Wait-free Solvability of Equality Negation Tasks

Éric Goubault
École Polytechnique, Palaiseau, France
eric.goubault@lix.polytechnique.fr

Marijana Lazić
TU München, Munich, Germany
lazic@in.tum.de

Jérémy Ledent
École Polytechnique, Palaiseau, France
jeremy.ledent@lix.polytechnique.fr

Sergio Rajsbaum
Instituto de Matemáticas, UNAM, Mexico City, Mexico
sergio.rajsbaum@gmail.com

Abstract

We introduce a family of tasks for n processes, as a generalization of the two process *equality negation* task of Lo and Hadzilacos (SICOMP 2000). Each process starts the computation with a private input value taken from a finite set of possible inputs. After communicating with the other processes using immediate snapshots, the process must decide on a binary output value, 0 or 1. The specification of the task is the following: in an execution, if the set of input values is large enough, the processes should agree on the same output; if the set of inputs is small enough, the processes should disagree; and in-between these two cases, any output is allowed. Formally, this specification depends on two threshold parameters k and ℓ, with $k < \ell$, indicating when the cardinality of the set of inputs becomes "small" or "large", respectively. We study the solvability of this task depending on those two parameters. First, we show that the task is solvable whenever $k + 2 \leq \ell$. For the remaining cases ($\ell = k + 1$), we use various combinatorial topology techniques to obtain two impossibility results: the task is unsolvable if either $k \leq n/2$ or $n - k$ is odd. The remaining cases are still open.

2012 ACM Subject Classification Theory of computation → Concurrency

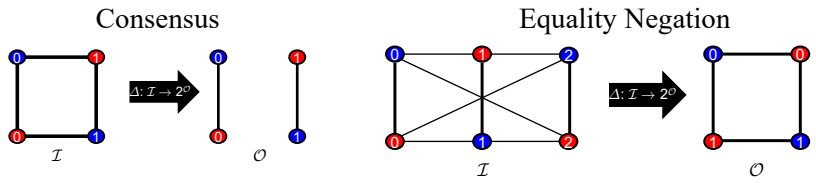
Keywords and phrases Equality negation, distributed computability, combinatorial topology

Acknowledgements The authors were supported by DGA project "Validation of Autonomous Drones and Swarms of Drones" and the academic chair "Complex Systems Engineering" of Ecole Polytechnique-ENSTA-Télécom-Thalès-Dassault-Naval Group-DGA-FX-FDO-Fondation ParisTech, by the UNAM-PAPIIT project IN109917, by the France-Mexico Binational SEP-CONACYT-ANUIES-ECOS grant M12M01, by the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme under grant agreement No 787367 (PaVeS), as well as by the Austrian Science Fund (FWF) through Doctoral College LogiCS (W1255-N23).

1 Introduction

The *equality negation* task is a variant of consensus that was defined by Lo and Hadzilacos [14], as the central idea to prove that the consensus hierarchy [10, 13] is not robust. Consider two processes P_0 and P_1, each of which has a private initial value, drawn from the set of possible input values $I = \{0, 1, 2\}$. Each process must irrevocably decide an output value either 0 or 1 so that the decisions of the processes are the same if and only if the initial values of the processes are different. It is well known that there is no wait-free consensus algorithm for two processes that uses only registers [6, 15]. The same is true for equality negation, because, as explained in [14], it is possible to solve consensus for two processes using an equality negation algorithm.

This result is intriguing for the following reason. It is well known that consensus is intimately related to connectedness. Namely, the reason why consensus is not wait-free solvable with registers is because its input complex is connected, while its specification requires deciding unto disconnected components of the output complex. The equality negation task is unsolvable for a different reason, since its output complex is connected. In our recent project [8], we have studied the unsolvability of this task, in the case of two processes, using methods from the field of episemic logic. The goal of this paper is to understand this task from a topological perspective. To do so, we extend it to the setting of more than two processes, and thus introduce a class of tasks which seem to be of a nature not previously studied.



Consider the following family of tasks, as a generalization of the equality negation problem, for n processes. These tasks are parametrized by two integers k, ℓ, with $1 \leq k < \ell \leq n$. In a given initial global state g of the system, each process has a private input value, and let $\mathit{InputSet}(g) \subseteq I$ be the set of these inputs, where I is the set of all possible input values. Thus, in one extreme, all processes may start with the same input value, and $|\mathit{InputSet}(g)| = 1$, and in the other they all start with different input values, and $|\mathit{InputSet}(g)| = n$. Similarly, let $\mathit{OutputSet}(g') \subseteq \{0, 1\}$ be the set of decided values in some final global state g' of the computation. For every $k, \ell \in \mathbb{N}$ such that $1 \leq k < \ell \leq n$ we define an equality negation task as follows, where g is any initial state, and g' is a final state of any execution starting with g:

- If $k < |\mathit{InputSet}(g)| < \ell$, the processes can decide any value in $\{0, 1\}$.
- If $|\mathit{InputSet}(g)| \geq \ell$, then $|\mathit{OutputSet}(g')| = 1$
- If $|\mathit{InputSet}(g)| \leq k$, then $|\mathit{OutputSet}(g')| > 1$

In this paper, we study the wait-free solvability of these tasks using read/write registers, depending on the two parameters k and ℓ. First, we show that if $\ell - k \geq 2$, the task is solvable. The remaining cases are the ones where $\ell = k+1$ and $1 \leq k \leq n-1$. In those cases, it is not possible anymore to have $k < |\mathit{InputSet}(g)| < \ell$: this means that there is no "gap" where the processes are free to decide any output without restriction. In order to prove that these remaining cases are unsolvable, we use various combinatorial topology techniques [11]. In particular, we show two impossibility results: the task is unsolvable if $k \leq n/2$ or if $n - k$ is odd. This leaves a few cases which are still open, namely, when $k > n/2$ and $n - k$ is even. We conjecture that they should also be unsolvable, but the right topological argument to prove it remains to be found.

Notice that the case of $n = 2$ processes is special, because the only option is that $k = 1$ and $\ell = 2$, in which case we obtain the equality negation task of Lo and Hadzilacos. Here the task is (trivially) solvable for $I = \{0, 1\}$, but becomes unsolvable if we add one more input value $I = \{0, 1, 2\}$. We discuss at the end of the paper why this case behaves differently.

Related work. The class of tasks that can be defined only in terms of sets of input values and corresponding sets of output values (without specifying which process gets which input nor which process should produce which output) are called *colorless tasks*. Since their introduction in [2], they have been thoroughly investigated; an overview and detailed

citations can be found in [11]. There is a very elegant characterization of wait-free colorless tasks solvability, in terms of the existence of a certain simplicial map between the complex of possible input values and the complex of possible output values. The central colorless task is *set-agreement*, where each of n processes starts with an input, and each must halt with some process' input, such that no more than $n-1$ distinct inputs are chosen. The proof that set agreement is unsolvable using registers [12] revealed the deep connection between distributed computing and topology. Other examples of colorless tasks are consensus, loop agreement and approximate agreement.

