$ is the colorless task $\mathrm{SA}_k = (\mathcal{I}, \mathcal{O}, \Delta)$ where $\mathcal{I} = \{\sigma \subseteq [k] \mid \sigma \neq \varnothing\}$, $\mathcal{O} = \{\tau \subseteq [k] \mid (\tau \neq \varnothing) \wedge (\tau \neq [k])\}$, and, for every $\sigma \in \mathcal{I}$,

$$\Delta(\sigma) = \{\tau \in \mathcal{O} \mid \tau \subseteq \sigma\}.$$

Said differently, $\Delta(\sigma) = \{\tau \subseteq \sigma\}$ if $\dim(\sigma) < k-1$, and $\Delta(\sigma) = \mathcal{O}$ otherwise. SA_k is solvable (in zero rounds) for $n < k$ processes, but not solvable for $n \geq k$ processes [10, 29, 40].

The Hexagone Task. The Hexagone Task is one basic example of the colorless *covering* tasks, thoroughly studied in Section 7.2. For every $t \geq 3$, let C_t denotes the t-node cycle. The *hexagone task* is the task $\mathrm{HX} = (C_3, C_6, \Delta)$ where $C_3 = (u_0, u_1, u_2)$, $C_6 = (v_0, v_1, \ldots, v_5)$ and Δ is defined as follows (see Figure 1). For every $i \in \{0, 1, 2\}$,

$$\Delta(u_i) = \{\{v_i\}, \{v_{i+3}\}\} \text{ and } \Delta(\{u_i, u_{i+1 \bmod 3}\}) = \{\{v_i, v_{i+1}\}, \{v_{i+3}, v_{i+4 \bmod 6}\}\},$$

where formally $\Delta(\{u_i, u_{i+1 \bmod 3}\})$ also contains all the vertices contained it its edges, i.e. $\{v_i\}, \{v_{i+1}\}, \{v_{i+3}\}, \{v_{i+4 \bmod 6}\}$. The map Δ is the inverse of the maps $f : V(C_6) \to V(C_3)$ defined as $f(u_i) = v_{i \bmod 3}$, where (C_6, f) is a covering of C_3.

2.2 Colorless Algorithms

The solvability of a colorless task may depend on the number n of processes involved in the computation. Remarkably, it is enough to consider *colorless computation* for colorless tasks, according to the following result (see, e.g., [28]), that summarizes and formalizes what we need to know about the model of computation[1]. The result holds both for the wait-free read/write memory model and for its iterated version, by the equivalence proved in [22].

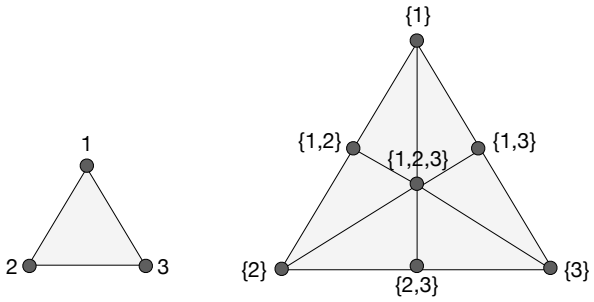

Figure 2 Barycentric subdivision.

▶ **Lemma 2.** *A colorless task $\Pi = (\mathcal{I}, \mathcal{O}, \Delta)$ is read/write solvable by n processes running a wait-free algorithm if and only if there exists $t \geq 0$ and a simplicial map*

$$f : \mathsf{Bary}^{(t)}(\mathsf{Skel}_n(\mathcal{I})) \to \mathcal{O}$$

that agrees with Δ, i.e., for every $\sigma \in \mathcal{I}$, $f(\mathsf{Bary}^{(t)}(\mathsf{Skel}_n(\sigma))) \subseteq \Delta(\sigma)$. Furthermore, Π is 1-round solvable by n processes running a wait-free colorless algorithm in the IIS model if and only if there is a simplicial map

$$f : \mathsf{Bary}(\mathsf{Skel}_n(\mathcal{I})) \to \mathcal{O}$$

that agrees with Δ.

In the above, $\mathsf{Bary}^{(t)}$ denotes t successive applications of the barycentric subdivision (see Fig. 2), and Skel_n denotes the $(n-1)$-dimensional skeleton operator, i.e., $\mathsf{Skel}_n(\mathcal{I})$ is the subcomplex of $\mathcal{I}$ (resp., of σ) including all simplices of $\mathcal{I}$ (resp., all faces of σ) with dimension at most $n-1$. When considering colored (general) tasks, a similar result holds when using chromatic simplicial maps and the *standard chromatic subdivision*; for colorless tasks, however, the barycentric subdivision suffices since if a colorless task is solvable then it is solvable by an algorithm ignoring multiplicities of identical views (or inputs) in the snapshots [28].

[1] Results similar to this one are known for several models of computation [23]. For instance, a corresponding result is known for Byzantine failures [23, Theorem 6.5.1] and dependent failures [26, Theorem 4.3] with an adversary of core size $c = 2$. For dependent failures in the case of wait-free computation, the theorem states that a colorless task is solvable if and only if there exists a continuous map between geometric realization $f : \mathsf{Skel}_{c-1}(\mathcal{I}) \to \mathcal{O}$ carried by Δ.

3 Round-Reduction Proofs

This section is essentially dedicated to the proof of Theorem A. The task transformation Cl in this theorem is based on a *closure* operator similar to the one in [20], where the main difference is that our closure operator is required to be colorless, and we stress that our result is not implied by the (colored) theorem of [20]. That is, given a class $\mathcal{A}$ of algorithms, requiring the closure operator to be in the class may not be sufficient for extending the speedup theorem in [20] to apply to algorithms in $\mathcal{A}$. We illustrate this in the full version of the paper for *comparison-based* algorithms, an important class of algorithms used for studying tasks such as renaming and weak symmetry-breaking. Using a comparison-based closure operator does not suffice for deriving a speedup theorem for comparison-based algorithms. On the other hand, in this section we show that for colorless algorithms, restricting the closure operator to be colorless suffices.

3.1 Colorless Closure

We first rephrase the notions of local tasks introduced in [20] in the context of colorless tasks.

▶ **Definition 3.** *Let $\Pi = (\mathcal{I}, \mathcal{O}, \Delta)$ be a colorless task, let $\sigma \in \mathcal{I}$, and let $\tau \subseteq V(\Delta(\sigma))$. Let us consider τ as a simplicial complex (with a unique facet). The* local task *with respect to σ and τ is the colorless task $\Pi_{\tau,\sigma} = (\tau, \Delta(\sigma), \Delta_{\tau,\sigma})$ where, for every face τ' of τ,*

$$\Delta_{\tau,\sigma}(\tau') = \begin{cases} v & \text{if } \tau' = v \text{ is a vertex,} \\ \mathsf{Skel}_{\dim(\tau')}(\Delta(\sigma)) & \text{otherwise.} \end{cases}$$

Note that $\Pi_{\tau,\sigma}$ is a well-defined colorless task, because $\mathsf{Skel}_{\dim(\tau')}$ guarantees that the output complex for τ' has dimension at most $\dim(\tau')$; this is true even if τ is not a simplex of $\Delta(\sigma)$, as seen in the second example in Section 3.1. Note also that the validity constraint of $\Pi_{\tau,\sigma}$ is just that if all processes start with the same input v, i.e., if they start from the same vertex $v \in V(\tau)$, then they must all output v. Without this constraint, the processes are only constrained to output values forming a legal set σ' of outputs w.r.t. σ, i.e., a set $\sigma' \in \Delta(\sigma)$ (and $\dim(\sigma') \leq \dim(\tau')$).

As opposed to the general (chromatic) tasks, which each has a fixed maximum number of participating processes, colorless tasks are defined for any number of processes. In particular, a colorless task may be solvable for a certain number of processes, but not for another number of processes. The definition below refers to solving local tasks with a prescribed number of processes.

▶ **Definition 4.** *The* colorless closure *of a colorless task $\Pi = (\mathcal{I}, \mathcal{O}, \Delta)$ is the colorless task $\mathsf{Cl}(\Pi) = (\mathcal{I}, \mathcal{O}', \Delta')$ where $V(\mathcal{O}') = V(\mathcal{O})$, and, for every $\sigma \in \mathcal{I}$, and every non-empty set $\tau \subseteq V(\mathcal{O})$, we set $\tau \in \Delta'(\sigma)$ if $\tau \subseteq V(\Delta(\sigma))$ and $\Pi_{\tau,\sigma}$ is solvable in one round by $\dim(\tau)+1$ processes running a colorless algorithm. The simplices of $\mathcal{O}'$ are the images of Δ', and all their faces.*

Note that this definition is constructive, i.e., one can check the 1-round solvability of $\Pi_{\tau,\sigma}$, by Lemma 2. The closure operator also has the nice property that it does not change the allowed input and output values, and the only difference between Π and $\mathsf{Cl}(\Pi)$ is in the addition of some allowed combinations of output values.

