The Opinion Number of Set-Agreement *

Pierre Fraigniaud
CNRS and University Paris Diderot, France. Pierre.Fraigniaud@liafa.univ-paris-diderot.fr†

Sergio Rajsbaum
Instituto de Matemáticas, UNAM, Mexico rajsbaum@math.unam.mx‡

Matthieu Roy
LAAS-CNRS, Univ. Toulouse, Franceroy@laas.fr

Corentin Travers
CNRS and U. of Bordeaux, France travers@labri.fr§

Abstract

This paper carries on the effort to bridging runtime verification with distributed computability, studying necessary conditions for monitoring failure prone asynchronous distributed systems. In a previous paper (to appear in RV 2014), it has been proved that there are correctness properties that require a large number of opinions (e.g., true, false, perhaps, probably true, probably no, etc.) to be monitored. The main outcome of this paper is that the need for this large number of opinions is not an artifact induced by the existence of artificial constructions. Instead, monitoring an important class of properties, requiring processes to produce at most k different values does requires a large number of opinions. Specifically, our main result is a proof that it is impossible to monitor k-set-agreement in an n-process system with fewer than $\min\{2k, n\}+1$ opinions. The lower bound is shown to be tight.

1 Introduction

Monitoring correctness properties at runtime, is a well established research domain. This domain has been recently receiving more attention, with the creation in 2001 of an annual international conference on *runtime verification*. The essential objective of runtime verification is to determine, at any point in time, whether a system is in a legal or illegal state, with respect to a given specification.

In runtime verification *monitors* are hardware or software components in charge of surveying the state of the system. In particular, the case of a distributed system whose execution is observed by several monitors has been considered in, e.g., [4, 6, 17]. As soon as a violation of the legality of the execution is revealed by any of these monitors at runtime, recovery code can be executed for bringing the system back to a legal state. The monitors may communicate with each other, and

*Work supported in part by ECOS-nord project M12M01.
†Additional supports from the ANR project DISPLEXITY, and from the INRIA project GANG.
‡Additional support from UNAM-PAPIIT project and CONACYT LAISLA.
§Additional support from ANR project DISPLEXITY.

every monitor i typically produces a binary *opinion* $o_i \in \{\text{true}, \text{false}\}$ so that a recovery code is launched as soon as one of these opinions is false.

When processes may fail and the distributed system is asynchronous, it is however no longer sufficient for the monitors to produce only two opinions. It has indeed been shown in [11] that there are correctness properties that cannot be monitored by interpreting the opinions produced by the monitors using the logical conjunction of all the opinions, where the system is in legal state if and only if all opinions are true. Even more problematic is the fact that it was also shown [12] later on that there are correctness properties for which no decentralized monitoring exists, even if we let the number of opinions grow to an arbitrary large constant $k \geq 2$. This holds no matter what the interpretation of the opinions is (logical conjunction or otherwise). The model studied in [11, 12] consists of a distributed system composed of n asynchronous processes communicating by reading and writing to a shared memory, and in which any number of processes may fail by crashing. This classical model has been studied in depth in the literature (see textbooks [14, 16]). In particular, it is known to be equivalent (with respect to task computability) to a message-passing model where less than half processes may crash, and other more powerful models can be reduced to it.

To be more specific, we consider the same crash-prone asynchronous distributed system, and assume as in [11, 12] that each process has a variable that contains the output of a computation that needs to be monitored. The simplest situation is considered, i.e., where the variable at each process is written only once. The correctness property to be monitored describes the set of values that are allowed to appear in these variables simultaneously. Such a set of values is called a *instance*. The set of correct instances could be expressed by a logic formula, but it turned out to be convenient to describe the correctness property by simply listing the set of all legal instances. This set is called a *distributed language*. The following illustrative example was detailed in [12], as it is arising often in practice [5]. Let us consider a system where *requests* are sent by clients, and *acknowledged* by servers. The execution is correct when (1) all requests have been acknowledged, and (2) every received acknowledgement corresponds to a previously sent request. Each monitor i observes a variable reporting a pair (R_i, A_i), namely, the subset R_i of requests that has been received by the servers, and the subset A_i of acknowledgements that has been sent by the servers. The corresponding distributed language is the set of instances whose variables satisfy conditions (1) and (2).

To monitor the system, processes observe the values in their variables, and communicate with each other as long as needed. Eventually each process must produce its opinion about the legality of the instance with respect to the given language. It is required that the multisets of opinions produced in case the instance is legal differs from those produced for illegal instances. This partition of multisets of opinion is captured by an *interpretation*. In the most common setting, each opinion is in the set $\{\text{true}, \text{false}\}$, and the (global) interpretation of opinions is the logical conjunction of these opinions. However, as mentioned above, there are languages that require more opinions and more complex interpretations that logical conjunction [11, 12].

In [12], it was shown that, for any k, $1 \leq k \leq n$, there exists a distributed language requiring monitors to produce at least k distinct opinions in a system with n monitors. To establish this result, an ad hoc language was constructed. The existence of a natural language that would be arbitrarily difficult in terms of the number of opinions needed to monitor it was left open.

In this paper, we study the number of opinions needed to monitor the family of *set agreement* tasks, widely studied in the distributed computing literature since its introduction in [8]. Recall that, for $n \geq 1$ processes, and $1 \leq k \leq n$, the (n, k)-set-agreement task is specified as follows. Each

processes proposes a *value*, from some ground set. After communicating with the other processes, each process has to decide on one of the proposed values, such that at most k different proposed values are decided. Hence, if $k = 1$ then we get the classic *consensus* task [9], and, as we increment the value of k, we get easier agreement tasks, until we get the trivial task for $k = n$.

More precisely, we consider the language $\mathcal{L}_{n,k}$ corresponding to (n, k)-set-agreement. The value to be monitored at each process i, $1 \leq i \leq n$, consists of a pair (s_i, t_i). Consider the set of pairs $u = \{(s_{i_1}, t_{i_1}), (s_{i_2}, t_{i_2}), \ldots, (s_{i_\ell}, t_{i_\ell})\}$ present at some time in the system, where the pair (s_{i_j}, t_{i_j}) is the variable of process i_j, $j = 1, \ldots, \ell$. Note that we may have $\ell < n$ since some processes may crash crash or be arbitrarily slow. Then u is in the language $\mathcal{L}_{n,k}$ (i.e., represents a correct execution of an algorithm $\mathcal{A}$ pretending to solve (n, k)-set-agreement) if and only if (1) there are at most k different t_{i_j}, $j = 1, \ldots, \ell$ (i.e., at most k different values have been decided by $\mathcal{A}$), and (2) for each t_{i_j}, there must exist a pair $(s_{i_{j'}}, t_{i_{j'}})$ in u such that $s_{i_{j'}} = t_{i_j}$ (i.e., t_i has been proposed by at least one participating process).

1.1 Our results

We first prove that the language $\mathcal{L}_{n,k}$ can be monitored using $\min\{2k, n\} + 1$ opinions. Then, our main result is a proof that it is impossible to monitor $\mathcal{L}_{n,k}$ with fewer opinions. Thus, in particular, consensus can be monitored with 3 opinions, and cannot be monitored with less than 3 opinions, even using interpretations different from the logical conjunction of boolean opinions.

The upper bound is a simple adaptation of the universal algorithm presented in [12]. Our proof technique for the lower bound is also inspired from the lower bound in [12], and, as such, is also based on combinatorial topology arguments using Sperner's lemma. However, a careful analysis of the *alternation number* of the set agreement language had to be achieved in this paper. This parameter captures the number of times a sequence of instances can alternate between legal and illegal over an execution of the system.