Some tasks of interest are not colorless, and they are often much more difficult to analyze. For instance, the *renaming* task requires n processes, each starting with a unique input name, to choose distinct output from a range of size $2n-2$. This task can be solved if and only if n is not a prime power. The proof uses more involved algebraic topology techniques [3, 5] than the basic impossibility proof of set agreement based on Sperner's lemma.

More generally, colorless tasks seem to be about agreement, while others seem to be about symmetry breaking. Many such tasks have been considered, where processes only have their own (distinct) names as inputs. In *weak symmetry breaking*, the processes must choose binary outputs so that if all n processes participate, at least one chooses 0 and one chooses 1. In an *election* task, one process outputs 0 and exactly $n-1$ processes output 1. These and others are included in the *generalized symmetry breaking* family [4], defined by a set of possible output values, and for each value v, a lower bound and an upper bound on the number of processes that have to decide this value. These tasks are generally not only more complicated to analyze, but weaker than agreement tasks [4].

Equality negation tasks look very much like colorless tasks, in the sense that they are specified only according to the sets of input values of the processes, with no regard for which process has which value. But in fact, equality negation tasks are not colorless according to the definition in Section 4.1.4 of [11]. Indeed, the process names play an important role in the algorithms that we present to solve some instances of equality negation in Section 3.

2 Preliminaries

2.1 Equality negation tasks: from 2 to n processes

Note that the equality negation task defined in [14] for two processes does not have a unique generalization to the case with more than two processes.

In this paper we introduce a generalization that allows any number of input values, but we keep the set of possible decision values as $\{0,1\}$. It seems natural to define a generalization such that: (i) all the processes decide the same output value if they all have (pairwise) different input values, and (ii) they do not decide on the same output value if they all initially have the same input value. Still, this definition does not cover all possible combinations of input values of all processes, for example, if there are two processes with the same input, and a process with a different one. Therefore, there is a lot of freedom for defining output constraints for the remaining input cases. Our goal is to analyse all the possible generalizations of a certain form.

We define a family of equality negation tasks, based on the cardinality of the set of input values of all processes. Let n be the number of processes and let there exist exactly n input values, namely $I = \{0, 1, 2, \ldots, n-1\}$. This assumption is done w.l.o.g., as we explain at the end of the paper: all our results can be extended to $|I| > n$ in a straightforward manner. Every initial global state g defines the set of input values of all processes in the state g, that we denote by $\mathit{InputSet}(g)$. As this set is always a non-empty subset of the set I, its

cardinality ranges from 1 to n. Note that the case when all processes have the same input value is represented by $|InputSet(g)| = 1$, and the case when all processes have different input values is the one when $|InputSet(g)| = n$. Different constraints on the intermediate cases introduce a family of tasks.

For every $k, \ell \in \mathbb{N}$ with $1 \leq k < \ell \leq n$ we define a generalized equality negation task:

- If in the initial global state g we have $|InputSet(g)| \geq \ell$, then all processes must decide the same value, either 0 or 1.
- If initially $|InputSet(g)| \leq k$, processes do not decide on the same value, i.e., there is at least one process deciding 0, and at least one deciding 1.
- If initially $k < |InputSet(g)| < \ell$, then each process can decide independently any value from $\{0, 1\}$.

The original equality negation task for two processes of Lo and Hadzilacos [14] is recovered as a special case, where $n = 2$, $I = \{0, 1, 2\}$, $k = 1$ and $\ell = 2$. It was shown to be unsolvable in the wait-free immediate shapshot model, by reduction to the consensus task. A more thorough study of this task for two processes, using topological methods and epistemic logic methods, can be found in [8].

2.2 Distributed computing

We consider the usual model consisting of n asynchronous processes that communicate through read/write shared registers of unbounded size [12]. When solving a task, initially, each process has some input value. After communicating with each other a finite number of times, each process produces an output value. The outputs produced are related to the inputs in the execution, as defined by the task specification.

In this paper, we do not consider crashing processes. All the n processes are present in the beginning of an execution, and they are guaranteed to run until the end of their program. Instead of crashes, the two ingredients that allow us to obtain impossibility results are asynchrony and wait-freedom. Although a bit unusual, this assumption is not restrictive: if we want to transpose our results to a model where the processes can crash, we can simply say that our task does not impose any restrictions on the outputs whenever at least one process has crashed.

The processes are executing in an *asynchronous* way, meaning that an execution consists of an arbitrary interleaving of the write and read operations of all the processes. Moreover, the program run by the processes is required to be *wait-free*: every process must be guaranteed to terminate its program in a bounded number of steps. Intuitively, these two restrictions mean that a process is not allowed to wait until it receives information from any of the other processes, since it may be arbitrarily slow.

We could describe our results in the general read/write registers model, but it is easier to do it in the *immediate snapshot* model. And we do so without loss of generality, because from the task computability perspective, the two models are known to be equivalent [1]. In the immediate snapshot model, each process has a designated memory cell where it can atomically write its input value. After doing so, it can atomically take a snapshot of the whole shared memory, in order to read the values that have been written by the other processes. Moreover, we restrict to a subset of all the executions described above: the snapshot is guaranteed to happen immediately after the write. For each pair of processes P and Q participating in this protocol, the immediate-snapshot may result in three possible outcomes:

- either P was faster than Q, in which case P did not see the value of Q, but Q saw P;
- or Q was faster than P, in which case the situation is reversed;
- or they executed concurrently, in which case they both saw each-other's input value.

Formally, an immediate snapshot execution consists of a sequence of sets of processes, where each set is called a *concurrency class*. All the processes in the same concurrency class will execute concurrently, and see each other's values, as well as the values of the previous concurrency classes.

These immediate snapshot executions give rise to simplicial complexes with nicer topological properties: namely, its effect on the input complex is to subdivide each simplex, thus preserving the topology. Immediate-snapshot operations can be wait-free implemented from read/write register operations, although at a quadratic cost in terms of the number of operations. For a survey including such results see e.g. [16].

2.3 Combinatorial topology

For our impossibility results, we use the combinatorial topology representation and basic results described in [11], briefly recalled here.

A pair (P_i, x_i) consisting of a process P_i along with its private (input or output) value x_i is called a *vertex*. An *input vertex* $v = (P_i, x_i)$ represents the initial state of process P_i, while an *output vertex* represents its decision at the end of a computation. A set σ of such vertices of cardinal $k+1$ is called a *k-simplex* or just *simplex*, and denotes the input values or output values of k distinct processes in an execution. It is also used to denote a global state in an execution, in which case x_i is the local state of P_i. If the x_i are input values, then σ is called an *input simplex*, if they are output values, it is an *output simplex*. A set of simplices closed under containment is called *simplicial complex*.

The *dimension* of a simplex σ, denoted $\dim(\sigma)$, is $|\sigma| - 1$, and it is *full* if it contains n vertices, one for each process. In a simplicial complex, a subset of a simplex, which is a simplex as well, is called a *face*. A simplex is *maximal* if it is not a strict subset of another simplex. A simplicial complex is *pure of dimension* $n-1$ if all its maximal simplices are full. The simplices of lower dimension are used in some distributed computing models to represent executions where some processes have crashed; but in our case, we will be mostly interested in the full simplices. The set of all possible input (resp. output) simplices forms an *input complex* $\mathcal{I}$ (resp. *output complex* $\mathcal{O}$).