For a simplex $\tau \in \Delta(\sigma)$, the local task $\Pi_{\tau,\sigma}$ is solvable in 0 rounds, by having each process starting with input $v \in V(\tau)$ output v. It follows that if $\tau \in \Delta(\sigma)$ then $\tau \in \Delta'(\sigma)$, and therefore, for every $\sigma \in \mathcal{I}$, $\Delta(\sigma) \subseteq \Delta'(\sigma)$. As a consequence, the colorless closure $\mathsf{Cl}(\Pi)$ of

a task Π is not more difficult to solve than Π. Whether or not $\mathsf{Cl}(\Pi)$ is *simpler* to solve than Π is one of the foci of this paper. Before going further, we establish hereafter that the input-output specification Δ' of the colorless closure of a colorless task is a *carrier* map, which is a condition that is often required for a task [23].

The I/O-Specification of a Colorless Closure is a Carrier Map. Let $\Pi = (\mathcal{I}, \mathcal{O}, \Delta)$ be a colorless task. Recall that Δ is a *carrier* map if, for every σ and σ' in $\mathcal{I}$, $\sigma \subseteq \sigma' \Longrightarrow \Delta(\sigma) \subseteq \Delta(\sigma')$, where the latter inclusion must be read as $\Delta(\sigma)$ is a subcomplex of $\Delta(\sigma')$. Being a carrier map is not necessary for Π to be solvable. However, if two simplices σ and σ' in $\mathcal{I}$ satisfy $\sigma \subseteq \sigma'$ and $\Delta(\sigma) \setminus \Delta(\sigma') \neq \varnothing$, the output values outside $\Delta(\sigma')$ cannot be used for a set of processes starting with input σ' since these output values cannot be extended in case processes with inputs in $\sigma' \setminus \sigma$ eventually participate later. Therefore, we may as well remove all simplices from $\Delta(\sigma)$ that are not in $\Delta(\sigma')$, and restrict ourselves to input-output specifications that are carrier maps.

▶ **Lemma 5.** *Let $\Pi = (\mathcal{I}, \mathcal{O}, \Delta)$ be a colorless task, and let $\mathsf{Cl}(\Pi) = (\mathcal{I}, \mathcal{O}', \Delta')$. If Δ is a carrier map then Δ' is a carrier map.*

Proof. Consider two simplices $\sigma, \sigma' \in \mathcal{I}$ satisfying $\sigma \subseteq \sigma'$, and let $\tau \in \Delta'(\sigma)$. By definition, the local task $\Pi_{\tau,\sigma} = (\tau, \Delta(\sigma), \Delta_{\tau,\sigma})$ is solvable in one round, by a simplicial map $g : \mathsf{Bary}^{(1)}(\tau) \to \Delta(\sigma)$ that agrees with $\Delta_{\tau,\sigma}$. To show that $\tau \in \Delta'(\sigma')$, we show that the local task $\Pi_{\tau,\sigma'} = (\tau, \Delta(\sigma'), \Delta_{\tau,\sigma'})$ is solvable in one round, using the same map g. Since $\Delta(\sigma) \subseteq \Delta(\sigma')$, g is a simplicial map from $\mathsf{Bary}^{(1)}(\tau)$ to $\Delta(\sigma')$. To see that g agrees with $\Delta_{\tau,\sigma'}$, let $\tau' \subseteq \tau$. If $\dim(\tau') > 0$, then $g(\tau') \in \Delta_{\tau,\sigma}(\tau') = \Delta(\sigma) \subseteq \Delta(\sigma') = \Delta_{\tau,\sigma'}(\tau')$. If $\tau' = \{v\}$ is a vertex, then $g(v) \in \Delta_{\tau,\sigma}(v) = v = \Delta_{\tau,\sigma'}(v)$. It follows that g agrees with $\Delta'_{\tau,\sigma'}$, and thus $\tau \in \Delta'(\sigma')$, from which we conclude that $\Delta'(\sigma) \subseteq \Delta'(\sigma')$, i.e., Δ' is a carrier map. ◀

Examples

- Let $k \geq 3$. For the set-agreement task SA_k, the closure is solvable in zero rounds by having each process outputting its input, since any combination of at most k input values forms a valid output configuration of $\mathsf{Cl}(\mathrm{SA}_k)$. To prove this, we show that for every $\sigma \in \mathcal{I}$ we have $\sigma \in \Delta'(\sigma)$. First, note that for $\sigma \in \mathcal{I}$ such that $\sigma \neq [k]$, we have $\sigma \in \Delta(\sigma)$ and $\sigma \in \Delta(\sigma) \Rightarrow \sigma \in \Delta'(\sigma)$, so we only have to show $[k] \in \Delta'([k])$. This is true since the local task $\Pi_{[k],[k]}$ is solvable in one round, by letting each process that sees more than one value output 1.
- On the other hand, for the Hexagon task HX we have $\mathsf{Cl}(\mathrm{HX}) = \mathrm{HX}$. To see this, consider an input simplex $\sigma = (\{u_i, u_{i+1 \bmod 3}\})$, for some $i \in \{0, 1, 2\}$, its image $\Delta(\{u_i, u_{i+1 \bmod 3}\}) = \{v_i, v_{i+1}\} \cup \{v_{i+3}, v_{i+4 \bmod 6}\}$, and two vertices in its image that do not already constitute a simplex, i.e. $w \in \{v_i, v_{i+1}\}$ and $w' \in \{v_{i+3}, v_{i+4 \bmod 6}\}$. Let $\tau = \{w, w'\}$.
  If the local task $\Pi_{\tau,\sigma} = (\tau, \Delta(\sigma), \Delta_{\tau,\sigma})$ would have been solvable in one round, then there would have been a map $f : V(\mathsf{Bary}^{(1)}(\tau)) \to \Delta(\sigma)$ satisfying $f(\{w\}) = w$ and $f(\{w'\}) = w'$. But $\mathsf{Bary}^{(1)}(\{w, w'\})$ is

  $$\{w\} \text{ ——— } \{w, w'\} \text{ ——— } \{w'\}$$

  and any such map cannot be simplicial by continuity: if $f(\{w, w'\}) \in \{v_i, v_{i+1}\}$ then $\{f(\{w'\}), f(\{w, w'\})\} \notin \Delta(\sigma)$, and similarly if $f(\{w, w'\}) \in \{v_{i+3}, v_{i+4 \bmod 6}\}$ then $\{f(\{w\}), f(\{w, w'\})\} \notin \Delta(\sigma)$.

3.2 Colorless Speedup Theorem

We now establish our speedup theorem for colorless algorithms, as stated next and proved in the full version of the paper.

▶ **Theorem 6.** *For every colorless task $\Pi = (\mathcal{I}, \mathcal{O}, \Delta)$, every $t > 0$, and every $n \geq 2$, if Π is solvable in t rounds by n processes running a wait-free colorless algorithm then the closure $\mathsf{Cl}(\Pi)$ is solvable in $t-1$ rounds by n processes running a wait-free colorless algorithm.*

Since the colorless closure task $\mathsf{Cl}(\Pi) = (\mathcal{I}, \mathcal{O}', \Delta')$ of a colorless task $\Pi = (\mathcal{I}, \mathcal{O}, \Delta)$ only potentially adds valid output simplices to $\Delta(\sigma)$ for forming $\Delta'(\sigma)$, for every $\sigma \in \mathcal{I}$, it follows that, after applying the closure operator for some finite number of times t, we get a fixed point, i.e. a task $\mathsf{Cl}^{(t)}(\Pi)$ such that $\mathsf{Cl}^{(t+1)}(\Pi) = \mathsf{Cl}^{(t)}(\Pi)$. Naturally, this also implies $\mathsf{Cl}^{(t')}(\Pi) = \mathsf{Cl}^{(t)}(\Pi)$ for every $t' \geq t$.

▶ **Definition 7.** *The* fixed-point *of a colorless task Π is the task $\Pi^* = (\mathcal{I}, \mathcal{O}^*, \Delta^*)$ such that $\Pi^* = \mathsf{Cl}^{(t)}(\Pi)$ for some $t \geq 0$, and $\mathsf{Cl}^{(t+1)}(\Pi) = \mathsf{Cl}^{(t)}(\Pi)$.*

As a direct consequence of the speedup theorem (Theorem 6) the fixed-point task Π^* of a task Π is either 0-round solvable, or not solvable at all. Indeed, consider $\Pi^* = (\mathcal{I}, \mathcal{O}^*, \Delta^*)$ and assume it is solvable in $t > 0$ rounds. By Theorem 6, $\Pi^* = \mathsf{Cl}(\Pi^*)$ is solvable in $t-1$ rounds. Repeating this argument for t times implies that Π^* is 0-round solvable.

▶ **Lemma 8.** *Let $n \geq 2$. Given a colorless task $\Pi = (\mathcal{I}, \mathcal{O}, \Delta)$, its fixed-point Π^* is either 0-round solvable by n processes running a wait-free colorless algorithm, or not solvable at all.*

The next corollary illustrates both the Theorem 6's interest and its simplicity.