Hence, the main outcome of this paper is the perhaps surprising result, that the difficulty of monitoring set agreement (in terms of number of opinions) is captured by the formula $\min\{2k, n\}+1$. The difficulty grows linearly as k is increased, at double the rate, and only until $2k = n$. For larger values of k, the number of opinions stays at its maximum, equal to $n + 1$ (no task requires more opinions). Indeed, monitoring the most important case in distributed computing, which is $k = n - 1$, does not require more opinions than when $2k = n$. We conjecture that the number of opinions has a direct relation with the number of logic values needed to design a temporal logic framework, and it is well known that more logic values dramatically increase the difficulty of reasoning in the logic. Hence this paper continues motivating further research at the border between runtime verification and distributed computability, in the context of studying necessary conditions for monitoring asynchronous distributed systems susceptible to failures.

1.2 Related work.

Most work on decentralized monitoring, is based on logical frameworks, using LTL or variants of it, see e.g. [4], where a formula ϕ is decomposed into local formulas, so monitor i evaluates locally ϕ_i, and emits a boolean-valued opinion. In our terminology, a logical conjunction interpratation is used. That is, it is assumed a global violation can always be detected locally by a process. See [12] for a more detailed discussion logic-based runtime verification. To the best of our knowledge, the effects of asynchrony and failures in a decentralized monitoring setting were considered for the first

time in [11], and subsequently in [12]. Related work in the distributed computing literature includes papers such as [7] for stable property detection in a failure-free message-passing environment, and e.g. [3] for distributed program checking in the context of self-stabilization.

1.3 Organization of this paper.

The distributed system model and the notion of distributed languages are defined is in Section 2. Wait-free monitoring is presented in Section 3. A monitor for the language $\mathcal{L}_{n,k}$ appears in Section 4. The lower bound on the number of opinions to monitor $\mathcal{L}_{n,k}$ is shown in Section 5. We conclude the paper and mention some open problems in Section 6.

2 Preliminaries

We consider the standard *wait-free shared memory model* [16]. The system consists of n processes denoted $\{p_1, \ldots, p_n\}$ that communicate via a shared memory. Processes are asynchronous and any number of them may fail by crashing (i.e., halt and never recover.). The shared memory is made of n shared registers, one per process, that support atomic read and write operations. To simplify the design of protocol, we assume that *atomic snapshots* [1] are available. An atomic snapshot consists in an n-components array R and supports two operations *update(v)* and *snapshot()*. *update(v)* by process p_i sets the value of $R[i]$ to v and *snapshot()* reads atomically all the components [1].

A *task* specifies for each process possible inputs and for each possible assignment of inputs to the processes, which are the valid outputs. More precisely, an *input* or *output set* is a non-empty set $s = \{(id_1, v_1), \ldots, (id_\ell, v_\ell)\}$ where $id_1, \ldots, id_k$ are distinct processes identities[1] and $v_1, \ldots, v_k$ are values from some set $\mathcal{V}$. s specifies an assignment of input or output values to each processes in $\mathrm{ID}(s) = \{id_1, \ldots, id_k\}$. The set of values in s is denoted by $val(s)$. Sometimes, the values in s are seen as a multiset, in which case we use the notation $mval(s)$. Two input or output sets s, t *match* if $\mathrm{ID}(s) = \mathrm{ID}(t)$. A task is specified by a triplet $(\mathcal{I}, \mathcal{O}, \Delta)$ where $\mathcal{I}$ and $\mathcal{O}$ are inclusion-closed sets of input and output sets respectively, and Δ maps each input set $s \in \mathcal{I}$ to a non-empty set of matching output sets.

In the *(n, k)-set agreement* task [8], each process input is a value taken from some set $\mathcal{V}$ of cardinality larger than k. Each process is required to output a value which is the input value of a participating process such that no more than k distinct values are output. More precisely, $\mathcal{I}$ is the set of the input sets s with $val(s) \subseteq \mathcal{V}$ and $t = \{(id_1, v_1), \ldots, (id_\ell, v_\ell)\} \in \mathcal{O}$ if and only if $|val(t)| \leq k$. For any input set $s \in \mathcal{I}$, $\Delta(s) = \{t \in \mathcal{O} : \mathrm{ID}(s) = \mathrm{ID}(t) \text{ and } val(t) \subseteq val(s)\}$.

2.1 Distributed Languages

Distributed languages have been introduced in [12] as a simple way to represent correctness properties of distributed systems. Given an alphabet of symbols A, a *distributed language* over A is a set of non-empty input sets with, for each $s \in \mathcal{L}$, $val(s) \subseteq A$. In the context of distributed languages, input sets s with $val(s) \subseteq A$ are called *instances*. An instance s is *legal* if $s \in \mathcal{L}$ and *illegal* otherwise. A distributed language might specify a global property that a distributed system should satisfy. In this case, each element of A encodes a local state and the sets in $\mathcal{L}$ represent the global states of the system that satisfy the property.

[1] Identities are equal to processes indexes. That is, the identity of process p_i is i.

We focus on checking that the result of a distributed computation satisfies the specification of a given task. More precisely, given a task $T = (\mathcal{I}, \mathcal{O}, \Delta)$, we define the *language $\mathcal{L}_T$ induced by T* as follows. Let $V_\mathcal{I}$ and $V_\mathcal{O}$ be respectively the sets of possible input and output values of T. Language $\mathcal{L}_T$ is defined over the alphabet $V_\mathcal{I} \times V_\mathcal{O}$. That is, each process input is a pair $(s_i, t_i) \in V_\mathcal{I} \times V_\mathcal{O}$. Given an input set $u = \{(id_1, (s_1, t_1)), \ldots, (id_\ell, (s_\ell, t_\ell))\}$, let $s_u = \{(id_1, s_1), \ldots, (id_\ell, s_\ell)\}$ and $t_u = \{(id_1, t_1), \ldots, (id_\ell, t_\ell)\}$. Then, $u \in \mathcal{L} \iff t_u \in \Delta(s_u)$. In other words, an input set u belongs to $\mathcal{L}_T$ if and only if each process i starting with input value s_i is allowed to decide t_i, according to the specification of T.

The language $\mathcal{L}_{n,k}$ induced by (n, k)-set agreement is defined as follows. The alphabet over which $\mathcal{L}_{n,k}$ is defined is $\mathcal{V} \times \mathcal{V}$ where $\mathcal{V}$ is the set from which values are proposed in the (n, k)-set agreement task. Then, for any instance $s = \{(id_1, (s_1, t_1)), \ldots, (id_\ell, (s_\ell))\}$, $s \in \mathcal{L}_{n,k}$ if and only if $|\{t_1, \ldots, t_\ell\}| \leq k$ and $\{t_1, \ldots, t_\ell\} \subseteq \{s_1, \ldots, s_\ell\}$.

3 Wait-free languages monitoring

As defined in [12], monitoring the correctness specified by a distributed language $\mathcal{L}$ over an alphabet A involves two components: an *opinion-maker* M, and an *interpretation* μ. The opinion-maker is a distributed protocol. For each process p_i, its input is a pair (id_i, a_i) where $a_i \in A$ and its output is an *opinion* about the legality of the input set. The processes running this algorithm are called *monitors*, and the (finite) set of possible individual opinions U, the *opinion set*.

The interpretation μ specifies how the collection of individual opinions produced by the monitors should be interpreted. It is required that the opinions of the monitors should be able to distinguish legal input sets from illegal ones according to $\mathcal{L}$. Thus, $\mu = (\mathbf{Y}, \mathbf{N})$ is a partition of all multi-sets of at most n elements over U. $\mathbf{Y}$ is called the "yes" set, and $\mathbf{N}$ is called the "no" set.

A pair (M, μ) is *a monitor for* language $\mathcal{L}$ over alphabet A if the following holds:

- For the opinion-maker protocol M, input of of each process i is any element a_i of A and the output an opinion u_i. M is required to be wait-free: each participating process is required to decide an opinion in a finite number of its own steps, regardless of the behavior of the other processes.