A *coloring* of a complex is a mapping that assigns an element of a certain domain, usually called a color, to each vertex of the complex. A complex and its coloring form a *chromatic complex*. In distributed computing, coloring often assigns a distinct process identity to each vertex v of a simplex. We say that a simplex is *properly colored* (or it has proper coloring) if every two of its vertices are differently colored. We call a simplex *monochromatic* if all its vertices are labeled with the same color.

A *task* T for n processes is a triple $(\mathcal{I}, \mathcal{O}, \Delta)$ where $\mathcal{I}$ and $\mathcal{O}$ are pure chromatic $(n-1)$-dimensional complexes, such that every simplex is properly colored. The set of colors is $\{0, \ldots, n-1\}$, and the colors are called *process names* or *ids*. The map Δ sends each simplex σ from $\mathcal{I}$ to a subcomplex $\Delta(\sigma)$ of $\mathcal{O}$. We say that Δ is a *carrier map* from the input complex $\mathcal{I}$ to the output complex $\mathcal{O}$. Briefly, the Asynchronous Computability Theorem [12] states that a task is wait-free solvable with registers if and only if there is a chromatic subdivision of the input complex and a chromatic simplicial map to the output complex, respecting the carrier map. Recall that a chromatic simplicial map sends simplices to simplices, preserving colors (process ids). We also use the fact that it can be assumed that a protocol solving a task using immediate snapshot operations induces an iterated chromatic subdivision of the input complex.

2.4 Index and Content of a Pseudomanifold

In this section, we introduce the main technical tool that we will use in Section 4.2.2 to prove impossibility results: the index lemma. The index lemma counts the properly colored n-simplices inside of a (not necessarily properly) colored and coherently oriented pseudomanifold, the *content*, by counting the properly colored $(n-1)$-simplices on the boundary, the *index*. The boundary determines the number of properly colored n-simplices in the interior of any pseudomanifold with that boundary.

Consider a set C with $n+1$ elements. A *sequence* S of C is an ordered list of the elements of C. We denote as S_i the *i-th* element of the sequence S, where $0 \leq i \leq n$. A *transposition* of S consists of interchanging the position of two elements S_i, S_j of S. If $i = j$ then we say that it is an *identity* transposition of S. A sequence S' of C is an *even transposition* of another sequence S if S' can be obtained with an even number of non identity transpositions over S. An *odd transposition* of S is defined similarly.

Let σ^n be a simplex with a proper coloring f with colors $ID^n = \{0, \ldots, n\}$. Consider a sequence S of ID^n. We say that S *induces* the sequence S' of σ^n with respect to f, when $f(S'_i) = S_i$, $0 \leq i \leq n$. If there is no ambiguity, we just say that S induces S'.

Let $\sigma^n = \{v_0, v_1, \ldots, v_n\}$ be a simplex. An *orientation* of σ^n is a set consisting of a sequence of its vertices and all even transpositions of this sequence. If $n > 0$ then there are exactly two possible orientations for σ^n: the sequence $\langle v_0, v_1, \ldots, v_n\rangle$ and all its even permutations, and the sequence $\langle v_1, v_0, v_2, \ldots, v_n\rangle$ and all its even permutations. For example, the two possible orientations of a 2-simplex can be seen as the clockwise and the counterclockwise directions, or the two possible orientations of a 1-simplex are the one from one of its vertices to the other, and the opposite direction. An orientation of σ^n, $n > 0$, *induces an orientation* on all of its $(n-1)$-faces: if σ_i^{n-1} is the $(n-1)$-face of σ^n without vertex v_i, then σ_i^{n-1} gets the same orientation if i is even, and otherwise it gets the opposite orientation. If σ^n, for example, is oriented $\langle v_0, v_1, \ldots, v_n\rangle$ then its face $\sigma_1^n = \langle v_0, v_2, v_3, \ldots, v_n\rangle$ gets orientation $\langle v_2, v_0, v_3, \ldots, v_n\rangle$, opposite of σ_1^n, as 1 is odd.

If σ^n has a proper coloring id with colors ID^n then we denote by $d = +1$ the orientation that contains the sequence induced by $\langle 0, 1, \ldots, n\rangle$, i.e., the sequence S of σ^n such that $id(S_i) = i$, $0 \leq i \leq n$, and denote by $d = -1$ the other orientation of σ^n which contains the sequence induced by $\langle 1, 0, \ldots, n\rangle$, the sequence S of σ^n such that $id(S_0) = 1$, $id(S_1) = 0$ and $id(S_i) = i$, $2 \leq i \leq n$. For $n = 0$, there is only one sequence of the vertices of a simplex σ^0, and then it has just one orientation, however we can associate $+1$ or -1 to this orientation. Hence, a 0-simplex also has two orientations. An orientation d of σ^n, $n > 0$, induces the orientation $(-1)^i d$ to σ_i^{n-1}, where σ_i^{n-1} is the $(n-1)$-face of σ^n without the vertex with *id* i.

A pseudomanifold K^n is *orientable* if there is an orientation for each of its n-simplices such that: if $\sigma, \sigma' \in K^n$ share a $(n-1)$-face τ then the two orientations on τ induced by σ and σ' are opposite. An orientation of K^n with this property is a *coherent orientation*. We say that K^n is *coherently oriented* if it has a coherent orientation. Let σ be an oriented n-simplex with a proper coloring c which uses colors ID^n. Consider a sequence S of ID^n. Let S' be the sequence of the vertices σ induced by S with respect to c. We say that S *belongs* to the orientation of σ with respect to c if S' belongs to the orientation of σ. The simplex σ is *counted by orientation* with respect to c in the following way: it is counted as $+1$ if the sequence $\langle 0, 1 \ldots n\rangle$ belongs to its orientation with respect to c, otherwise it is counted as -1. In the next definition, if $i \in ID^n$ is a color, we write $ID_i^n = ID^n \setminus \{i\}$.

▶ **Definition 1** (Index and Content). *Consider a coherently oriented pseudomanifold K^n with the induced orientation on its boundary $bd(K^n)$. Let c be a coloring, not necessarily proper, of K^n with ID^n.*

1. *The* content *of* K^n*,* $C(K^n)$*, with respect to* c *is the number of the properly colored* n*-simplices of* K^n *counted by orientation.*
2. *The* index *of* K^n*,* $I_i(K^n)$*, with respect to* c *is the number of properly colored* $(n-1)$*-simplices of* $bd(K^n)$ *with* ID_i^n *counted by orientation.*

If there is no ambiguity we simply write C^n or I_i^n. The next lemma is the restatement of Corollary 2 in [7] using our notation, and [9, pp. 46-47] for a simple version of dimension 2.