▶ **Corollary 9.** *For every $n \geq 2$, the hexagon task cannot be solved by n processes running a wait-free colorless algorithm.*

Proof. If HX is solvable wait-free by $n \geq 2$ processes in the IIS model, then there exists $t \geq 0$ such that HX is solvable in t rounds. We have seen that $\mathsf{Cl}(\mathrm{HX}) = \mathrm{HX}$. It follows from Theorem 6 that if HX is solvable wait-free, then it is solvable wait-free in zero rounds.

Consider a possible zero-round algorithm for HX with $n \geq 2$ processes, and its decision map δ. As the algorithm must produce valid outputs for executions with a single input, we have $\delta(u_i) \in \{v_i, v_{i+3}\}$ for every $i \in \{0, 1, 2\}$. Let $v_j = \delta(u_0)$ (and hence $j \in \{0, 3\}$). The definition of Δ for executions with two different inputs guarantees $\delta(v_1) \in \{u_j, u_{j+1}\}$ and by the above we have $\delta(v_1) = u_{j+1}$. Similarly, $\delta(v_2) = u_{j+2}$, and hence $\delta\{v_0, v_2\} = \{u_j, u_{j+2}\} \notin \Delta\{v_0, v_2\}$, a contradiction. ◀

4 The Topology of the Closure

In this section, we study the topology of the colorless closure task, which, as opposed to the general closure, displays very simple properties. For stating these properties, let us recall that a complex $\mathcal{K}$ is *complete* if $V(\mathcal{K})$ is a simplex of $\mathcal{K}$. It is *complete up to dimension d* if every set $\tau \subseteq V(\mathcal{K})$ with $0 \leq \dim(\tau) \leq d$ satisfies $\tau \in \mathcal{K}$. A *connected component* of a complex $\mathcal{K}$ is the subcomplex of $\mathcal{K}$ induced by all the vertices in a connected component of the graph $\mathsf{Skel}_1(\mathcal{K})$. We mainly show that, for every $\sigma \in \mathcal{I}$, the closure of each connected component of $\Delta(\sigma)$ eventually becomes complete after a bounded number of closure operations.

For a colorless task $\Pi = (\mathcal{I}, \mathcal{O}, \Delta)$, we denote by $\mathsf{Cl}^{(k)}(\Pi) = (\mathcal{I}, \mathcal{O}^{(k)}, \Delta^{(k)})$ the k-th closure of Π, defined as $\mathsf{Cl}^{(k)}(\Pi) = \mathsf{Cl}(\mathsf{Cl}^{(k-1)}(\Pi))$, where k is a positive integer and $\mathsf{Cl}^{(0)}(\Pi) = \Pi$.

▶ **Theorem 10.** *Given a colorless task* $\Pi = (\mathcal{I}, \mathcal{O}, \Delta)$ *and a simplex* $\sigma \in \mathcal{I}$*, the following hold.*

- *Let* D *be the largest diameter of a connected component in the graph* $\mathsf{Skel}_1(\Delta(\sigma))$*, and let* $\ell = \lceil \log_2 D \rceil + 1$*. Then all the connected components of* $\Delta^{(\ell)}(\sigma)$ *are complete up to dimension* $\dim(\sigma)$*.*
- *For every* $k \geq 0$*, a set of vertices in* $V(\Delta(\sigma))$ *is a connected component of* $\mathsf{Skel}_1(\Delta(\sigma))$ *if and only if it is a connected component of* $\mathsf{Skel}_1(\Delta^{(k)}(\sigma))$*.*

See Appendix A for the proof of this theorem.

Examples

- In Section 3.1 we have proved that the closure of k-set agreement contains any combination of at most k input values. This fact can now be directly derived from Theorem 10, as $\Delta([k])$ is connected and contains all the values of $[k]$.
- In the same section, we have proved that $\mathsf{Cl}(\mathrm{HX}) = \mathrm{HX}$. This is also a direct consequence of Theorem 10, as for every $\sigma \in \mathcal{I}$, each of the connected components of $\Delta(\sigma)$ is full.

5 Round-Reduction is Complete for 1-Dimensional Tasks

1-dimensional tasks are tasks for which the input and output complexes are of dimension at most 1. In this section, we establish the completeness of the round-reduction proof technique for 1-dimensional colorless tasks, thus establishing part of Theorem C. FLP-style proofs are also complete in this case, but the proof of this fact is deferred to the next section, where we show that the techniques are equivalent. The next theorem asserts that for colorless tasks of dimension at most 1 the reciprocal of Theorem 6 holds as well, thus establishing the completeness of the round-reduction technique in this case; see Appendix A for the proof.

▶ **Theorem 11.** *For every* 1*-dimensional colorless task* $\Pi = (\mathcal{I}, \mathcal{O}, \Delta)$*, every* $t > 0$*, and every* $n \geq 2$*, if the colorless closure* $\mathsf{Cl}(\Pi)$ *is solvable in* $t-1$ *rounds by* n *processes running a wait-free colorless algorithm, then* Π *is solvable in* t *rounds by* n *processes running a wait-free colorless algorithm.*

By combining Theorems 6 and 11, we obtain the desired result.

▶ **Corollary 12.** *The round-reduction proof techniques is complete for 1-dimensional colorless tasks and wait-free colorless algorithms.*

6 Relations Between Round-Reduction and FLP-Style Proofs

In this section we establish tight connections between the FLP-style proof strategy and the round-reduction proof technique, in the context of wait-free solvability of colorless tasks. Specifically, we first establish Theorem B which asserts that the existence of a round-reduction proof implies the existence of an FLP-style proof, from which the remaining part of Theorem C (completeness of FLP-style proofs for 1-dimensional tasks) will follow. This implies that the round-reduction techniques and FLP-style proofs have the same power when considering impossibility proofs for 1-dimensional colorless tasks. In Theorem 16, we give a direct proof for this: the existence of an FLP-style proof implies the existence of a round-reduction proof for the impossibility of such tasks (complementing Theorem B, for 1-dimensional tasks). This direct proof may suggest that a more tight connection between the two proof techniques exists. We start by formally defining FLP-style proofs.

6.1 FLP-Style Proofs

Given a colorless task $\Pi = (\mathcal{I}, \mathcal{O}, \Delta)$, an FLP-style proof constructs an infinite sequence of simplices $\sigma_0, \sigma_1, \ldots$, where $\sigma_0 \in \mathcal{I} = \mathsf{Bary}^{(0)}(\mathcal{I})$, and, for every $t \geq 1$, $\sigma_t \in \mathsf{Bary}^{(1)}(\sigma_{t-1}) \subseteq \mathsf{Bary}^{(t)}(\mathcal{I})$, as follows.

The proof P assumes for contradiction the existence of an algorithm A solving Π. P starts by asking A to reveal, for every $\sigma \in \mathcal{I}$, the *valency* of σ, that is, the set of output values that are returned by A in executions starting with the input values forming the simplex σ. Then, P chooses a simplex $\sigma_0 \in \mathcal{I}$; in order to create an infinite sequence, P must choose σ_0 such that A is not able to claim it terminates in 0 rounds, i.e., such that any possible assignment of outputs to the processes in σ_0 is inconsistent with the valencies of $\mathcal{I}$.

Given a sequence $\sigma_0, \ldots, \sigma_t$ constructed by P so far, σ_{t+1} is obtained analogously, as follows. P asks A to reveal the valencies of all simplices in $\mathsf{Bary}^{(1)}(\sigma_t)$, that is, for each $\sigma \in \mathsf{Bary}^{(1)}(\sigma_t)$, the set of output values produced by A in all valid executions starting from σ. Based on these valencies, P chooses one simplex $\sigma_{t+1} \in \mathsf{Bary}^{(1)}(\sigma_t)$; as in the choice of σ_0, it must choose σ_{t+1} such that A is not able to assign outputs to the processes in σ_{t+1} consistent with the valencies of $\mathsf{Bary}^{(1)}(\sigma_t)$.

Let $\mathsf{val} : 2^{\sigma_0} \cup 2^{\sigma_1} \cup \cdots \to 2^{\mathcal{O}}$ be the function defined by the valencies returned by A. For any correct algorithm, the function val must satisfy some basic conditions of consistency with the task specification and with other valencies returned. We next formalize these conditions for σ_t.

- Consistent with itself: each $\sigma \in \mathsf{Bary}^{(1)}(\sigma_t)$ satisfies $\mathsf{val}(\sigma) \subseteq \mathsf{val}(\sigma_t)$; moreover, $\cup_{\sigma \in \mathsf{Bary}^{(1)}(\sigma_t)} \mathsf{val}(\sigma) = \mathsf{val}(\sigma_t)$.
- Consistent with Δ: For each $\sigma \in \mathsf{Bary}^{(1)}(\sigma_t)$, $\mathsf{val}(\sigma) \subseteq \Delta(\sigma_0)$.
- Monotone: for each $\sigma' \subseteq \sigma \in \mathsf{Bary}^{(1)}(\sigma_t)$, $\mathsf{val}(\sigma') \subseteq \mathsf{val}(\sigma)$.