- For any input set s, in any execution of M with input s where all participating processes decide, the multiset S of opinions that are decided satisfies:

$$s \in \mathcal{L} \iff S \in \mathbf{Y}.$$

  Thus, if the input set s is illegal, i.e., $s \notin \mathcal{L}$, then the processes must produce a multiset of opinions in $\mathbf{N}$.

By extension, a pair (M, μ) is a *monitor for a task T* if (M, μ) is a monitor for the induced language $\mathcal{L}_T$.

3.1 Example: AND-interpretation

When the set of opinions consists only in two values, i.e., $U = \{0, 1\}$, a natural interpretation μ is induced by the AND-operator [10, 11] as follows. For every multi-set of opinions S, $S \in \mathbf{Y}$ if every opinion in S is 1, otherwise, $S \in \mathbf{N}$. Intuitively, a process outputs 1 if according to its view the

instance is legal and 0 otherwise. Note that the interpretation is id-oblivious: for each opinion, the identities of the processes that produce that opinion is not taken into account by the interpretation.

Not every language has a wait-free monitor with an AND-interpretation. Indeed, the languages that can be monitored with the AND-interpretation are exactly those that are *projection-closed*[2] [11]. In particular, the language induced by the consensus task is not projection-closed and therefore does not have a wait-free monitor with the AND-interpretation. In fact, our main result implies that three opinions are necessary and sufficient to wait-free monitor consensus.

3.2 Opinion number and alternation number

The *opinion number* #opinion($\mathcal{L}$) of a language $\mathcal{L}$, is the smallest size of the opinion set U for which there exists a monitor with opinion U for $\mathcal{L}$. The *alternation number* #altern($\mathcal{L}$) of a language $\mathcal{L}$ is the longest sequence of increasing instances that alternates between legal and illegal instances. More precisely, #altern($\mathcal{L}$) is the largest integer ℓ for which there exists instances $s_1 \subset s_2 \subset \ldots \subset s_\ell$ such that for every i, $1 \leq i < \ell$, $s_i \subset s_{i+1}$, and either $s_i \in \mathcal{L} \wedge s_{i+1} \notin \mathcal{L}$ or $s_i \notin \mathcal{L} \wedge s_{i+1} \in \mathcal{L}$.

A strong relationship between opinion and alternation numbers of languages is exposed in [12]. First, it is established that the alternation number gives an upper bound on the opinion number: For every language $\mathcal{L}$, #opinion($\mathcal{L}$) $\leq$ #altern($\mathcal{L}$)+1. Second, it is shown that for any $k, 1 \leq k \leq n$, there exists a n-processes language $\mathcal{L}$ with #altern($\mathcal{L}$) $= k$ that requires at least k opinions to be monitored, i.e., #opinion($\mathcal{L}$) $\geq k$.

For (n, k)-set agreement, the alternation number of the induced language $\mathcal{L}_{n,k}$ is:

Lemma 3.1. #altern($\mathcal{L}_{n,k}$) $= \min\{2k+1, n\}$.

Proof. Consider the following sequence of instances: $s_1 = \{(1,(0,1))\}$, $s_2 = \{(1,(0,1)),(2,(1,1))\}$, ..., $s_{2i} = \{(1,(0,1)),(2,(1,1)),\ldots,(2i,(i,i))\}$, $s_{2i+1} = \{(1,(0,1)),\ldots,(2i,(i,i)),(2i+1,(i,i+1))\}$,... for $i : 2i, 2i+1 \leq n$. Note that $s_{2i+1} \notin \mathcal{L}_{n,k}$ as this instance corresponds to the case in which a value, namely $i+1$ is decided whereas it has not been proposed. In s_{2i}, however, i values are decided and each of them is also a proposed value. Thus, if the number of decided values is less than or equal to k, that is, $i \leq k$, $s_{2i} \in \mathcal{L}_{n,k}$. Hence, the increasing sequence $s_1, s_2, \ldots, s_{\min\{2k+1,n\}}$ alternates between legal and illegal instances. Thus, #altern($\mathcal{L}_{n,k}$) $\geq \min\{2k+1, n\}$.

Let $s_1 \subset \ldots \subset s_x$ be a sequence of increasing instances such that for every $i, 1 \leq i < x$, $s_i \in \mathcal{L}_{n,k} \wedge s_{i+1} \notin \mathcal{L}_{n,k}$ or $s_i \notin \mathcal{L}_{n,k} \wedge s_{i+1} \in \mathcal{L}_{n,k}$. Let $D(i)$ denote the size of the set of values decided in s_i. That is, if $s_i = \{(id_1,(a_1,b_1)),\ldots,(id_\ell,(a_\ell,b_\ell))\}$, $D(i) = |\{b_1,\ldots,b_\ell\}|$. Note that $D(1) \geq 1$. Consider instances s_i, s_{i+1} with i such that with $s_i \in \mathcal{L}_{n,k}$. As $s_i \subset s_{i+1}$, and since in s_i no more that k values are decided and every decided value is valid, it follows that a value not decided in s_i is decided in s_{i+1}, i.e., $D(i) < D(i+1)$.

Let $s_{x'}$ be the largest legal instance in the sequence. Note that $x' = x$ or $x' = x-1$. On one hand, we have $D(x') \leq k$. On the other hand, $1 + \lfloor \frac{x'-1}{2} \rfloor \leq D(x')$, since the number of alternations from a legal instance to an illegal one in the sequence $s_1, \ldots, s_{x'}$ is at least $\lfloor \frac{x'-1}{2} \rfloor$ and $D(1) \geq 1$. Hence, $x' \leq 2k$, and as $x \leq x'+1$, we obtain $x \leq 2k+1$. Therefore, any sequence of increasing and alternating instances is of length at most $2k+1$. Hence, #altern($\mathcal{L}_{n,k}$) $\leq \min\{2k+1, n\}$. □

The two next sections focus on determining how many opinions are needed to monitor (n, k)-set agreement. We show that #opinion($\mathcal{L}_{n,k}$) $=$ #altern($\mathcal{L}_{n,k}$) if #altern($\mathcal{L}_{n,k}$) $< n$ and #opinion($\mathcal{L}_{n,k}$) $= n+1$ otherwise, that is, #opinion($\mathcal{L}_{n,k}$) $= \min\{2k, n\} + 1$.

[2]Language $\mathcal{L}$ is projection-closed if and only if for each $s \in \mathcal{L}$ and each $s' \subset s$, $s' \in \mathcal{L}$.

4 Monitoring k-set agreement

We prove in this section that (n,k)-set-agreement can be wait-free monitored using $\min\{2k,n\}+1$ opinions.

In [12], an *universal monitor* is presented that uses at most $n+1$ opinions for any n-processes language. The opinion-maker of this monitor depends on the language being checked whereas the interpretation depends solely on the opinions output by the processes. Independently of the language, the opinions produced by each process belongs to a set of size $n+1$. If $k > \lfloor \frac{n}{2} \rfloor$, $\min\{2k,n\}+1 = n+1$ and thus in this case the language $\mathcal{L}_{n,k}$ induced by (n,k)-set-agreement can be checked by the universal monitor described in [12].

For the case $k \leq \lfloor \frac{n}{2} \rfloor$, we present a monitor (M,μ) for (n,k)-set agreement that uses $2k+1$ opinions. The set of opinions is:

$$U = \{\text{red}\} \cup \{(\text{green},\ell) : 1 \leq \ell \leq k)\} \cup \{(\text{orange},\ell) : 1 \leq \ell \leq k)\}.$$

Note that $|U| = 2k+1$. The partition $(\mathbf{Y},\mathbf{N})$ is defined as follows. For any multiset S with values in U, let

$$\text{maxlevel}(S) = \max\{\ell : (\text{green},\ell) \in S \vee \ (\text{orange},\ell) \in S\}.$$

Then,

$$S \in \mathbf{Y} \iff (\text{green}, \text{maxlevel}(S)) \in S \wedge \text{red} \notin S \tag{1}$$

Equivalently, any multiset S that contains the value red or a pair (orange, ℓ) and no pair (green, ℓ') with $\ell \leq \ell'$ is in the "no" set $\mathbf{N}$.