▶ **Lemma 2** (Index Lemma). *Let* K^n *be a coherently oriented pseudomanifold colored with* ID^n*. Then* $C^n = (-1)^i I_i^n$*.*

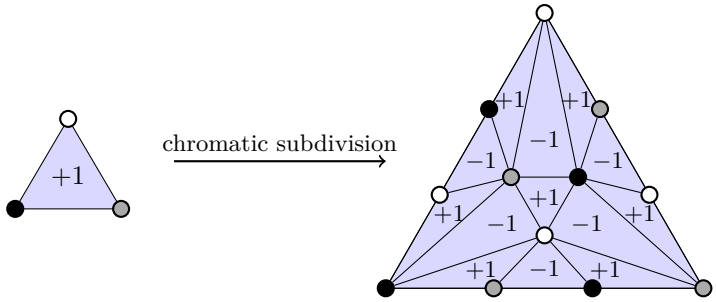
The coloring c of K^n is a simplicial map from $bd(K^n)$ to the boundary of a properly colored n-simplex σ^n with ID^n. Thus, we can think of the content of K^n as the number of times that $bd(K^n)$ is *wrapped around* $bd(\sigma^n)$, i.e., a combinatorial version of the notion of *degree* in topology.

To compute the index and content of chromatic subdivisions, we can just compute it from the original complex before subdivision happens. Indeed, we have the following Lemma:

▶ **Lemma 3.** *Let* σ^n *be a properly colored* n*-simplex with colors* ID^n*. Let* $\chi(\sigma^n)$ *be a chromatic subdivision of* σ^n*. Then* $C(\chi(\sigma^n)) = C(\sigma^n)$*.*

Proof. Assume w.l.o.g. that σ^n is counted positively. Thus, $C(\sigma^n) = 1$. We proceed by induction on n. For $n = 0$ (σ^0 is a single vertex), chromatic subdivisions do nothing so the result trivially holds. Now assume σ^n is of higher dimension.

By the index lemma, $C^n(\chi(\sigma^n)) = (-1)^n\, I_n^n(\chi(\sigma^n))$. Note that $ID_n^n = ID^{n-1}$. Let f be the $(n-1)$-face of σ^n with colors ID^{n-1}. Since $\chi(\sigma^n)$ is a chromatic subdivision of σ^n, the set K of properly colored $(n-1)$-simplexes on the boundary of $\chi(\sigma^n)$ with colors in ID^{n-1} is a chromatic subdivision of f. Thus, $I_n^n(\chi(\sigma^n)) = C^{n-1}(K) = C^{n-1}(f)$ by induction hypothesis, and $C^{n-1}(f) = I_n^n(\sigma^n) = (-1)^n\, C^n(\sigma^n)$. This concludes the proof. ◀



▶ **Corollary 4.** *Chromatic subdivisions preserve the index and content.*

3 Solvability analysis

If processes are not anonymous, we claim that the equality negation task, defined as above, is solvable if it holds that $\ell - k \geq 2$ (and $1 \leq k < \ell \leq n$).

▶ **Theorem 5.** *The algorithm from Figure 2 solves the equality-negation task if* $\ell - k \geq 2$*.*

Before we prove Theorem 5, and in order to understand the intuition behind it, let us first focus on its special case when $n \geq 3$, $k = 1$ and $\ell = n$, presented in Figure 1. In this figure, `immediateSnapshot(v)` returns the global state `V` of the system, `|V|` denotes the cardinal

```
1  Boolean v := input_value
2
3  P_0: V := immediateSnapshot(v);
4       return 0;
5
6  P_i: V := immediateSnapshot(v);
7       if |V| ≥ 2 and |val(V)| ≤ 1 then
8         return 1
9       else return 0;
```

Figure 1 Case $k = 1$ and $\ell = n$, for $n \geq 3$.

```
1  Boolean v := input_value
2
3  P_0: V := immediateSnapshot(v);
4       return 0;
5
6  P_i: V := immediateSnapshot(v);
7       if |V| ≥ n+k−ℓ+1 and |val(V)| ≤ k then
8         return 1
9       else return 0;
```

Figure 2 Arbitrary k and ℓ with $\ell - k \geq 2$.

of the set `V` and `val(V)` denotes the set of local values that appear in the global state `V`. In this algorithm, one fixed process decides 0 independently of its input and the result of the snapshot. All the other processes decide depending on the output of the snapshot protocol. If a process sees at least 2 processes, and their input values are the same, the process decides 1. Otherwise it decides 0.

▶ **Proposition 6.** *The algorithm from Figure 1 solves the equality negation task if* $n \geq 3$, $k = 1$ *and* $\ell = n$.

Proof. As $k = 1$ and $\ell = n$, we need to show that (i) when all processes have the same input value, they will disagree, and (ii) when all processes have different inputs, then they all decide the same value (in this case 0).

Case (i). The distinguished process P_0 will decide 0, and thus we need to ensure that at least one process will decide 1. Using the properties of immediate snapshot, we know that among the processes P_i, $i \neq 0$, there is at least one process P_j that sees at least $n - 1$ processes. As $k \geq 1$ and $n \geq \ell \geq k + 2$, we have that $n \geq 3$, and thus $n - 1 \geq 2$. Hence, the processes P_j sees $n - 1 \geq 2$ processes, and they must have the same input value by the initial assumption of the case (i). Thus, P_j decides 1, which is enough for the disagreement.

Case (ii). Next we discuss the second requirement, that is, when no two processes have the same input value. We prove that all processes in this case agree and decide 0. Suppose by contradiction that a process P_i decides 1. According to the algorithm, this can happen only if $i \neq 0$, and if P_i has seen 2 processes with the same input value. This contradicts the initial assumption of this case. Hence, all processes decide 0, which concludes the proof. ◀

Let us now return to the general case, for any n, k and ℓ with $\ell - k \geq 2$ and $1 \leq k < \ell \leq n$, given in Figure 2. Similarly as in Figure 1, we fix one process that decides 0 independently of its input and the result of the snapshot. All the other processes decide depending on the output of the snapshot protocol. If a process sees at least $n + k - \ell + 1$ processes, and among their input values there are at most k different values, the process decides 1, and otherwise it decides 0. Now we are ready to prove Theorem 5.

Proof of Theorem 5. We prove that the algorithm from Figure 2 indeed solves equality-negation task analogously as in the case with $k = 1$ and $\ell = n$. We need to prove that (i) if $|\mathit{InputSet}(g)| \leq k$ for an initial state g, then at least one process decides 0 and at least one decides 1, and (ii) if $|\mathit{InputSet}(g)| \geq \ell$, then all processes decide the same value, here 0.

In the case (i) again we have process P_0 that decides 0 and we need to show that there is at least one process that decides 1. Similarly as in the special case, we know there is a process P_j, $j \neq 0$ that sees at least $n - 1$ processes. As $\ell - k \geq 2$, we have that $n - 1 \geq n + k - \ell + 1$,

and thus P_j has seen at least $n+k-\ell+1$ processes and the set of their input values must have cardinality at most k by the assumption of the case (i). Hence, P_j decides 1.