If this strategy can proceed forever, constructing an infinite sequence $\sigma_0, \sigma_1, \ldots$ of simplices, then A does not terminate in this execution, disproving the existence of an algorithm A solving Π. At the core of FLP-style proofs stands a *choice mechanism* that picks the next simplex σ_{t+1}. We present such a mechanism for one-dimensional colorless tasks.

Our work on the FLP-style technique continues recent lines of work regarding the power of extension-based proofs [2, 5, 14]. We allow less diverse queries compared to these works, yet our results imply that if a one-dimensional colorless task is unsolvable, then the simple queries we allow are sufficient for proving this impossibility.

Example. The impossibility of solving the hexagon task can be proved using an FLP-style proof, by constructing a sequence $\sigma_0, \sigma_1, \ldots$ as follows. All the simplices $\sigma_0, \sigma_1, \ldots$ will be edges, and we will fix $i \in \{0, 1, 2\}$ such that each σ_t satisfies the invariants

(1) $\mathsf{val}(\sigma_t) \subseteq \{v_i, v_{i+1}\} \cup \{v_{i+3}, v_{i+4}\}$, and

(2) $\mathsf{val}(\sigma_t) \cap \{v_i, v_{i+1}\} \neq \emptyset$ and $\mathsf{val}(\sigma_t) \cap \{v_{i+3}, v_{i+4}\} \neq \emptyset$,

where the indices here and below are computed modulo 6, unless otherwise specified.

To choose σ_0 and i, inspect the valencies of the three edges $(u_i, u_{i+1 \bmod 3})$, for $i \in \{0, 1, 2\}$. Since $\mathsf{val}(u_i) \subseteq \mathsf{val}(u_{i-1 \bmod 3}, u_i) \cap \mathsf{val}(u_i, u_{i+1 \bmod 3})$ and valency is monotone, the valencies of every two edges must intersect, and thus for some $i \in \{0, 1, 2\}$ the edge $e = (u_i, u_{i+1 \bmod 3})$ must satisfy both $\mathsf{val}(e) \cap \{v_i, v_{i+1}\} \neq \emptyset$ and $\mathsf{val}(e) \cap \{v_{i+3}, v_{i+4}\} \neq \emptyset$; we fix this i, and set $\sigma_0 = e$, guaranteeing (2). Invariant (1) holds since the valency must be consistent with Δ.

Assume $\sigma_0, \ldots, \sigma_t$ are chosen and satisfy both invariants, and that $\mathsf{val}(\sigma)$ is known for each $\sigma \in \mathsf{Bary}_1(\sigma_t)$. Since $\mathsf{val}(\sigma) \subseteq \mathsf{val}(\sigma_t)$ for every $\sigma \in \mathsf{Bary}_1(\sigma_t)$, Invariant (1) will hold for any $\sigma_{t+1} \in \mathsf{Bary}_1(\sigma_t)$ we may choose. Let $\sigma_t = \{w_0, w_1\}$, then $\mathsf{Bary}_1(\sigma_t)$ is composed of the vertices $\{w_0\}, \{w_0, w_1\}, \{w_1\}$ and the edges $e_0 = (\{w_0\}, \{w_0, w_1\})$ and $e_1 = (\{w_1\}, \{w_0, w_1\})$.

We have $\mathsf{val}(e_0) \cap \mathsf{val}(e_1) \neq \emptyset$, and by Invariant (1) we also have $\mathsf{val}(e_0) \cap \mathsf{val}(e_1) \subseteq \mathsf{val}(\sigma_t) \subseteq \{v_i, v_{i+1}, v_{i+3}, v_{i+4}\}$, so at least one value $v \in \{v_i, v_{i+1}, v_{i+3}, v_{i+4}\}$ satisfies $v \in \mathsf{val}(e_0) \cap \mathsf{val}(e_1)$; assume without loss of generality that $v \in \{v_i, v_{i+1}\}$.

The valencies of the vertices are contained in the valencies of the edges, hence $\mathsf{val}(e_0) \cup \mathsf{val}(e_1) = \mathsf{val}(\sigma_t)$, so Invariant (2) implies that both $(\mathsf{val}(e_0) \cup \mathsf{val}(e_1)) \cap \{v_i, v_{i+1}\} \neq \emptyset$ and $(\mathsf{val}(e_0) \cup \mathsf{val}(e_1)) \cap \{v_{i+3}, v_{i+4}\} \neq \emptyset$ hold. Hence, at least one edge $e \in \{e_0, e_1\}$ has $\mathsf{val}(e) \cap \{v_{i+3}, v_{i+4}\} \neq \emptyset$, and since $v \in \mathsf{val}(e)$ it also have $\mathsf{val}(e) \cap \{v_i, v_{i+1}\} \neq \emptyset$. This edge is set as σ_{t+1}, and Invariant (2) is satisfied for σ_{t+1} as well. Since this process can continue for every $t \geq 0$, the proof is complete.

6.2 Connections Between the Proof Techniques

The next theorem, Theorem B, states that the existance of a round reduction impossibility proof implies the existence of an FLP-style proof. The theorem is proved in Appendix A.

▶ **Theorem 13.** *For every colorless task* Π *and* $n \geq 2$*, if there is a round-reduction proof establishing the impossibility of solving* Π *by* n *processes running a wait-free colorless algorithm, then there is an FLP-style proof establishing the same impossibility.*

We next prove Theorem C for FLP-style proofs, i.e. we show these are complete for 1-dimensional colorless tasks. For this, first observe that if there is an FLP-style proof for the impossibility of a 1-dimensional colorless task Π, then Π is unsolvable. By the completeness of the round-reduction proofs (cf. Corollary 12), there is a round-reduction impossibility proof for that task. On the other hand, Theorem 13 asserts that if a colorless task has a round-reduction impossibility proof then it also has an FLP-style impossibility proof. The establishes the desired equivalence between the two forms of proofs for colorless tasks.

▶ **Corollary 14.** *Let* Π *be a* 1*-dimensional colorless task. There is an FLP-style proof for the impossibility of solving* Π *using wait-free colorless algorithms if and only if there is a round-reduction proof of this impossibility for* Π*.*

This corollary can also be proved directly: one direction is Theorem 13 (for any dimension), and the other is given in Theorem 16 (below). We get that, for 1-dimensional tasks, the FLP-style proof style can be mechanized, i.e., for any 1-dimensional task Π, the FLP-style proof technique succeeds for Π if and only if Π is not wait-free solvable in the IIS model.

▶ **Corollary 15.** *The FLP-style proof technique is complete for* 1*-dimensional tasks.*

Proof. By Theorem 13, if the FLP-style proof technique fails for Π, then the round-reduction proof also fails for Π as well. By Lemma 8, this implies that Π^* is 0-round solvable. By Theorem 11, the original task Π is solvable. ◀

In the full version of this paper, we also give a direct proof for the converse of Theorem 13 for 1-dimensional tasks, stated next.

▶ **Theorem 16.** *For every* 1*-dimensional colorless task* Π *and* $n \geq 2$*, if there is an FLP-style proof for the impossibility of solving* Π *by* n *processes running a wait-free colorless algorithm, then there is a round-reduction proof for the same impossibility.*

7 Applications

Throughout this paper, we have shown how the theory we developed applies for set agreement and the Hexagon task. We complete the paper by presenting some further applications. Note that in terms of techniques, all these proofs are completely different from previous proofs of similar results: round-reduction works directly on the task specification, in an algorithmic way that does not depend on the specific task at hand. Hence, the arguments in round-reduction proofs are applied directly on the task specification, and not on executions of a protocol (which are encapsulated in the round-reduction theorem).

7.1 Time Lower Bound for Approximate Agreement

Let us show that the bound $\lceil \log_2 D \rceil$ in Theorem 10 is tight. For this purpose, consider the approximate agreement task. For an integer $N \geq 1$, let $\epsilon = 1/N$, and the ϵ-agreement task defined as follows. The input complex $\mathcal{I}$ of ϵ-agreement is merely the edge

$0 \text{ —— } 1$

The output complex $\mathcal{O}$ is the path

$0 \text{ —— } \epsilon \text{ —— } 2\epsilon \text{ —— } \ldots \text{ —— } (N-1)\epsilon \text{ —— } 1$

Finally, the input-output specification Δ satisfies for each set $S \subseteq \mathcal{I}$

$$\Delta(S) = \{T \subseteq \mathcal{O} \mid \min S \leq \min T \text{ and } \max T \leq \max S\}$$

and specifically, for an element $x \in \mathcal{I}$ it specifies $\Delta(\{x\}) = \{x\}$.