Recall that each process p_i starts with an input-output pair (s_i,t_i), runs the opinion-maker protocol and produces an opinion based on what it sees during the execution of the protocol. Since we consider wait-free protocol and processes run asynchronously, it may be the case that a process p_i does not see the input-output pair of some participating processes.

Intuitively, a process outputs red if in the collection (s',t') of input-output pairs it sees does not fit the specification of (n,k)-set-agreement, and furthermore, this cannot be fixed by completing (s',t') with other input-output pairs. This occurs if agreement is broken, i.e., more than k values are decided in t'. In the case where agreement is satisfied (t' contains at most k distinct values) as well as validity (every decided values is proposed, that is, appears in s'), the process outputs (green,d) where $d \leq k$ is the number of decided values. In the last case, namely agreement is satisfied but validity is not (a decided value in t' does not appear in s'), (orange,d) is output. Note that in this case, the global input (s,t) consisting in the collection of input-output pairs of each participating process may still fit the specification of (n,k)-set agreement. The pseudo-code of the opinion-maker appears in Fig. 1.

```
Opinion-maker M at process i with input (i, (s_i, t_i)):
  R.update(i, (s_i, t_i));
  r ← R.snapshot();
  (s', t') ← ({(j, s_j) : r[j] ≠ ⊥}, {(j, t_j) : r[j] ≠ ⊥});
  let val(s') and val(t') denote the set of values in s' and t' respectively;
  if |val(t')| > k then decide red
  else if val(t') ⊆ val(s') then decide (green, |val(t')|)
       else decide (orange, |val(t')|)
```

Figure 1: Opinion-maker for (n,k)-set-agreement, $k \leq \lfloor \frac{n}{2} \rfloor$.

The following Lemma summarizes the result of this section:

Lemma 4.1. *(n,k)-set-agreement has a monitor that uses* $\min\{2k,n\}+1$ *opinions, i.e.,* $\#\text{opinion}(\mathcal{L}_{n,k}) \leq \min\{2k,n\}+1$.

Proof. We prove that, for $k \leq \lfloor \frac{n}{2} \rfloor$, the opinion-maker M of Fig. 1 together with the interpretation μ described in Equation 1 are a wait-free monitor for (n,k)-set agreement task $(\mathcal{I},\mathcal{O},\Delta)$. Note that the opinion set of M is of size $2k+1$, as desired.

Let V denote the set of possible input values for the (n,k)-set agreement task. Let us consider an execution e of the opinion-maker with input set $\{(id_1,(s_1,t_1)),\ldots,(id_\ell,(s_\ell,t_\ell))\}$ where $(s_i,t_i) \in V \times V$ for each $i, 1 \leq i \leq \ell$. Assume that every participating process decides, and let S denote the multiset of opinions that are decided in this execution. In addition, let $s = \{(id_1,s_1),\ldots,(id_\ell,s_\ell)\}$ and $t = \{(id_1,t_1),\ldots,(id_\ell,t_\ell)\}$.

A process p_i decides red if more than k values are decided in the output set t' it sees. Even if i has only a partial view of the input, i.e., $(s',t') \subsetneq (s,t)$, the lack of agreement cannot be fixed in (s,t) since every value decided in t' is also decided in t. Therefore, $t \notin \Delta(s)$, and consequently, it is correct that $S \in \mathbf{N}$.

Suppose that no processes decides red. Let p_i be the last process that writes its input (s_i,t_i) into shared memory. The invocation of $R.\mathsf{snapshot}()$ by p_i thus starts after every participating process p_j has updated the shared object with its input (s_j,t_j). It thus follows that p_i observes the full input (s,t). Denote by d the number of decided values in t. Since no process decides red, $d \leq k$. We consider two cases, depending on whether $t \in \Delta(s)$ or not.

- Assume first that $t \in \Delta(s)$. Since p_i sees the full output, it decides (green,d). As in every partial input $(s',t') \subseteq (s,t)$, the number of decided values is at most d, each process that sees a partial input decides either (green,d') or (orange,d') with $d' \leq d$. Therefore, $\text{maxlevel}(S) = d$, and thus $S \in \mathbf{Y}$.

- Assume now that $t \notin \Delta(s)$. Since no processes decide red, the number of decided values in t is at most k. Because $t \notin \Delta(s)$, it must be the case that a decided value is not valid, i.e., there exists a value $u \in \mathit{val}(t)$ that is not contained in $\mathit{val}(s)$. Therefore p_i decides (orange,d). Assume for contradiction that a process p_j decides (green,ℓ) with $\ell \geq d$. Observe that, at process j, the collection of input-output pairs (s',t') seen by j is such that $(s',t') \subseteq (s,t)$. Hence, every value decided in t' is also decided in t. Thus $\ell = |\mathit{val}(t')|$, the number of decided values seen by p_j, is smaller than of equal to d.

  Therefore $\ell = d$ and processes p_i and p_j see the same output set $W = \mathit{val}(t) = \mathit{val}(t')$ of decided values. Moreover, as p_j decides (green,ℓ), validity holds for the pair (s',t'), that is $W = \mathit{val}(t') \subseteq \mathit{val}(s')$. Consequently, as $s' \subseteq s$, each value $v \in W$ is also proposed in s. Therefore, agreement and validity are verified in (s,t) and p_i should decide (green,d): a contradiction. Hence, for every $(\text{green},\ell) \in S$, $\ell < d$ from which we conclude that $S \in \mathbf{N}$. □

5 Opinion Number of (n,k)-set agreement

We now establish a lower bound on the number of opinions required to monitor $\mathcal{L}_{n,k}$. The proof is by contradiction.

Assume for contradiction that $\mathcal{L}_{n,k}$ has a wait-free monitor (M,μ) with opinion set U, $|U| < \min\{2k,n\}+1$. The existence of this monitor implies that a specific task $T_{U,\mu}$ has a wait-free

protocol. When solving a task, each participating process starts with a private input value and has to eventually decide irrevocably on an output value. Let $\mathcal{V}, \mathcal{V} \supseteq \{0, \ldots, k\}$, the alphabet of the language $\mathcal{L}_{n,k}$. In task $T_{U,\mu}$, process p_i's input is any pair $(s_i, t_i) \in \mathcal{V} \times \mathcal{V}$ and it is required to output a value $u_i \in U$. When the input set s is a legal instance, i.e., $s \in \mathcal{L}_{n,k}$, the multiset of output value must belong to the "yes" set $\mathbf{Y}$ and in the "no" set $\mathbf{N}$ otherwise. We show that as long as the set of opinion U is of cardinality less than $\min\{2k, n\} + 1$, whatever the interpretation μ, task $T_{U,\mu}$ is not wait-free solvable. To that end, we rely on the representation of wait-free computations by topological objects, as in, e.g., [2, 14, 15]. Our main tool is a variant of Sperner's Lemma.

5.1 Preliminaries

5.1.1 Basic notions of topology

We first review some basic definitions. A *complex* $\mathcal{K}$ is a set of vertices $V(\mathcal{K})$, and a family of finite, nonempty subsets of $V(\mathcal{K})$, called *simplexes*, satisfying: (1) if $v \in V(\mathcal{K})$ then $\{v\}$ is a simplex, and (2) if s is a simplex, so is every nonempty subset of s. The *dimension* of a simplex s is $|s| - 1$, the dimension of $\mathcal{K}$ is the largest dimension of its simplexes, and $\mathcal{K}$ is *pure* of dimension k if every simplex belongs to a k-dimensional simplex. A simplex τ is a *face* of a simplex σ if τ is a subset of σ. If τ is not equal to σ then τ is a *proper face* of σ. The *complex induced by a simplex* σ consists in σ and all its faces.