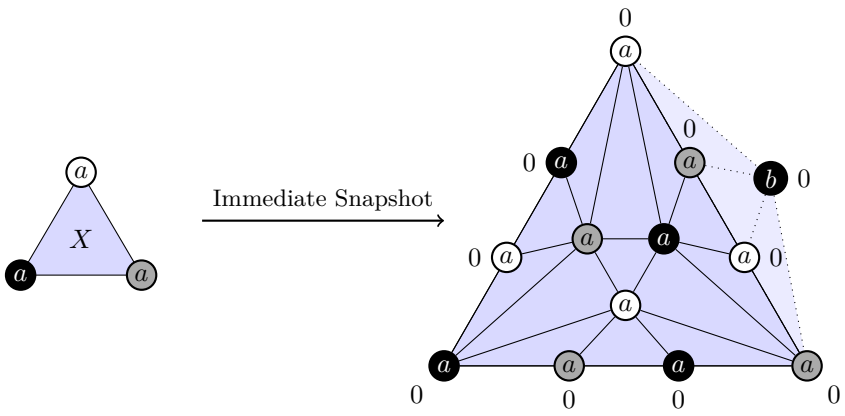
In the case (ii), initially there are at least ℓ different values. We want to prove that no process will decide 1. By means of contradiction, suppose that a process P_i decides 1. According to the algorithm, we have that $i \neq 0$ and P_i has seen at least $n+k-\ell+1$ processes with at most k different input values. Note that P_i did not see any process with the remaining values, which are at least $\ell-k$, and that there must be at least one process with each of those values. Thus, we have at least $n+k-\ell+1$ processes that P_i has seen and at least $\ell-k$ processes that P_i did not see, which is in total $n+1$. As the total number of processes is n, this is a contradiction. This concludes the proof. ◀

4 Unsolvable cases

In the previous section, we have proved that whenever $\ell-k \geq 2$, the corresponding equality negation task is solvable. The remaining cases are the ones where $\ell = k+1$, i.e., when there is no "gap" where the processes are allowed to decide any output value without constraints. We will show that most of the remaining cases are unsolvable, namely, when $k \leq n/2$ or when $n-k$ is odd. In the rest of the paper, we drop the ℓ parameter, since it will always be equal to $k+1$. For these remaining cases, we provide partial results. First we give a simple argument to show that the task is unsolvable whenever $k \leq n/2$. Then, for the other values of k, we show that the task is unsolvable if $n-k$ is odd. The remaining cases are still open.

4.1 Impossibility proof when k is small

For simplicity, we first look at the case where $k=1$; we will see later that it can be easily generalized to any $k \leq n/2$. For $k=1$, the goal of the task is the following. If all the input values are the same, then the processes should disagree (i.e., not all of them should decide the same output). In all other cases, the processes should agree.



Let us assume for contradiction that the task is solvable in the immediate snapshot model. We focus on what happens in one of the initial global states where all the processes have the same input value a. Let us call $X = \{(P_0, a), \ldots, (P_{n-1}, a)\}$ the corresponding simplex of the input complex. After the processes exchange information using iterated immediate snapshot communications, we obtain in the protocol complex a chromatic subdivision $\chi(X)$ of the original simplex X. The situation for 3 processes and one round of immediate snapshot is

depicted above. The color of a vertex represents the process name; the value written inside a vertex is its input value; and next to it is the decision value.

In the input complex, the simplex X is surrounded by other simplices where not all inputs are the same. Thus, after the immediate snapshot computation occurs, the boundary of the subdivided simplex $\chi(X)$ is still surrounded by simplices with a different input value (three of them are depicted above, with input value b). In these simplices, the task specifies that all the processes must decide the same output. Assume w.l.o.g. that this output is 0. Since the boundary of $\chi(X)$ is connected, we can propagate the 0's step by step and we obtain that every vertex on the boundary of $\chi(X)$ must decide value 0.

To reach a contradiction, we will show that, given any assignment of output values 0 or 1 to the inner vertices of $\chi(X)$, there will always be at least one simplex of $\chi(X)$ where all the outputs are equal. This contradicts the task specification. Note that the next part of the proof only works because $\chi(X)$ is a *chromatic* subdivision; the statement does not hold for other subdivisions. If we regard the output values 0 and 1 as colors, the result that we want to prove looks like Sperner's lemma, where instead of counting the properly colored simplices inside the subdivision, we want to count the monochromatic ones, with respect to output values 0 and 1. To be able to use Sperner's lemma in this situation, we use a recoloring trick which was already used in [5] to obtain lower bounds on the renaming problem.

Suppose we are given an assignment of output values to the vertices of $\chi(X)$, such that all the vertices on the boundary decide 0. For v a vertex of $\chi(X)$, we write $d_v \in \{0, 1\}$ the decision value of v, and $\mathrm{id}_v \in \{0, \ldots, n-1\}$ its process number. We define the recoloring of v to be $c_v := (\mathrm{id}_v + d_v) \mod n$. We have the following property:

▶ **Lemma 7.** *A simplex σ of $\chi(X)$ is monochromatic w.r.t. the decision values d_v if and only if it is proper w.r.t. the recoloring c_v.*

Proof. Since $\chi(X)$ is a chromatic subdivision of the simplex X, every simplex σ of $\chi(X)$ is properly colored w.r.t. the process numbers id_v. Thus, if σ is monochromatic w.r.t. the decision values d_v (that is, if all its vertices have the same decision value d), it is clear that the recoloring $c_v = (id_v + d) \mod n$ will still be proper. Conversely, we proceed by contraposition: suppose that not all decision values are the same. Then, there must be two vertices v and w of σ, such that $\mathrm{id}_w = (id_v + 1) \mod n$ and $d_v = 1$ and $d_w = 0$. Thus $c_v = c_w$, and the recoloring c is not proper. ◀

We can now easily conclude the proof. Since on the boundary of $\chi(X)$ all the vertices have the decision value 0, the recoloring c is just the process number: $c_v = \mathrm{id}_v$ for every boundary vertex v. Moreover, since $\chi(X)$ is just the (iterated) chromatic subdivision of a single input simplex X, we know that the process numbers id_v form a Sperner coloring on its boundary. Thus, by Sperner's lemma [11], there must exist a simplex $\sigma \in \chi(X)$ which is properly colored w.r.t. the recoloring c, and by Lemma 7 this means that all the vertices of this simplex decide the same output. Therefore, the equality negation task with parameter $k = 1$ cannot be solved by iterated immediate snapshot.