▶ **Proposition 17.** *For every $\epsilon \in (0,1)$, ϵ-agreement cannot be solved by $n \geq 2$ processes in less than $\lceil \log_2 1/\epsilon \rceil$ rounds.*

The proof of this preposition appears in the appendix. Note that for $n > 2$ processes this bound is tight, and is the same for colored and colorless algorithm. Interestingly, for $n = 2$ processes there is a colored algorithm requiring only $\lceil \log_3 1/\epsilon \rceil$ rounds [4, 30]. Hence, while colorless and colored algorithms have the same computability power, colored algorithms are provably stronger in terms of time complexity.

7.2 Impossibility of Covering Tasks

Recall that, for two connected simplicial complexes $\mathcal{I}$ and $\mathcal{O}$, and for a simplicial map $f : \mathcal{O} \to \mathcal{I}$, the pair $(\mathcal{O}, f)$ is a *covering complex* of $\mathcal{I}$ if, for every $\sigma \in \mathcal{I}$, $f^{-1}(\sigma)$ is a union of pairwise disjoint simplexes. This condition can be rephrased as $f^{-1}(\sigma) = \cup_{i=1}^{k} \tau_i$ with $f_{|\tau_i} : \tau_i \to \sigma$ is one-one. The simplexes τ_i, $i = 1, \ldots, k$, are called the *sheets* of σ. We often refer to f as a *covering map*. The following observations follow directly from the definition of covering complex (see, e.g., [39]).

- If $\sigma \in \mathcal{I}$ is a simplex of dimension d, each sheet τ_i of σ is also a simplex of dimension d.
- The two complexes $\mathcal{I}$ and $\mathcal{O}$ are *locally isomorphic*, in the sense that for each vertex $v \in \mathcal{O}$ the complex star(v) is isomorphic to the complex star$(f(v))$. (The star of a vertex v in a complex $\mathcal{K}$ is the complex star(v) consisting of all the simplexes of $\mathcal{K}$ that contain v.
- All the simplices in $\mathcal{O}$ have the same number of sheets.)

We define below a colorless variant of the chromatic covering tasks introduced in [21].

▶ **Definition 18.** *Given a covering complex $(\mathcal{O}, f)$ of a complex $\mathcal{I}$, the colorless* covering task *$(\mathcal{I}, \mathcal{O}, \Delta)$ is the task where Δ is defined, for every $\sigma \in \mathcal{I}$, by*

$$\Delta(\sigma) = \{\tau \in \mathcal{O} \mid f(\tau) \subseteq \sigma\},$$

where $f(\tau) \subseteq \sigma$ means that $f(\tau)$ is a sub-complex of the complex defined by σ and all its faces. A covering complex is non-trivial *if each simplex in $\mathcal{I}$ has more than one sheet.*

The Hexagone task discussed above is a basic example of a covering task, and another example of a covering task can be see in Figure 4 in the appendix.

We now turn to a general impossibility result for covering tasks, proved in Appendix A.

▶ **Theorem 19.** *No non-trivial colorless covering tasks can be solved by $n \geq 2$ processes running a wait-free colorless algorithm.*

A similar result was proved in the past using an ad-hoc argument [21] for colored covering tasks, and here we give a round-reduction based proof for colorless covering tasks. By Theorem 13, this also means that the claim has an FLP-style proof, proving Theorem D.

8 Conclusion

The purpose of this paper is to relate round-reduction proof techniques (formally stated in the Speedup Theorem) and FLP-style proof techniques, when applied to colorless tasks within the framework of wait-free computing in the IIS model.

The round-reduction technique offers many good features, including the fact that it is mechanical (it is sufficient to check whether the fixed-point closure is solvable in zero rounds), it enables to derive not only impossibility results but also complexity lower bounds (e.g., for approximate agreement), and it extends to wait-free computing in models stronger than IIS (e.g., IIS augmented with Test&Set objects). On the other hand, FLP-style proofs are very generic, and essentially apply to all models, including t-resilient models. Moreover, we have shown that FLP-style proofs are not weaker than round-reduction proofs, and it is possible that they are stronger. Nevertheless, we have also shown that for 1-dimensional colorless tasks the two techniques have exactly the same power, and are both complete in the sense that if a task is not solvable then any of the two techniques will enable to establish this fact.

It would be interesting to know whether the equivalence between the two techniques holds for arbitrary colorless tasks, and not only for the 1-dimensional ones, and we conjecture that this is indeed the case. Note however that if this conjecture is true, then these two proof techniques cannot be complete, simply because it is known that set-agreement impossibility has no extension-based proof [2, 5].

The round-reduction CI we define and study in this work is not the only one possible (nor are F_{IIS} used in [20]); defining a more powerful operator for proving lower bounds on general algorithms is a central open question left in [20], and we leave a similar question open here with regard to colorless algorithm. Nevertheless, we have shown that for 1-dimensional colorless tasks, CI is in fact the most powerful operator possible – this operator is shown to be complete for such tasks in Corollary 12. The more general question of whether there exists if-and-only-if operators for wait-free computing, as was shown for synchronous failure-free computing in networks, is another central open question.

Another research direction is to try to design an analog of round-reduction for other computational models, such as t-resilient models (the speedup theorem of [20], as well as ours, assume the ability of each process to run solo). The ultimate goal of the line of study initiated in this paper is to better understand the relation between backward and forward induction in the context of distributed computing.

References

1. Marcos Kawazoe Aguilera and Sam Toueg. A simple bivalency proof that t-resilient consensus requires $t + 1$ rounds. *Inf. Process. Lett.*, 71(3-4):155–158, 1999. doi:10.1016/S0020-0190(99)00100-3.
2. Dan Alistarh, James Aspnes, Faith Ellen, Rati Gelashvili, and Leqi Zhu. Why extension-based proofs fail. In Moses Charikar and Edith Cohen, editors, *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing, STOC*, pages 986–996. ACM, 2019. doi:10.1145/3313276.3316407.
3. Dan Alistarh, Faith Ellen, and Joel Rybicki. Wait-free approximate agreement on graphs. In Tomasz Jurdzinski and Stefan Schmid, editors, *Structural Information and Communication Complexity - 28th International Colloquium, SIROCCO*, volume 12810 of *Lecture Notes in Computer Science*, pages 87–105. Springer, 2021. doi:10.1007/978-3-030-79527-6_6.
4. J. Aspnes and M. Herlihy. Wait-free data structures in the asynchronous PRAM model. In *2nd ACM Symposium on Parallel Algorithms and Architectures (SPAA)*, pages 340–349, 1990. doi:10.1145/97444.97701.
5. Hagit Attiya, Armando Castañeda, and Sergio Rajsbaum. Locally solvable tasks and the limitations of valency arguments. *J. Parallel Distributed Comput.*, 176:28–40, 2023. doi:10.1016/j.jpdc.2023.02.002.
6. Hagit Attiya and Faith Ellen. *Impossibility Results for Distributed Computing*. Synthesis Lectures on Distributed Computing Theory. Morgan & Claypool Publishers, 2014. doi:10.2200/S00551ED1V01Y201311DCT012.
7. Hagit Attiya and Jennifer Welch. *Distributed Computing: Fundamentals, Simulations and Advanced Topics*. John Wiley & Sons, Hoboken, NJ, USA, 2004.
8. Alkida Balliu, Sebastian Brandt, Juho Hirvonen, Dennis Olivetti, Mikaël Rabie, and Jukka Suomela. Lower bounds for maximal matchings and maximal independent sets. In *60th IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 481–497, 2019. doi:10.1109/FOCS.2019.00037.
9. Ofer Biran, Shlomo Moran, and Shmuel Zaks. A combinatorial characterization of the distributed 1-solvable tasks. *J. Algorithms*, 11(3):420–440, 1990. doi:10.1016/0196-6774(90)90020-F.
10. Elizabeth Borowsky and Eli Gafni. Generalized FLP impossibility result for t-resilient asynchronous computations. In *25 ACM Symposium on Theory of Computing (STOC)*, pages 91–100, 1993. doi:10.1145/167088.167119.
11. Elizabeth Borowsky and Eli Gafni. A simple algorithmically reasoned characterization of wait-free computations. In *16th ACM Symposium on Principles of Distributed Computing (PODC)*, pages 189–198, 1997. doi:10.1145/259380.259439.
12. Elizabeth Borowsky, Eli Gafni, Nancy A. Lynch, and Sergio Rajsbaum. The BG distributed simulation algorithm. *Distributed Comput.*, 14(3):127–146, 2001. doi:10.1007/PL00008933.
13. Sebastian Brandt. An automatic speedup theorem for distributed problems. In *38th ACM Symposium on Principles of Distributed Computing (PODC)*, pages 379–388, 2019. doi:10.1145/3293611.3331611.
14. Kayman Brusse and Faith Ellen. Reductions and extension-based proofs. In *PODC '21: ACM Symposium on Principles of Distributed Computing*, pages 497–507. ACM, 2021. doi:10.1145/3465084.3467906.
15. Armando Castañeda, Damien Imbs, Sergio Rajsbaum, and Michel Raynal. Generalized symmetry breaking tasks and nondeterminism in concurrent objects. *SIAM J. Comput.*, 45(2):379–414, 2016. doi:10.1137/130936828.
16. Armando Castañeda, Sergio Rajsbaum, and Matthieu Roy. Two convergence problems for robots on graphs. In *2016 Seventh Latin-American Symposium on Dependable Computing, LADC*, pages 81–90. IEEE Computer Society, 2016. doi:10.1109/LADC.2016.21.
17. Soma Chaudhuri. More choices allow more faults: Set consensus problems in totally asynchronous systems. *Inf. Comput.*, 105(1):132–158, 1993.