In distributed computing, a vertex represents a local state, a simplex a global state and a complex a collection of global states. Hence, one of the labels of each vertex is an identity in $[n]$. We denote by $\mathrm{ID}(\sigma)$ the identities of the vertexes of σ. A simplex is *chromatic* if it is properly colored with ids in $[n]$. A complex is *chromatic* if each of its simplex is chromatic.

A *simplicial map* f from complex $\mathcal{K}$ to complex $\mathcal{L}$ is a function from $V(\mathcal{K})$ to $V(\mathcal{K})$ that preserves simplexes. That is, if $\tau = \{v_1, \ldots, v_\ell\}$ is a simplex of $\mathcal{K}$, then $\{f(v_1), \ldots, f(v_\ell)\}$ is a simplex of $\mathcal{L}$. In addition, if $\mathcal{K}$ and $\mathcal{L}$ are chromatic complexes, f is said to be *id-preserving* if for any simplex $\tau = \{v_1, \ldots, v_\ell\} \in \mathcal{K}$, $\mathrm{ID}(\tau) = \mathrm{ID}(\{f(v_1), \ldots, f(v_\ell)\})$.

5.1.2 Pseudomanifold and divided images

A complex $\mathcal{K}$ of dimension n is a *pseudomanifold with boundary* if it is strongly connected, and each $(n-1)$-simplex in $\mathcal{K}$ is a face of precisely one or two n-simplexes. For simplicity, a pseudomanifold with boundary will simply be called a *pseudomanifold*. We sometimes write n-pseudomanifold as a shorthand for a n-dimensional pseudomanifold. Let $\mathcal{K}$ be a n-pseudomanifold. A $(n-1)$-simplex σ is said to be *internal* if it is a face of exactly two n-simplexes of $\mathcal{K}$ and *external* otherwise. The boundary of $\mathcal{K}$, denoted $\partial\mathcal{K}$, is the sub-complex of $\mathcal{K}$ induced by its external simplexes. More precisely, $\partial\mathcal{K}$ consists in each $(n-1)$-simplex of $\mathcal{K}$ that is a face of exactly one n-simplex together with all its faces.

Divided images of complexes have been introduced in [2] as a combinatorial tool to represent certain classes of executions of read/write wait-free protocols. A subdivision of a complex is a divided image, but subdivided images are not always subdivisions. Divided images capture the essential properties to study wait-free computability.

Definition 5.1 ([2], Definition 4.1)**.** *Let $\mathcal{K}$ and $\mathcal{L}$ be finite n-dimensional complexes and ψ a function that maps every simplex of $\mathcal{K}$ to a subcomplex of $\mathcal{L}$. The complex $\mathcal{K}$ is a* divided image of $\mathcal{L}$ under ψ *if and only if*

1. $\psi(\emptyset) = \emptyset$,
2. *for every* 0*-simplex* $\sigma \in \mathcal{L}$, $\psi(\sigma)$ *is a single vertex,*
3. *for every* $\sigma, \sigma' \in \mathcal{L}$, $\psi(\sigma \cap \sigma') = \psi(\sigma) \cap \psi(\sigma')$, *and*
4. *for every* $\sigma \in \mathcal{L}$, $\psi(\sigma)$ *is a* $\dim(\sigma)$*-pseudomanifold with* $\partial\psi(\sigma) = \psi(\partial\sigma)$.

When ψ is clear from the context or not relevant, we simply say that $\mathcal{K}$ *is a subdivided image of* $\mathcal{L}$. Next Lemma state some properties of divided:

Lemma 5.1 ([2], Lemma 4.2)**.** *Let* $\mathcal{K}, \mathcal{L}$ *be* n*-dimensional complexes such that* $\mathcal{K}$ *is a divided image of* $\mathcal{L}$ *under* ψ.

1. *For every* $\sigma, \sigma' \in \mathcal{L}$, *if* $\sigma \subseteq \sigma'$, *then* $\psi(\sigma) \subseteq \psi(\sigma)$.
2. *For every pair of* j *simplexes* $\sigma, \sigma' \in \mathcal{K}$, *if* $\sigma \neq \sigma'$ *and* $\sigma \cap \sigma' \neq \emptyset$, *then* $\psi(\sigma \cap \sigma')$ *is a pseudomanifold of dimension strictly smaller than* j.
3. *A* $(n-1)$*-dimensional simplex* $\tau \in \mathcal{K}$ *is external if and only if for some external* $(n-1)$*-dimensionnal simplex* $\sigma \in \mathcal{L}$, $\tau \in \psi(\sigma)$.

The *carrier* of simplex $\tau \in \mathcal{K}$, denoted *carrier*(τ) is the simplex $\sigma \in \mathcal{L}$ of smallest dimension such that $\tau \in \psi(\sigma)$. Note that, by Definition 5.1(3), *carrier*(τ) is well defined and by Lemma 5.1(2), it is unique. If $\mathcal{L}$ is a chromatic complex, $\mathcal{K}$ is a *chromatic divided image* [2] of $\mathcal{L}$ if $\mathcal{K}$ is a divided image of $\mathcal{L}$, $\mathcal{K}$ is a *chromatic* complex, and for any $\tau \in \mathcal{K}$, $\mathrm{ID}(\tau) \subseteq \mathrm{ID}(\mathit{carrier}(\tau))$. Note in particular that if $\mathit{dim}(\tau) = \mathit{dim}(\mathit{carrier}(\tau))$, τ is properly colored with the ids in $\mathrm{ID}(\tau)$.

5.1.3 Combinatorial implication of wait-free computability

Simplexes and complexes are a convenient way to represent tasks and distributed protocol. A tasks $T = (\mathcal{I}, \mathcal{O}, \Delta)$ can be equivalently described by an input complex $\widetilde{\mathcal{I}}$, an output complex $\widetilde{\mathcal{O}}$ and a function $\widetilde{\Delta}$ that maps each simplex of $\widetilde{\mathcal{I}}$ to a sub-complex of $\widetilde{\mathcal{O}}$. Vertexes of $\widetilde{\mathcal{I}}$ and $\widetilde{\mathcal{O}}$ are labeled with an identity and a value and there is a simplex $s = \{(id_1, v_1), \ldots, (v_\ell, id_\ell)\}$ in $\widetilde{\mathcal{I}}$ (respectively, $\widetilde{\mathcal{O}}$) if and only if s is an input set in $\mathcal{I}$ (respectively, an output set in $\mathcal{O}$). Similarly, $t \in \widetilde{\Delta}(s)$ if and only if the corresponding output set t is in $\Delta(s)$. In the following, we consider the topological representation for tasks and drop the $\widetilde{\ }$ notation.

Without loss of generality, a read/write wait-free protocol consists in a certain number B of (asynchronous) rounds. In each round, process p_i writes its state in its cell $R[i]$, takes a snapshot of the memory and updates its state. The process initial state is its input. At the end of the B rounds, a final state is reached on which depends the process decision. A *protocol complex* $\mathcal{P}$ represents all possible final states for some execution. Each vertex is labeled with an *id* and a possible final state. $\sigma = \{(id_1, v_1), \ldots, (id_\ell, v_\ell)\}$ is a simplex in $\mathcal{P}$ if there is an execution at the end of which process p_i with identity id_i is in state v_i, for $1 \leq i \leq \ell$.