The same proof can be adapted for any $k \leq n/2$:

▶ **Theorem 8.** *The equality negation task for $n \geq 3$ processes, with parameters $k \leq n/2$ and $\ell = k + 1$ is not solvable in the immediate snapshot model.*

Proof. The only difference with the case $k = 1$ above is that we now start with the simplex $X = \{(P_0, 0), \ldots, (P_{k-1}, k-1), (P_k, 0), \ldots, (P_{2k-1}, k-1), \ldots\}$, which has exactly k distinct input values, and every input appears at least twice. Therefore, if we remove one vertex

from X, we obtain a face of X (of cardinality $n-1$) which still has k distinct inputs. Then, this face belongs to another simplex with $k+1$ input values. Thus, after subdivision, every process on the boundary of X must decide the same output (say, 0). Now the rest of the proof is exactly the same as in the case of $k=1$: we have a simplicial complex $\chi(X)$ which is a chromatic subdivision of X, and we know that every process on its boundary has decision value 0. By Sperner's lemma, there is a simplex $\sigma \in \chi(X)$ which is properly colored w.r.t. the recoloring c, and by Lemma 7 it is monochromatic w.r.t. the decision values. Since X has $\leq k$ distinct input values, this set of outputs is not allowed. ◀

4.2 Some impossibility results depending on the parity of $n-k$

We now extend the proof of Section 4.1 to show that the task is not solvable by iterated immediate snapshot in half of the remaining cases: namely, whenever $n-k$ is odd, the task is unsolvable. We will rely on the same recoloring trick (and the associated Lemma 7), but instead of Sperner's lemma, we will use a slightly more powerful combinatorial topology tool called the *index lemma*. Sperner's lemma is equivalent to Brouwer's fixed point theorem which in turn, can be seen as reducing to distinguishing between the topology of a ball of dimension n with its boundary, the $(n-1)$-dimensional sphere. In more details, their topology is essentially different since there cannot be a continuous map from the n-ball *onto* (meaning, surjective) the $(n-1)$-sphere. The index lemma relates to a more subtle topological distinction, about the inexistence of continuous maps that would be winding around holes in a non-consistent way, so is not just about the image of the map. Sperner's lemma can be recovered as a direct consequence of the index lemma.

4.2.1 Low-dimensional example

Before we proceed to the general construction, let us illustrate the idea of the proof using the particular case of 3 processes and parameter $k=2$. This instance of the task is the following: if the three input values are different, then the processes should decide the same output; otherwise, they should disagree. The input complex $\mathcal{I}$ contains every possible assignment of input values $\{0,1,2\}$ to the three processes $\{P,Q,R\}$. More formally, its maximal simplices are of the form $\sigma_{ijk} = \{(P,i),(Q,j),(R,k)\}$, for all $i,j,k \in \{0,1,2\}$. We can decompose this input complex $\mathcal{I}$ into two parts (depicted in Figure 3 below):

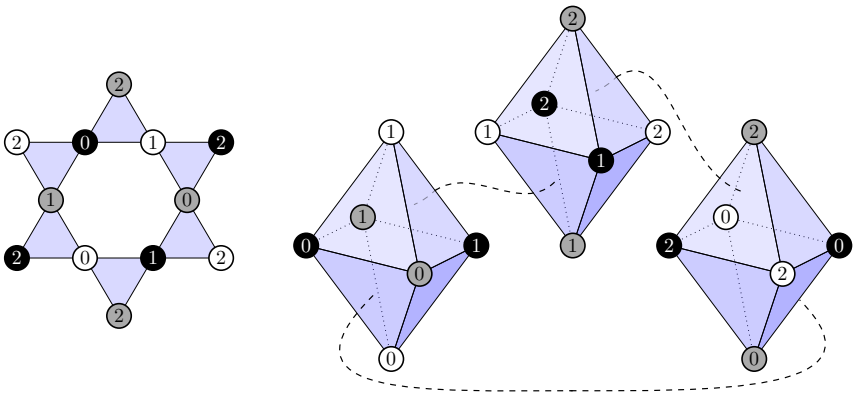


Figure 3 Exploded view of the input complex for $n=3$ processes. All vertices with the same color and input value should be identified.

- There are 6 maximal simplices σ_{ijk} such that $|\{i,j,k\}| = 3$, that is, where all three processes have distinct values. In each of these simplices, the processes should agree on a common output value; moreover, since this subcomplex is connected, this common output has to be the same in all six of them.
- The rest of the input complex consists of simplices σ_{ijk} such that $|\{i,j,k\}| \leq 2$. In that part of the input, the processes should not agree. This part of the input complex is topologically a "pearl necklace" of three spheres.

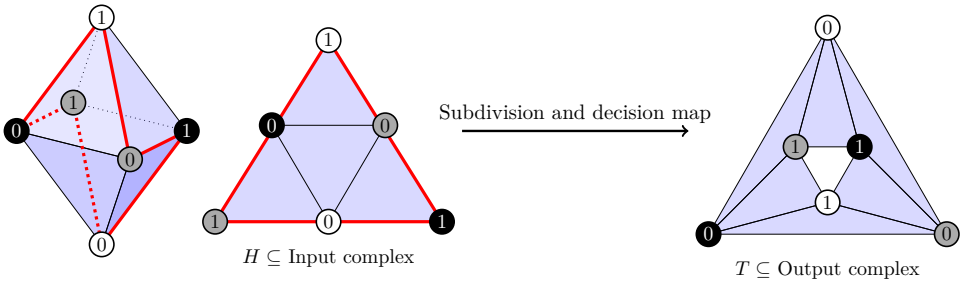
Note that the 6 simplices with three distinct inputs are actually filling the hole in the middle of the "necklace". Thus, $\mathcal{I}$ is homotopy equivalent to a wedge of 2-spheres; in the terminology of [11], it is a *pseudosphere*.

We now focus on one of the three spheres depicted in Figure 3; for example, the one where all inputs are either 0 or 1. That sphere has six edges which are labeled with two distinct input values 0 and 1. Each of these 01-edges also belongs to one of the six simplices with three inputs $0, 1, 2$: therefore, on each of these edges, all processes have to decide the same output value (say, w.l.o.g., that the common output value is 0). In the picture below, the 01-edges of the sphere are depicted in red. They form a circle on the surface of the sphere, splitting it in two half-spheres. We now look at only one of those two half-spheres, and call it H. It consists of four simplices, and its boundary contains only 01-edges. Thus, after the immediate snapshot computation occurs, this complex will be subdivided, and in order to solve the task, we should satisfy the following two conditions:

(1) All the vertices on the (red) boundary must be mapped to the same output, say, 0; and

(2) In every maximal simplex, not all processes should decide the same output.

This means we should have a simplicial map from a chromatic subdivision of H to the subcomplex T of the output complex where not all decision values are the same.



Since the processes on the boundary of H all decide output 0, the boundary of H has to be mapped to the outside boundary of T. Therefore, it seems clear topologically that some simplex inside H will necessarily be sent to the "111-hole" in the middle of T, contradicting condition (2). Notice however that the boundary of H is winding *twice* around the boundary of T. Moreover, this winding number is preserved by the chromatic subdivisions of H induced by the immediate snapshot protocol. Sperner's lemma, that we used in Section 4.1, only works if the boundaries are matched exactly (i.e., if we have a Sperner coloring). That's why we need to use the index lemma, which is able to handle cases where one boundary is winding several times around the other.

▶ **Proposition 9.** *The equality negation task for $n = 3$ processes, with parameters $k = 2$ and $\ell = 3$ is not solvable in the immediate snapshot model.*

Proof. Assume for contradiction that the task is solvable, and let H be the subcomplex of the input complex described above. So, there should be a chromatic subdivision $\chi(H)$ of H,

and an assignment of decision values to its vertices, which solves the task. Remember that all the vertices on the boundary of $\chi(H)$ must decide the same output value, for example 0. We choose an arbitrary (coherent) orientation for the simplices of H; and we assign colors using the same recoloring as in Section 4.1. What we want to do now is prove that the content of $\chi(H)$ is non-zero. Using the index lemma, we can also compute its index. And by Corollary 4, applied to the boundary of $\chi(H)$, it is equal to the index of H itself.