18 Faith E. Fich and Eric Ruppert. Hundreds of impossibility results for distributed computing. *Distributed Comput.*, 16(2-3):121–163, 2003. doi:10.1007/s00446-003-0091-y.

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

20 Pierre Fraigniaud, Ami Paz, and Sergio Rajsbaum. A speedup theorem for asynchronous computation with applications to consensus and approximate agreement. In *PODC*, pages 460–470. ACM, 2022.

21 Pierre Fraigniaud, Sergio Rajsbaum, and Corentin Travers. Locality and checkability in wait-free computing. *Distributed Comput.*, 26(4):223–242, 2013. doi:10.1007/s00446-013-0188-x.

22 Eli Gafni and Sergio Rajsbaum. Distributed programming with tasks. In *14th International Conference on Principles of Distributed Systems (OPODIS)*, LNCS 6490, pages 205–218. Springer, 2010. doi:10.1007/978-3-642-17653-1_17.

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

24 Maurice Herlihy and Sergio Rajsbaum. The decidability of distributed decision tasks. In *Proceedings of the Twenty-Ninth Annual ACM Symposium on the Theory of Computing*, pages 589–598. ACM, 1997. doi:10.1145/258533.258652.

25 Maurice Herlihy and Sergio Rajsbaum. A classification of wait-free loop agreement tasks. *Theor. Comput. Sci.*, 291(1):55–77, 2003. doi:10.1016/S0304-3975(01)00396-6.

26 Maurice Herlihy and Sergio Rajsbaum. The topology of shared-memory adversaries. In *PODC*, pages 105–113. ACM, 2010.

27 Maurice Herlihy and Sergio Rajsbaum. Simulations and reductions for colorless tasks. In *ACM Symposium on Principles of Distributed Computing, PODC*, pages 253–260. ACM, 2012. doi:10.1145/2332432.2332483.

28 Maurice Herlihy, Sergio Rajsbaum, Michel Raynal, and Julien Stainer. From wait-free to arbitrary concurrent solo executions in colorless distributed computing. *Theor. Comput. Sci.*, 683:1–21, 2017.

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

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

31 Idit Keidar and Sergio Rajsbaum. A simple proof of the uniform consensus synchronous lower bound. *Inf. Process. Lett.*, 85(1):47–52, 2003. doi:10.1016/S0020-0190(02)00333-2.

32 Petr Kuznetsov, Thibault Rieutord, and Yuan He. An asynchronous computability theorem for fair adversaries. In *PODC*, pages 387–396. ACM, 2018.

33 Nathan Linial. Locality in distributed graph algorithms. *SIAM J. Comput.*, 21(1):193–201, 1992.

34 Nancy A. Lynch. A hundred impossibility proofs for distributed computing. In *8th ACM Symposium on Principles of Distributed Computing (PODC)*, pages 1–28, 1989. doi:10.1145/72981.72982.

35 Nancy A. Lynch and Sergio Rajsbaum. On the borowsky-gafni simulation algorithm. In *ISTCS*, pages 4–15. IEEE Computer Society, 1996.

36 Moni Naor and Larry J. Stockmeyer. What can be computed locally? *SIAM J. Comput.*, 24(6):1259–1277, 1995. doi:10.1137/S0097539793254571.

37 David Peleg. *Distributed Computing: A Locality-Sensitive Approach*. SIAM, 2000.

38 Sergio Rajsbaum. Iterated shared memory models. In *LATIN*, volume 6034 of *Lecture Notes in Computer Science*, pages 407–416. Springer, 2010.

39 Joseph Rotman. Covering complexes with applications to algebra. *Rocky Mountain Journal of Mathematics*, 3(4):641–674, 1973. doi:10.1216/RMJ-1973-3-4-641.

40 Michael E. Saks and Fotios Zaharoglou. Wait-free k-set agreement is impossible: the topology of public knowledge. In *25th ACM Symposium on Theory of Computing (STOC)*, pages 101–110, 1993. doi:10.1145/167088.167122.

41 Hans van Ditmarsch, Éric Goubault, Marijana Lazic, Jérémy Ledent, and Sergio Rajsbaum. A dynamic epistemic logic analysis of equality negation and other epistemic covering tasks. *J. Log. Algebraic Methods Program.*, 121:100662, 2021. doi:10.1016/j.jlamp.2021.100662.

A Omitted Proofs

Proof of Theorem 10. To establish the theorem, we first prove two auxiliary claims, with the same notations as in the statement of the theorem.

▷ Claim 20. Every two vertices $u \neq w$ of the same connected component of $\Delta(\sigma)$ satisfy $\{u, w\} \in \Delta^{(\ell-1)}(\sigma)$.

Proof of claim. Let u, v, w be three vertices of the same connected component $\mathcal{K}$, such that $\{u, v\} \in \mathcal{K}, \{v, w\} \in \mathcal{K}$, but $\{u, w\} \notin \mathcal{K}$, and let us show that $\{u, w\} \in \Delta^{(1)}(\sigma)$. (If there are no such three vertices then we are done.) For this, it is sufficient to define a simplicial map

$$f : \mathsf{Bary}(\{u, w\}) \to \Delta(\sigma)$$

which agrees with $\Delta_{\{u,w\},\sigma}$. We set

$$f(\{u\}) = u,\ f(\{w\}) = w, \text{ and } f(\{u, w\}) = v.$$

In this way, any edge of $\mathsf{Bary}(\{u, w\})$ is mapped to either $\{u, v\}$ or $\{v, w\}$, which both belong to $\Delta(\sigma)$. It follows that f is simplicial, and agrees with $\Delta_{\{u,w\},\sigma}$. Therefore, for any two vertices u, w of $\mathcal{K}$ at distance at most 2 in the graph $\mathsf{Skel}_1(\mathcal{K})$, $\{u, w\} \in \Delta^{(1)}(\sigma)$. By the same argument, any two vertices u, w of $\mathcal{K}$ at distance at most 4 in the graph $\mathsf{Skel}_1(\mathcal{K})$ satisfy $\{u, w\} \in \Delta^{(2)}(\sigma)$, and more generally, for any two vertices u, w of $\mathcal{K}$ at distance at most 2^r in the graph $\mathsf{Skel}_1(\mathcal{K})$, $\{u, w\} \in \Delta^{(r)}(\sigma)$. As a consequence, for every two vertices u, w of $\mathcal{K}$, $\{u, w\} \in \Delta^{(\ell-1)}(\sigma)$. ◁

We next show that a similar claim holds for any set of vertices in a connected component of $\Delta(\sigma)$, and not only for pairs.

▷ Claim 21. Every set $\tau \subseteq V(\Delta(\sigma))$ of vertices of the same connected component of $\Delta(\sigma)$ with $2 < |\tau| \leq |\sigma|$ satisfies $\tau \in \Delta^{(\ell)}(\sigma)$.

Proof of claim. It is sufficient to show that the local task $\Pi_{\tau,\sigma}^{(\ell-1)} = (\tau, \Delta^{(\ell-1)}(\sigma), \Delta_{\tau,\sigma}^{(\ell-1)})$ is solvable in one round, which we do by defining a simplicial map

$$f : \mathsf{Bary}(\tau) \to \Delta^{(\ell-1)}(\sigma)$$

that agrees with $\Delta_{\tau,\sigma}^{(\ell-1)}$, as follows. Let $\tau = \{v_1, \ldots, v_d\}$ where the vertices of τ are indexed in an arbitrary order, where $d = |\tau|$. For every singleton set $\{v_i\} \subseteq \tau$ we set

$$f(\{v_i\}) = v_i$$

while for every set $S \subseteq \tau$ of cardinality $|S| > 1$, we set

$$f(S) = v_1.$$

Let τ' be a face of τ. The views encoded by different vertices in a simplex $\rho \in \mathsf{Bary}(\tau')$ of a barycentric subdivision are totally ordered by inclusion, so ρ may contain at most one vertex that corresponds to singleton view (i.e., a view composed of a single vertex of τ'). As a consequence, there are only three possible cases:

- $f(\rho) = v_i$ whenever $\rho = \{v_i\}$, or
- $f(\rho) = v_1$ whenever ρ contains no singleton sets but possibly $\{v_1\}$, or
- $f(\rho) = \{v_1, v_i\}$ for some $i \in \{2, \ldots, d\}$ whenever ρ contains the singleton $\{v_i\}$ plus other non-singleton vertices.