An *immediate snapshot execution* can be divided into blocks. In each block, a subset of the participating processes are active. They first simultaneously write before taking a snapshot. One important result of the topological approach is a characterization of the structural properties of the protocol complex of immediate snapshot executions, namely the immediate snapshot protocol complex is a chromatic divided image of the input complex [2]. If a protocol solves a task $T =$

$(\mathcal{I}, \mathcal{O}, \Delta)$, in any execution, the final states can be mapped to decision values in such a way that the output set is allowed for the input set of the execution according to Δ. As considering only a subset of all possible executions might be sufficient to derive an impossibility result, we obtain the following necessary condition for read/write wait-free solvability of tasks:

Theorem 5.1 ([2], Theorem 5.10)**.** *Let $T = (\mathcal{I}, \mathcal{O}, \Delta)$ a task. If there is a read/write wait-free protocol which solves T, then there is a chromatic divided image $\mathcal{I}^*$ of $\mathcal{I}$ and a id-preserving simplicial map δ from $\mathcal{I}^*$ to $\mathcal{O}$ that agrees with Δ.*

5.1.4 A variant of Sperner's Lemma

Given a function $f : V(\mathcal{K}) \longrightarrow U$, for each $\sigma = \{v_0, \ldots, v_\ell\}$ simplex of $\mathcal{K}$, $f(\sigma)$ denote the set of labels of the vertexes of σ by f, i.e., $f(\sigma) = \{f(v_0), \ldots, f(v_\ell)\}$. The proof can be found in appendix A.

Lemma 5.2. *Let $\mathcal{K}$ be a n-pseudomanifold, let U be a set of at least $n+1$ elements and let $f : V(\mathcal{K}) \to U$. If for some subset $B \subset U$ of size n, there is an odd number of B-labeled $(n-1)$-simplexes in the boundary of $\mathcal{K}$, then there is an odd number of C-labeled n-simplexes in $\mathcal{K}$, for some set C of $n+1$ elements, $B \subset C \subseteq U$:*

$$\Big|\{\sigma \in \partial\mathcal{K} : dim(\sigma) = n-1 \text{ and } f(\sigma) = B\}\Big| \text{ is odd } \implies \exists\, C, |C| = n+1, \Big|\{\sigma \in \mathcal{K} : dim(\sigma) = n \text{ and } f(\sigma) = C\}\Big| \text{ is odd .}$$

5.2 The lower bound

Lemma 5.3. *Let M be a wait-free opinion-maker with opinion set U and μ be an interpretation over U. If (M, μ) is a monitor for (n,k)-set-agreement, $|U| \geq \min\{2k, n\} + 1$.*

Proof. Recall that in the (n,k)-set agreement task, each process starts with a value from some set $\mathcal{V}$ and is required to decide an initial value of some process such that no more than k distinct values are decided. For the proof, we assume without loss of generality that $\mathcal{V} = \{0, \ldots, k\}$.

The proof is by contradiction. We assume that there exists a monitor (M, μ) with opinion set $U, |U| < \min\{n, 2k\}+1$. The opinion-maker M is a wait-free protocol. In any of its executions, each participating process i starts with a pair $(s_i, t_i) \in \mathcal{V} \times \mathcal{V}$ and is directed to decided a value $u_i \in U$. The interpretation μ defines a partition $(\mathbf{Y}, \mathbf{N})$ of all the multisets with values in U; the multiset of decided values belongs to $\mathbf{Y}$ if and only if the set of initial pairs (s_i, t_i) verify the specification of (n,k)-set agreement.

M is therefore a wait-free protocol for the task $T = (\mathcal{I}, \mathcal{U}, \Delta_\mu)$ where $\mathcal{I} = \mathit{complex}(\mathcal{V} \times \mathcal{V}, n)$, $\mathcal{U} = \mathit{complex}(U, n)$. Given any finite set X, $\mathit{complex}(X, n)$ is the $(n-1)$-dimensional *pseudosphere* [14] complex induced by X: for each $i \in [n]$ and each $x \in X$, there is a vertex labeled (i, x) in the vertex set of $\mathit{complex}(X, n)$ and $s = \{(id_1, x_1), \ldots, (id_\ell, x_\ell)\}$ is a simplex of $\mathit{complex}(X, n)$ if and only if $\{x_1, \ldots, x_\ell\} \subseteq X$ and s is properly colored with identities. The relation Δ_μ is defined next.

Each simplex $s = \{v_1, \ldots, v_\ell\}$ of $\mathcal{I}$ represents the initial and decided values of the participating processes in some execution of a (n,k)-set agreement protocol. In more details, each vertex v_j has two labels: an id id_j, a pair $(s_i, t_i) \in \mathcal{V} \times \mathcal{V}$ representing the process input and output value for the (n,k)-set agreement tasks. Let us define two predicates, $\text{agree}_k(s)$ and $\text{valid}(s)$, that are verified

if the set of input-output pairs (s_i, t_i) in s satisfy the agreement property and validity property, respectively, of (n,k)-set agreement. That is,

$$\begin{aligned} \text{agree}_k(s) &\iff |\{t_j : (id_j, (s_j, t_j)) \in s\}| \leq k \\ \text{valid}(s) &\iff \{t_j : (id_j, (s_j, t_j)) \in s\} \subseteq \{s_j : (id_j, (s_j, t_j)) \in s\} \end{aligned}$$

A vertex of a simplex $t \in \mathcal{U}$ has two labels: and identity id and an opinion $u \in U$. Recall that $\text{ID}(t)$ and $mval(t)$ then denote the set of ids and the multiset of opinions, respectively, of the vertexes of t. Δ_μ is then defined as follows. For any $s \in \mathcal{I}$, $t \in \mathcal{U}$,

$$t \in \Delta_\mu(s) \iff \text{ID}(t) = \text{ID}(s) \text{ and } \begin{cases} mval(t) \in \mathbf{Y} & \text{if agree}_k(s) \text{ and valid}(s) \\ mval(t) \in \mathbf{N} & \text{otherwise.} \end{cases}$$

We associate with each simplex $s \in \mathcal{I}$ a value in $\{+1, -1\}$ depending on whether agreement and validity hold in s:

$$\forall s \in \mathcal{I} : sign(s) = \begin{cases} +1 & \text{if agree}_k(s) \text{ and valid}(s) \\ -1 & \text{otherwise.} \end{cases}$$

We focus on the following simplexes of $\mathcal{I}$ (See also Figure 2):

$$\sigma_{2i} = \{(1,(0,1)), (2,(1,1)), (3,(1,2)), (4,(2,2)), \ldots, (2i-1,(i-1,i)), (2i,(i,i))\}, \text{ for } i \geq 1,$$

$$\sigma_{2i+1} = \{(1,(0,1)), (2,(1,1)), (3,(1,2)), (4,(2,2)), \ldots, (2i,(i,i)), (2i+1,(i,i+1))\}, \text{ for } i \geq 0.$$

Note that $\sigma_1 \subset \sigma_2 \subset \ldots \subset \sigma_n$ and that for each simplex σ_i, the complex induced is a $(i-1)$-pseudomanifold. Furthermore, we have

$$\begin{cases} \text{valid}(\sigma_i) \text{ and agree}_k(\sigma_i) & \text{if } 1 \leq i \leq 2k \text{ and } i \text{ is even} \\ \neg\text{valid}(\sigma_i) \text{ and agree}_k(\sigma_i) & \text{if } 1 \leq i \leq 2k \text{ and } i \text{ is odd} \\ \neg\text{agree}_k(\sigma_i) & \text{if } 2k < i \end{cases}$$

Hence, $sign(\sigma_i) = (-1)^i$ if $1 \leq i \leq 2k+1$ and $sign(\sigma_i) = -1$ for $2k+1 \leq i$. Another noteworthy property is the fact that every $(dim(\sigma_i)-1)$-dimensional face of σ_i except σ_{i-1} has the same sign as σ_i:

$$\forall i \in [2, 2k+1], \forall \sigma \subset \sigma_i, dim(\sigma) = dim(\sigma_i) - 1 \ \wedge \ \sigma \neq \sigma_{i-1} \implies sign(\sigma) = sign(\sigma_i) \qquad (2)$$

Since there is a wait-free protocol for the task $T = (\mathcal{I}, \mathcal{U}, \Delta_\mu)$, namely the opinion-maker M, it follows from Theorem 5.1 that there is a chromatic divided image $\mathcal{I}^*$ of $\mathcal{I}$ under a function ψ and a id-preserving simplicial map $\delta : \mathcal{I}^* \to \mathcal{U}$ that agrees with Δ_μ. Since δ is a simplicial map, it maps each vertex $v \in V(\mathcal{I}^*)$ to a vertex $\delta(v)$ in $\mathcal{U}$. Recall that each vertex of $\mathcal{U}$ has two labels: a process id $\in [n]$ and an opinion $u \in U$. δ thus implies a (not necessarily proper) coloring $c : V(\mathcal{I}^*) \to U$ on the vertexes of $\mathcal{I}^*$: For each vertex v of $\mathcal{I}^*$, $c(v) = val(\delta(v))$. Given a simplex $s = \{v_1, \ldots, v_\ell\} \in \mathcal{I}^*$ we denote by slight abuse of notation $c(s) = \{c(v_1), \ldots, c(v_\ell)\} \subseteq U$ the *multiset* of colors induced by δ of the vertex of s. Figure 3 represents a chromatic divided image of the input simplex σ_3.