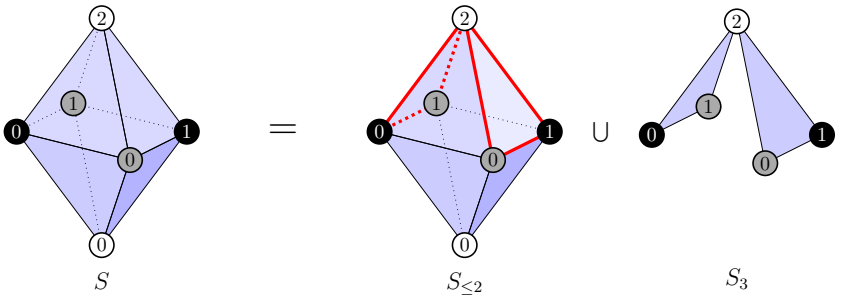
Then, an easy calculation shows that the index of H is either 2 or -2 depending on the orientation that we chose. So, no matter how many rounds of immediate snapshot we do, the index of $\chi(H)$ is either 2 or -2. By the index lemma, its content must be equal to its index (in absolute value). Since it is non-zero, there must exist proper triangles in $\chi(H)$, which by Lemma 7 correspond to monochromatic triangles w.r.t. the decision values. This contradicts the solvability of the task. ◀

4.2.2 General case

We now work with any k and n (remember however that $k < n$ since $\ell = k + 1 \leq n$), and we try to follow the same recipe as in the previous section. So, our goal is to find a subcomplex of the input complex, which is a pseudomanifold (in order to be able to apply the index lemma), and whose boundary consists of simplices with exactly k distinct input values (so that we can say that every process on the boundary has to decide the same output). A simple way to obtain a pseudomanifold is to choose a sphere in the input complex, and restrict ourselves to a subcomplex of this sphere. Although this idea is inspired from what we did in the low-dimensional example, the reader should be warned that the general construction we are about to do, when instantiated with $n = 3$ and $k = 2$, does not give the same proof as the one of Section 4.2.1.

Consider the subcomplex S of the input complex, which contains all the vertices of the form (P_i, a_i), where $a_i = 0$ or $a_i = i$ if $1 \leq i \leq k$, and $a_i = 0$ or $a_i = 1$ for all other values of i. The simplices of S are all combinations of such vertices, which are properly colored w.r.t. the process names. For example, $\sigma = \{(P_0, 0), (P_1, 0), \ldots, (P_{n-1}, 0)\}$ is a simplex of S. For ease of notation, when talking about maximal simplices, we omit the process names and just write $\sigma = \langle 0, \ldots, 0\rangle$, where the i-th component is the value of process P_i. Another simplex of S is $\langle 0, 1, 2, \ldots, k, 0, \ldots, 0\rangle$.

For the case $n = 3$, $k = 2$, the sphere $S = \{\langle a, b, c\rangle \mid a, b \in \{0, 1\}, c \in \{0, 2\}\}$ is represented below (the colors black, gray, white correspond to P_0, P_1, P_2, respectively).



Since every process can take exactly two different input values, independently from the other processes, the complex S is a $(n-1)$-sphere. In particular, it is a pseudomanifold. Among the maximal simplices of S, some of them contain exactly $k+1$ distinct input values, and the others contain k values or fewer. We write $S_{k+1} \subseteq S$ for the subcomplex of S which contains all the simplices with $k+1$ input values, and $S_{\leq k} = S \setminus S_{k+1}$. Both S_{k+1} and $S_{\leq k}$

are pseudomanifolds, and they have the same boundary $\partial S_{k+1} = \partial S_{\leq k} = S_{k+1} \cap S_{\leq k}$. In every simplex of S_{k+1}, the processes must decide the same value (since S_{k+1} is connected); assume w.l.o.g. that they decide the output value 0.

The complex $S_{\leq k}$ will play the role of the complex H that we had in Section 4.2.1. In the case of $n = 3$, $k = 2$, it corresponds to the sphere S with two holes corresponding to the two simplices with three distinct inputs. Its boundary is represented in red in the picture above. So the situation is a bit different than what we had in Section 4.2.1: instead of one boundary winding twice around the output, we now have two holes, each of them winding once around the output complex. Fortunately, the index lemma can also deal with such cases, as long as the two holes are winding in the same direction around the output (otherwise they would cancel each other). This is taken into account when we compute the index: here, one can check that the two holes have the same orientation, so the index will be either 2 or -2. For general k and n, the boundary of $S_{\leq k}$ can have many holes, each of them winding several times around the output complex. Our goal is once again to prove that the index is non-zero.

▶ **Theorem 10.** *The equality negation task for $n \geq 3$ processes, with parameters k and $\ell = k + 1$ such that $n - k$ is odd, is not solvable in the immediate snapshot model.*

Proof. After the immediate snapshot communication occurs, we obtain a chromatic subdivision $\chi(S_{\leq k})$. We already know that it is a pseudomanifold, and that on its boundary every process has to decide 0. Our goal is now to use the index lemma to prove that there exists a simplex inside $\chi(S_{\leq k})$ which is monochromatic w.r.t. the decision values. First, we use the recoloring of Lemma 7, so that we are now searching for a properly colored simplex in $\chi(S_{\leq k})$. All we need to show is that the content of $\chi(S_{\leq k})$ is non-zero; thus, we want to compute its index. Since the index is preserved by chromatic subdivisions (see Corollary 4), we just need to compute the index of $S_{\leq k}$. Since the boundary of $S_{\leq k}$ is the same as the boundary of S_{k+1}, their index is the same (in absolute value), and since the index of S_{k+1} is equal to its content, we want to compute the content of S_{k+1}. In S_{k+1}, every simplex is properly colored (because everyone decides 0), so we just need to count the simplices of S_{k+1} by orientation. If the result is non-zero, this concludes the proof.

So let us pick an arbitrary orientation, say that the simplex $\langle 0, \ldots, 0\rangle$ is oriented positively. Each time we change the value of one coordinate, the orientation changes: for example, the simplex $\langle 1, 0, \ldots, 0\rangle$ is oriented negatively. Let us first characterize the maximal simplices of S_{k+1}: they are of the form $\langle a_0, a_1, 2, 3, \ldots, k, a_{k+1}, \ldots, a_{n-1}\rangle$, where for each $i \notin \{2, \ldots, k\}$, $a_i \in \{0, 1\}$, and at least one of the a_i must be 0, and at least one must be 1. There are exactly $2^{n-k+1} - 2$ such simplices: all combinations of 0's and 1's are possible, except for $z = \langle 0, 0, 2, 3, \ldots, k, 0, \ldots, 0\rangle$ and $u = \langle 1, 1, 2, 3, \ldots, k, 1, \ldots, 1\rangle$. To go from z to u, we need to change $n - k + 1$ coordinates. Thus, since $n - k + 1$ is even, then z and u have the same orientation; otherwise, they would have opposite orientations.