In all three cases, the image of ρ is a simplex of $\Delta^{(\ell-1)}_{\tau,\sigma}(\tau')$, by Claim 20. Therefore f is simplicial, and agrees with $\Delta^{(\ell-1)}_{\tau,\sigma}$, which implies that $\tau \in \Delta^{(\ell)}(\sigma)$. ◁

We are now ready to prove Theorem 10. For the first part of the theorem, let $\mathcal{K}$ be a connected component of $\Delta(\sigma)$, and our goal is to show that every $\tau \subseteq V(\mathcal{K})$ with $1 \leq |\tau| \leq \dim(\sigma) + 1$ satisfies $\tau \in \Delta^{(\ell)}(\sigma)$. The claim holds for $|\tau| = 1$ (i.e., for vertices) as every vertex of $\mathcal{K}$ is by definition a vertex of $\Delta(\sigma)$. It holds for $|\tau| = 2$ (i.e., for edges) by Claim 20, and for $|\tau| > 2$ by Claim 21.

For establishing the second item, first note that one direction of the if and only if statement is trivial, as the closure operator can only add simplices, so connected components of $\mathsf{Skel}_1(\Delta^{(k-1)}(\sigma))$ can only merge when moving to $\mathsf{Skel}_1(\Delta^{(k)}(\sigma))$ and not break. For the other direction, we proceed by induction on $k \geq 0$. The statement is trivial for $k = 0$.

Let us assume that the statement holds for k and show that it holds for $k+1$. Let u and v be two vertices of $\Delta(\sigma)$ in two different connected components of $\Delta(\sigma)$, and by assumption in two different connected components of $\Delta^{(k)}(\sigma)$. Let us show that $\{u, v\} \notin \Delta^{(k+1)}(\sigma)$. For the purpose of contradiction, let us consider a map

$$f : \mathsf{Bary}(\{u, v\}) \to \Delta^{(k)}(\sigma)$$

that agrees with $\Delta^{(k)}_{\{u,v\},\sigma}$. We must have $f(\{u\}) = u$, $f(\{v\}) = v$, and $f(\{u, v\}) = w$ for some vertex $w \in V(\Delta^{(k)}(\sigma))$. However, u and v are in two different connected components of $\Delta^{(k)}(\sigma)$, which implies that $\{u, w\}$ or $\{v, w\}$ is not an edge of $\Delta^{(k)}(\sigma)$. As a consequence, f is not simplicial, and thus $\{u, v\} \notin \Delta^{(k+1)}(\sigma)$. In other words, no edges can be added between different connected components of $\Delta(\sigma)$ during successive closure operations. ◀

Proof of Theorem 11. Let $\Pi = (\mathcal{I}, \mathcal{O}, \Delta)$ such that $\mathsf{Cl}(\Pi)$ is solvable in $t-1$ rounds by n processes. Let

$$\delta' : \mathsf{Bary}^{(t-1)}(\mathsf{Skel}_n(\mathcal{I})) \to \mathcal{O}$$

be a simplicial map agreeing with Δ. Roughly, an algorithm for solving Π consists of two phases: first solving $\mathsf{Cl}(\Pi)$ using δ', and, second, given the output of δ', reconciliating these outputs (valid for $\mathsf{Cl}(\Pi)$, but not necessarily for Π) using the algorithm solving the local task for these outputs (see Fig. 3). More formally, les us consider a process p with input x, i.e., $x \in \mathcal{I}$ is a vertex. After $t-1$ rounds, this process gets a view w which is a vertex of $\mathsf{Bary}^{(t-1)}(\mathcal{I})$. Then, p proceeds with one more round of communication, and gets a view in $\mathsf{Bary}^{(t-1)}(\mathcal{I})$, which is either of the form $\{w\}$ or of the form $\{w, w'\}$ where $\{w, w'\}$ is an edge of $\mathsf{Bary}^{(t-1)}(\mathcal{I})$. Note that the property of the Barycentric subdivision guarantee that in the latter case w' must contain an input $x' \neq x$ of another process, where $e = \{x, x'\}$ was the actual input. Let $y = \delta'(w)$ and $y' = \delta'(w')$. Moreover, let $e' = \{y, y'\}$. Note that e' is an edge of $\Delta'(e)$ as δ' solves $\mathsf{Cl}(\Pi)$, but it is not necessarily an edge of $\Delta(e)$. The algorithm solving Π is as follows:

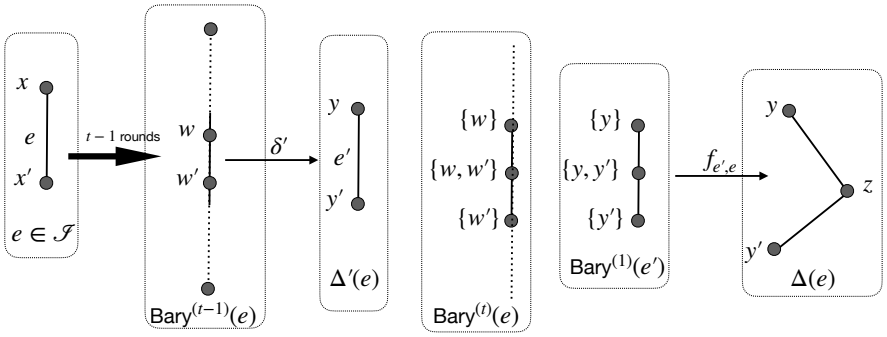

Figure 3 Proof of Theorem 11.

- If the view of p after t rounds is $\{w\}$, then p outputs y;
- If the view of p after t rounds is $\{w, w'\}$, then p outputs $z = f_{e',e}(\{y, y'\})$ where

$$f_{e',e} : \mathsf{Bary}^{(1)}(e') \to \Delta(e)$$

is a simplicial map[2] solving the local task $\Pi_{e',e}$.

This algorithm is well defined, as if p has view $\{w, w'\}$, then it knows x and x', and it can compute y and y' from the views w and w'. To show correctness, let $\sigma \in \mathcal{I}$, and let us show that our algorithm produces a simplex $\tau \in \Delta(\sigma)$. If σ is a vertex x, then all processes output $y \in \Delta'(x)$, which is a vertex of $\Delta(x)$, as desired. If σ is an edge $e = \{x, x'\}$, then let $\{w, w'\} \in \mathsf{Bary}^{(t-1)}(e)$ be the edge of the barycentric subdivision corresponding the the current configuration after $t-1$ rounds. Assume, w.l.o.g., that, during the t-th round, some processes (maybe none) get view $\{w\}$ while some other processes (at least one) get view $\{w, w'\}$ – the latter view is a vertex of $\mathsf{Bary}^{(t)}(e)$, whereas it was an edge of $\mathsf{Bary}^{(t-1)}(e)$. Note that starting from $\{w, w'\} \in \mathsf{Bary}^{(t-1)}(e)$, it is not possible that some processes reads a view $\{w\}$ in $\mathsf{Bary}^{(t)}(e)$ while some other processes reads $\{w'\}$ in $\mathsf{Bary}^{(t)}(e)$. It follows that a group of processes may output $y = \delta'(x)$ while another group of processes may output $z = f_{e',e}(\{y, y'\})$. The crucial property is that $f_{e',e}$ fixes vertices, that is, $f_{e',e}(\{y\}) = y$. Since $f_{e',e}$ is simplicial and agrees with $\Delta_{e',e}$, we get that

$$\{y, z\} = \{f_{e',e}(\{y\}), f_{e',e}(\{y, y'\})\} \in \Delta_{e',e}(e') = \Delta(e),$$

as desired. This completes the proof of the theorem. ◀

Proof of Theorem 13. Fix a colorless task Π that has a round-reduction impossibility proof. By Lemma 8, Π^* is not 0-round solvable. For each simplex $\sigma \in \mathcal{O}$, all the connected components of $\Delta^*(\sigma)$ are complete up to dimension $\dim(\sigma)$ by Claim 21: if some simplices of dimension at most $\dim(\sigma)$ are missing in $\Delta^*(\sigma)$ then at least one of them would have been added to it when applying the closure operator, contradicting the fact that Π^* is a fixed-point.

To construct an FLP-style proof, we consider an algorithm A that claims to solve Π, and define $\delta : V(\mathcal{I}) \to V(\mathcal{O})$ to map each input value $x \in \mathcal{I}$ to the output value $\delta(x)$ produced by A in the execution where only the value x appears; $\delta(x)$ is unique since the execution

[2] There might be more than one simplicial map from $\mathsf{Bary}^{(1)}(e')$ to $\Delta(e)$, in which case one selects one of them arbitrarily for defining the algorithm solving Π.

is unique and the algorithm is deterministic. The fact that $\mathsf{Cl}^*(\Pi)$ is not 0-round solvable means that there is a simplex $\sigma \in \mathcal{I}$ such that $\delta(\sigma) = \{\delta(x) \mid x \in \sigma\} \notin \Delta^*(\sigma)$. As the connected components are complete up to dimension $\dim(\sigma)$, the simplex σ is mapped to (at least) two different connected components, i.e. there are two input values $x, x' \in \sigma$ and two connected components C, C' of $\Delta(\sigma)$ such that $\delta(x) \in C$ and $\delta(x') \in C'$.