Claim 5.1. *Let σ, σ' be j-simplexes in $\mathcal{I}$ with $sign(\sigma) = -sign(\sigma')$ and let s, s' be j-simplexes in $\mathcal{I}^*$. If $s \in \psi(\sigma)$ and $s' \in \psi(\sigma')$, $c(s) \neq c(s')$.*

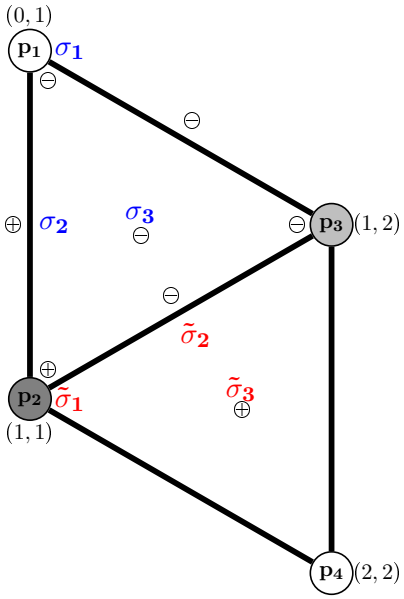

Figure 2: The simplexes $\sigma_1 \subset \sigma_2 \subset \sigma_3$, and $\widetilde{\sigma}_1 \subset \widetilde{\sigma}_2 \subset \widetilde{\sigma}_3$ and their sign.

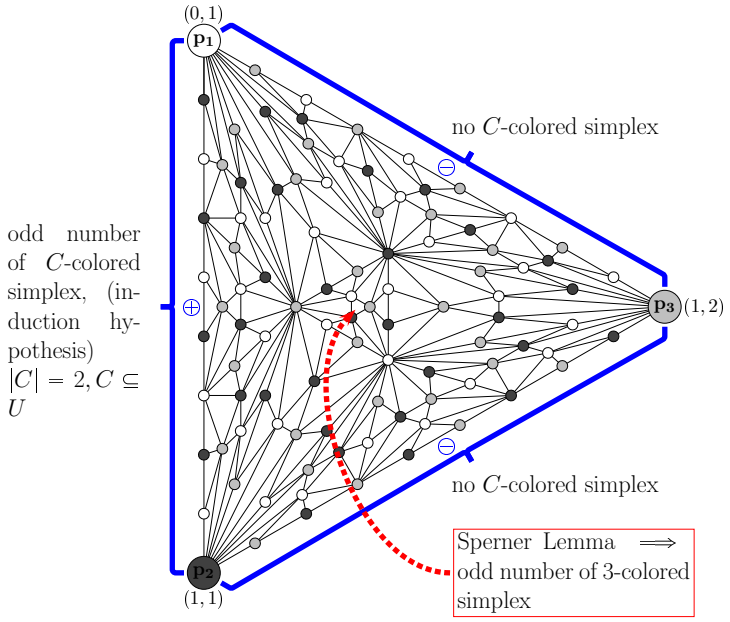

Figure 3: Proof strategy.

Proof of Claim 5.1. $c(s)$ is the multiset of opinions output by the participating processes (those processes with ids $\mathrm{ID}(s) = \mathrm{ID}(\sigma)$) in some execution e of the opinion-maker M in which the initial configuration is σ. Similarly, in some execution e' of M with initial configuration σ', the opinions collectively output by the participating processes is $c(s')$. As $sign(\sigma) = -sign(\sigma')$, the validity and agreement properties of (n,k)-set agreement in one initial configuration are satisfied but this is not the case in the other execution. Therefore the same multiset of opinion cannot be output in both executions. For example, in Figure 3, for any 1-dimensional simplex s in the subdivided left edge $\{(p_1,(0,1)),(p_2,(1,1))\}$ and any s' in the subdivided top right edge $\{(p_1,(0,1)),(p_3,(1,2))\}$, $c(s) \neq c(s')$. □

Next, we show that for each $j, 1 \leq j \leq \min\{2k+1, n\}$, there exists at least one execution of the opinion-maker where the input configuration is σ_j in which the j participating processes output j distinct opinions. More precisely, we establish by induction on j the following claim. The proof strategy is illustrated in Figure 3.

Claim 5.2. *For each $j, 1 \leq j \leq \min\{2k+1, n\}$, there exists a set $C_j \subseteq U$ of size j such that $\left|\{s \in \psi(\sigma_j) : c(s) = C_j\}\right|$ is odd*

Proof of Claim 5.2.

- Base case $j = 1$. By definition, σ_1 is a vertex. By Definition 5.1(2), $\psi(\sigma_1)$ is also a vertex v of $\mathcal{I}^*$. Since δ is a simplicial map, $\delta(\psi(\sigma_1))$ is a vertex in $\mathcal{U}$. Let $C_0 = val(\delta(\psi(\sigma_1)))$, i.e., $C_0 = \{c(v)\}$. That is, C_0 is the singleton consisting in the opinion output by process 1 when it runs alone with input/output pair $(0,1)$.
- Induction step. Let $j \geq 2$ and assume that the claim is true for $j-1$.

By Definition 5.1(4), $\psi(\sigma_j)$ is a $(j-1)$-pseudomanifold. In order to apply our variant of Sperner's Lemma, we establish that the number of $(j-2)$-simplexes in the boundary of $\psi(\sigma_j)$ that are colored with C_{j-1} is odd. To that end, let $s \in \partial\psi(\sigma_j)$ be an $(dim(\sigma_j)-1)$ dimensional simplex and assume that $c(s) = C_{j-1}$. We prove that $s \in \psi(\sigma_{j-1})$.

Note that $dim(s) = j-2$. By Lemma 5.1(3), there exist a $(j-2)$ dimensional face σ of σ_j such that $s \in \psi(\sigma)$. Suppose for contradiction that $\sigma \neq \sigma_j$. It then follows that $sign(\sigma) = sign(\sigma_j)$ (Equation (2) from which we have $sign(\sigma) = -sign(\sigma_{j-1})$. By the induction hypothesis, there is a least one $(j-2)$-simplex $s' \in \psi(\sigma_{j-1})$ with $c(s') = C_{j-1}$. Since $sign(\sigma) = -sign(\sigma_{j-1})$, it follows from Claim 5.1 that no $(j-2)$-simplex in $\psi(\sigma)$ is C_{j-1}-colored. In particular, $c(s) \neq C_{j-1}$: a contradiction.

Therefore, the only simplexes $s \in \partial\psi(\sigma_j)$ that are C_{j-1}-colored are the simplexes s in $\psi(\sigma_{j-1})$ such that $c(s) = C_{j-1}$. By the induction hypothesis, the number of such simplex is odd. It thus follows from Lemma 5.2 that there exists a set $C = C_j \supset C_{j-1} \supseteq U$ of j elements such that the number of $(j-1)$-simplexes $s \in \psi(\sigma_j)$ colored with C_j is odd.