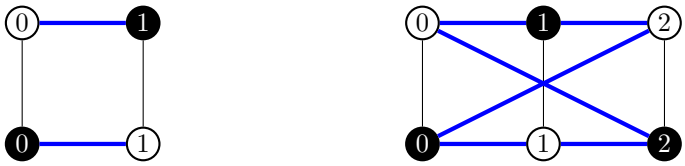
A simple calculation shows that summing the orientations of all the simplices of S_{k+1}, plus z and u, gives 0 as a result. Indeed, if we omit the middle components "$2, 3, \ldots, k$", these 2^{n-k+1} simplices correspond to all the binary sequences of size $n - k + 1$. We can, for example, enumerate those sequences using Gray code, so that the orientations are alternating, hence there is the same number of positively and negatively oriented simplices. Since z and u are not in S_{k+1}, and they have the same orientation, then the content of S_{k+1} must be either 2 or -2. In any case, it is non-zero: so the task is not solvable whenever $n - k$ is odd. ◀

In particular, the case of $k = n - 1$ is not solvable for any n, as $n - k = 1$ is odd. Note that in this proof, we do not really use the full power of the index lemma: we use it to show that the content of $S_{\leq k}$ is equal (in absolute value) to the content of S_{k+1}. This also follows

from the simple fact that the content of a sphere is always 0, which is a direct consequence of the index lemma; so the contents of $S_{\leq k}$ and S_{k+1} must be opposite.

5 Discussion on the number of input values

In all this paper, we have been working with a set of input values $I = \{0, \ldots, n-1\}$ of size n, for a number $n \geq 3$ of processes. This might seem a bit surprising, considering that in the original equality negation task for two processes [14], the size of the input set matters a lot. For $I = \{0, 1\}$, the task is solvable, but for $I = \{0, 1, 2\}$, it becomes unsolvable. It is quite informative to think about why our unsolvability proofs of Theorems 8 and 10 fail in the case of two processes and $I = \{0, 1, 2\}$. Both of them fail for a similar reason. We identify a subcomplex of the input complex ($\chi(X)$ in Section 4.1 and $S_{\leq k}$ in Section 4.2.2), and we surround its boundary by simplexes where every process must decide the same output. However, in order to assume that every vertex on the boudary decides the same output 0, we need to know that the part of the input complex which decides the same output is connected. For two processes, this is not the case if $I = \{0, 1\}$, however taking $I = \{0, 1, 2\}$ fixes this issue. Both input complexes are depicted below; the subcomplex where decisions should be the same is represented in blue.



No such problem occurs when there are more than 3 processes. Nevertheless, we can still wonder what happens when we allow an input set I of size $|I| > n$. Actually, all the results of this paper can be extended to such a setting in a straightforward way:

- Theorem 5 can easily be shown to work when more than n input values are allowed: intuitively, the algorithm relies only on the size of the sets of values that are seen by the processes. It never looks at the actual values.
- In all our unsolvability proofs (Theorems 8 and 10, and Proposition 9), we proceed by picking a subcomplex of the input complex, and then we work in this subcomplex to find a contradiction. If we add more input values, this just makes the input complex bigger, but all those proofs still work as they stand.

6 Conclusion

We have defined a family of equality negation tasks and studied their solvability and unsolvability. A few cases remain open questions; we conjecture that they should be unsolvable. The same proof method as in Section 4.2.2 using the index lemma might work for the remaining cases, but we would need to find another subcomplex of the input complex on which to apply it. Unfortunately, our attempts to do so have failed.

There are a number of other variations of this task that we could have studied. For example, instead of having binary outputs in $\{0, 1\}$, we could have a larger set of output values. Then, we could introduce two more parameters to define what it means to "agree" or to "disagree" in that context. That kind of parameters appear in the generalized symmetry breaking task [4].

References

1 H. Attiya and S. Rajsbaum. The combinatorial structure of wait-free solvable tasks. *SIAM J. Comput.*, 31(4):1286–1313, 2002. doi:10.1137/S0097539797330689.

2 E. Borowsky, E. Gafni, N. Lynch, and S. Rajsbaum. The BG distributed simulation algorithm. *Distrib. Comput.*, 14(3):127–146, October 2001. URL: http://dx.doi.org/10.1007/PL00008933, doi:10.1007/PL00008933.

3 Armando Castañeda, Maurice Herlihy, and Sergio Rajsbaum. An equivariance theorem with applications to renaming. *Algorithmica*, 70(2):171–194, Oct 2014. URL: https://doi.org/10.1007/s00453-013-9855-3, doi:10.1007/s00453-013-9855-3.

4 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.

5 Armando Castañeda and Sergio Rajsbaum. New combinatorial topology bounds for renaming: the lower bound. *Distributed Computing*, 22(5):287–301, Aug 2010. doi:10.1007/s00446-010-0108-2.

6 Benny Chor, Amos Israeli, and Ming Li. On processor coordination using asynchronous hardware. In *Proceedings of the Sixth Annual ACM Symposium on Principles of Distributed Computing*, PODC '87, pages 86–97, New York, NY, USA, 1987. ACM. URL: http://doi.acm.org/10.1145/41840.41848, doi:10.1145/41840.41848.

7 Ky Fan. Simplicial maps from an orientable n-pseudomanifold into S^m with the octahedral triangulation. *Journal of Combinatorial Theory*, 2(4):588–602, 1967. URL: http://www.sciencedirect.com/science/article/pii/S0021980067800632, doi:https://doi.org/10.1016/S0021-9800(67)80063-2.

8 Éric Goubault, Marijana Lazić, Jérémy Ledent, and Sergio Rajsbaum. A dynamic epistemic logic analysis of the equality negation task. *Dynamic Logic: New Trends and Applications, DaLi 2019*, to appear.

9 Michael Henle. *A Combinatorial Introduction to Topology*. Dover, 1983. doi:10.2307/1574757.

10 Maurice Herlihy. Wait-free synchronization. *ACM Trans. Program. Lang. Syst.*, 13(1):124–149, January 1991. URL: http://doi.acm.org/10.1145/114005.102808, doi:10.1145/114005.102808.

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

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

13 Prasad Jayanti. On the robustness of Herlihy's hierarchy. In *Proceedings of the Twelfth Annual ACM Symposium on Principles of Distributed Computing*, PODC '93, pages 145–157, New York, NY, USA, 1993. ACM. URL: http://doi.acm.org/10.1145/164051.164070, doi:10.1145/164051.164070.

14 Wai-Kau Lo and Vassos Hadzilacos. All of us are smarter than any of us: Nondeterministic wait-free hierarchies are not robust. *SIAM J. Comput.*, 30(3):689–728, 2000. URL: https://doi.org/10.1137/S0097539798335766, doi:10.1137/S0097539798335766.

15 Hosame H Abu-Amara Michael C Loui. Memory requirements for agreement among unreliable asynchronous processes. In *Advances in Computing research*, pages 163–183, Greenwich, CT, 1987. JAI Press.

16 Sergio Rajsbaum, Michel Raynal, and Panagiota Fatourou. An introductory tutorial to concurrency-related distributed recursion. *Bulletin of the EATCS*, 111, 2013. URL: http://eatcs.org/beatcs/index.php/beatcs/article/view/223.