Let $\sigma_0 = \{x, x'\}$. As $\sigma_0 \subseteq \sigma$, the fact that Δ^* is a carrier map (Lemma 5) implies that the connected components of $\Delta(\sigma_0)$ are a refinement of the connected components of $\Delta(\sigma)$. Hence, there is a connected component $C_0 \subseteq C$ of $\Delta(\sigma_0)$ such that $\mathsf{val}(x) \cap C_0 \neq \emptyset$, and similarly a different connected component $C'_0 \subseteq C'$ of $\Delta(\sigma_0)$ such that $\mathsf{val}(x') \cap C'_0 \neq \emptyset$. Note that Theorem 10 asserts that the connected components of $\Delta(\sigma_0)$ and of $\Delta^*(\sigma_0)$ are the same.

Let us say that a configuration reachable from σ_0 is *bivalent* (w.r.t. σ_0) if a valency query on it returns output values in at least two different connected components of $\Delta(\sigma_0)$. Note that σ_0 is bivalent by construction, and that if the algorithm is in a bivalent configuration it cannot decide without taking further steps.

We construct an infinite sequence $\sigma_0, \sigma_1, \ldots$ of bivalent configurations. Each σ_t will consist only of two views

$$\sigma_t = \{w_t, w'_t\}.$$

Assume $\sigma_0, \ldots, \sigma_t$ are chosen and $\mathsf{val}(\sigma)$ is known for each $\sigma \in \mathsf{Bary}_1(\sigma_t)$. Recall that $\mathsf{Bary}_1(\sigma_t)$ is composed of the vertices $\{w_t\}, \{w_t, w'_t\}, \{w'_t\}$ and the edges $e = (\{w_t\}, \{w_t, w'_t\})$ and $e' = (\{w'_t\}, \{w_t, w'_t\})$. Let C_t be a connected component of $\Delta(\sigma_0)$ such that $\mathsf{val}(\{w_t, w'_t\}) \cap C_t \neq \emptyset$.

The fact that σ_t is bivalent implies that there is a connected component C'_t of $\Delta(\sigma_0)$, $C'_t \neq C_t$, such that $\mathsf{val}(\sigma_t) \cap C'_t \neq \emptyset$. As the valencies of the vertices are contained in the valencies of the edges, we have $\mathsf{val}(e) \cup \mathsf{val}(e') = \mathsf{val}(\sigma_t)$. Hence, at least one edge $f \in \{e, e'\}$ has $\mathsf{val}(f) \cap C'_t \neq \emptyset$. Since $\{w_t, w'_t\} \in f$, it also has $\mathsf{val}(f) \cap C_t \neq \emptyset$. The edge f is thus bivalent, and we set it as σ_{t+1}. This process can continue for every $t \geq 0$, and the proof is complete. ◀

Proof of Proposition 17. The diameter D of $\mathcal{O}$ is $N = 1/\epsilon$. Let us first show that if $0 \leq k < \lceil \log_2 1/\epsilon \rceil$, we have $\{0,1\} \notin \Delta^{(k)}(\{0,1\})$. The proof is based on the following fact: for any two distinct vertices u and v of $\Delta^{(k)}(\{0,1\})$ at distance at least 3 in $\mathsf{Skel}_1(\Delta^{(k)}(\{0,1\}))$, we have $\{u,v\} \notin \Delta^{(k+1)}(\{0,1\})$. To establish this fact, let us consider any map

$$f : \mathsf{Bary}^{(1)}(\{u,v\}) \to \Delta^{(k)}(\{0,1\})$$

agreeing with $\Delta^{(k)}_{\{u,v\},\{0,1\}}$. Since f agrees with $\Delta^{(k)}_{\{u,v\},\{0,1\}}$, we must have $f(u) = u$ and $f(v) = v$. Therefore, if $w = f(\{u,v\})$ then either $\{u,w\}$ or $\{w,v\}$ is not an edge of $\Delta^{(k)}(\{0,1\})$ because u and v are at distance greater than 2. Therefore f is not simplicial, which shows that $\{u,v\} \notin \Delta^{(k+1)}(\{0,1\})$, as claimed. It follows that, for $k < \lceil \log_2 1/\epsilon \rceil$, we have $\{0,1\} \notin \Delta^{(k)}(\{0,1\})$.

This latter fact implies that the k-th closure of ϵ-agreement is not solvable in zero rounds: an algorithm f solving this k-th closure must satisfy $f(0) = 0$ and $f(1) = 1$. As a consequence, if some processes start with 0, and some other processes start with input 1, all these processes jointly output the set $\{0,1\}$, which is not a valid output as $\{0,1\} \notin \Delta^{(k)}(\{0,1\})$. ◀

Proof of Theorem 19. Consider a non-trivial covering complex $(\mathcal{O}, f)$ of a complex $\mathcal{I}$, and the corresponding covering task $\Pi = (\mathcal{I}, \mathcal{O}, \Delta)$. For the case of the Hexagon task we have seen that $\mathsf{Cl}(\mathrm{HX}) = \mathrm{HX}$, and here we start the same why.

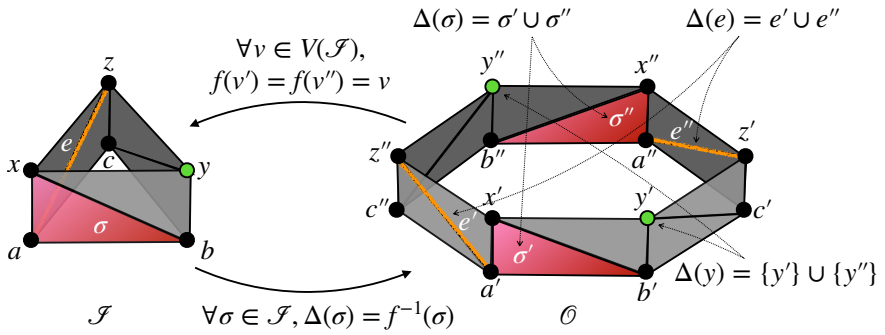

Figure 4 A 2-dimensional covering task extending the Hexagone task to a higher dimension. Here, $f(\sigma') = f(\sigma'') = \sigma$, and accordingly the image of σ under Δ is the union of the two complexes with unique facets σ' and σ''.

Consider an input simplex $\sigma \in \mathcal{I}$, and note that each of its sheets is a simplex, and that its sheets do not intersect. Hence, all the connected components of $\Delta(\sigma)$ are complete (each is of dimension $\dim(\sigma)$), and Theorem 10 implies that $\Delta^*(\sigma) = \Delta(\sigma)$, so $\mathsf{Cl}(\Pi) = \Pi$. Hence, if Π is solvable wait-free by $n \geq 2$ processes in the IIS model, then by Lemma 8 it is wait-free solvable in zero rounds.

Consider a possible zero-round algorithm for Π, and its decision map δ. Recall that δ must be simplicial, i.e. maps simplices to simplices, and hence also paths to paths.

Fix $x \in V(\mathcal{I})$, its image $y = \delta(x) \in V(\mathcal{O})$ under δ, and note that $f(y) = x$. Since $(\mathcal{O}, f)$ is non-trivial, there is another vertex $y' \in V(\mathcal{O})$, $y' \neq y$, such that $f(y') = x$. As $\mathcal{O}$ is connected, it contains a path $(y_0 = y, y_1, \ldots, y_k = y')$ connecting y and y', and $\mathcal{I}$ contains its image $C = (x_0 = f(y_0), \ldots, x_k = f(y_k))$ under f. Note that $x = x_0 = x_k$, so C is in fact a cycle in $\mathcal{I}$. Apply δ to C, and the fact that δ is simplicial gives a cycle $\delta(C)$ in $\mathcal{O}$.

As $\delta(x_0) = y_0$ but $\delta(x_k) \neq y_k$, there must exist a minimal index $0 \leq i < k$ such that $\delta(x_i) = y_i$ and $\delta(x_{i+1}) \neq y_{i+1}$. By the construction of the path, $f(y_i, y_{i+1}) = (x_i, x_{i+1})$, and by the assumption that δ solves the task and hence comply with Δ we have $f(y_i, \delta(x_{i+1})) = (x_i, x_{i+1})$. Hence $f^{-1}(x_i, x_{i+1})$ contains both (y_i, y_{i+1}) and $(y_i, \delta(x_{i+1}))$, i.e. (x_i, x_{i+1}) has two intersecting sheets, in contradiction to $(\mathcal{O}, f)$ being a covering complex. ◀