If $\min\{2k+1, n\} = 2k+1$, we can stop here: we have just proved that $\psi(\sigma_j) \subseteq \mathcal{I}^*$ contains at least one $2k$-dimensional simplex colored with $2k+1$ distinct colors. Hence $|U| \geq 2k+1 = \min\{2k, n\} + 1$, as desired.

Suppose that $n < 2k+1$. A similar reasoning can be carried over the collection of simplexes $\widetilde{\sigma}_1 \subset \ldots \subset \widetilde{\sigma}_n$ defined as follows (See also Figure 2):

$$\widetilde{\sigma}_{2i-1} = \{(2,(1,1)),(3,(1,2)),(4,(2,2)),\ldots,(2i-1,((i-1),i)),(2i,(i,i))\},\ \text{for } 1 \leq i \leq \left\lfloor \frac{n}{2} \right\rfloor,$$

$$\widetilde{\sigma}_{2i} = \{(2,(1,1)),(3,(1,2)),(4,(2,2)),\ldots,(2i,(i,i)),(2i+1,(i,i+1))\},\ \text{for } 1 \leq i \leq \left\lfloor \frac{n-1}{2} \right\rfloor,$$

$$\widetilde{\sigma}_n = \begin{cases} \{(2,1,1),(3,1,2),(4,2,2),(n,\frac{n}{2},\frac{n}{2}),\ldots,(1,\frac{n}{2},\frac{n}{2}+1)\} & \text{if } n \text{ is even} \\ \{(2,1,1),(3,1,2),(4,2,2),(n,\frac{n-1}{2},\frac{n-1}{2}+1),\ldots,(1,\frac{n-1}{2}+1,\frac{n-1}{2}+1)\} & \text{if } n \text{ is odd} \end{cases}$$

Note that $sign(\widetilde{\sigma}_i) = (-1)^{i+1}$ and if $\widetilde{\sigma}$ is a $(dim(\widetilde{\sigma}_i)-1)$ face of $\widetilde{\sigma}_i$ different from $\widetilde{\sigma}_{i-1}$, then $sign(\widetilde{\sigma}) = sign(\widetilde{\sigma}_i)$.

We have established above (Claim 5.2) that $\psi(\sigma_n)$ contains at least one $(n-1)$-dimensional simplex s with $c(s) = C_n$, where C_n is the a set of n opinions. By applying the same reasoning as in Claim 5.2, considering simplexes $\widetilde{\sigma}_1, \ldots, \widetilde{\sigma}_n$ instead, one can show that $\psi(\widetilde{\sigma}_n)$ contains a $(n-1)$-dimensional simplex $\widetilde{s}$ colored with a set $\widetilde{C}_n \subseteq U$ of size n. Since $sign(\widetilde{\sigma}_n) = (-1)^{n+1} = -sign(\sigma_n)$, $\widetilde{C}_n \neq C_n$ (Claim 5.1), from which we conclude that $|U| \geq n+1$.

6 Conclusion

We have investigated the minimum number of opinions needed to monitor (n,k)-set agreement, or equivalently, to monitor language $\mathcal{L}_{n,k}$. We have shown that this number is related to the alternation number as follows: $\#\text{opinion}(\mathcal{L}_{n,k}) = \#\text{altern}(\mathcal{L}_{n,k}) = 2k+1$ if $\#\text{altern}(\mathcal{L}_{n,k}) < n$ and $n+1$ otherwise.

Several problems remain open for future research including:

1. *Non-id-oblivious interpretation.* In our settings, the interpretation is id-oblivious in the sense that it does not take into account for each opinion the ids of the processes that output this opinion. We do not know if taking into account the identities could help reducing the number of opinions.

2. *A general lower bound.* In the case of (n, k)-set agreement, the alternation number is a lower bound on the minimum number of opinions. On the other hand, [12] shows that any language with opinion number α can be monitored with $O(\alpha)$ opinions. It would be interesting to generalize the lower bound to other languages or to find a language that can be monitored with strictly fewer opinions than its alternation number.

3. *Generalization to long-lived computation.* In our setting, one output per process is produced the distributed monitor must check that the set of outputs produced is correct. It would of course be interesting to extend our results to the case where a sequence of outputs is produced.

References

[1] Afek Y., Attiya H., Dolev D., Gafni E., Merritt M., and Shavit N., Atomic snapshots of shared memory. *J. ACM*, 40(4):873–890, 1993.

[2] Attiya H. and Rajsbaum S., The Combinatorial Structure of Wait-free Solvable Tasks. *SIAM Journal of Computing,* 31(4): 1286–1313 (2002).

[3] Awerbuch B., Patt-Shamir B. and Varghese G., Self-stabilization by Local Checking and Correction. *32nd Annual Symposium on Foundations of Computer Science* (FOCS), pp. 268–277, 1991.

[4] Bauer A. and Falcone Y., Decentralised LTL monitoring. *Formal Methods*, lncs #7436, pp. 85–100. Springer, 2012.

[5] Bauer A., Leucker M., and Schallhart C., Comparing LTL semantics for runtime verification. *J. Log. and Comput.*, 20(3):651–674, 2010.

[6] Berkovich S., Bonakdarpour B., and Fischmeister S., Gpu-based runtime verification. *IPDPS*, pp. 1025–1036. IEEE, 2013.

[7] Chandy M. and Lamport L., Distributed Snapshots: Determining Global States of Distributed Systems. *ACM Trans. Comput. Syst.*, 3(1):63–75. ACM, 1985.

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

[9] Fischer M., Lynch N. and Paterson M., Impossibility of Distributed Consensus with One Faulty Process. *J. ACM*, 32(2):374–382, 1985.

[10] Fraigniaud P., Korman A. and Peleg D., Local Distributed Decision. *52nd Annual IEEE Symposium on Foundations of Computer Science* (FOCS), pp 708-717, 2011.

[11] Fraigniaud P., Rajsbaum S., and Travers C., Locality and checkability in wait-free computing. *Distributed Computing*, 26(4):223–242, 2013.

[12] Fraigniaud P., Rajsbaum S., and Travers C., On the Number of Opinions Needed for Fault-Tolerant Run-Time Monitoring in Distributed Systems. *14th International Conference on Runtime Verification* (RV), lncs #8734, pp. 92–107. Springer, 2014.

[13] Henle M., *A Combinatorial Introduction to Topology.* Dover 1994.

[14] Herlihy M., Kozlov D., and Rajsbaum S., *Distributed Computing Through Combinatorial Topology.* Morgan Kaufmann-Elsevier, 2013.

[15] Herlihy M. and Shavit N., The Topological Structure of Asynchronous Computability. *J. ACM*, 46(6):858–923, 1999.

[16] Raynal M., *Concurrent Programming - Algorithms, Principles, and Foundations.* Springer, 2013.

[17] Sen K., Vardhan A., Agha G., and Rosu G., Decentralized runtime analysis of multithreaded applications. *IPDPS.* IEEE, 2006.

A A variant of Sperner's Lemma

Lemma 5.2. *Let $\mathcal{K}$ be a n-pseudomanifold, let U be a set of at least $n+1$ elements and let $f : V(\mathcal{K}) \to U$ be a function that labels each vertex of $\mathcal{K}$ with an elements of U.*

Suppose that, for some subset $B \subset U$ of size n, the number of B-labeled $(n-1)$-simplexes in the boundary of $\mathcal{K}$ is odd. Then exists a set C of $n+1$ elements, $B \subset C \subseteq U$ such that the number of C-labeled n-simplexes of $\mathcal{K}$ is odd. That is,

$$\Big|\{\sigma \in \partial\mathcal{K} : dim(\sigma) = n-1 \text{ and } f(\sigma) = B\}\Big| \text{ is odd } \implies$$
$$\exists\, C, |C| = n+1, \Big|\{\sigma \in \mathcal{K} : dim(\sigma) = n \text{ and } f(\sigma) = C\}\Big| \text{ is odd .}$$

Proof. The proof follows the proof of Sperner's Lemma [13], we construct a graph $G = (V, E)$. We associate a vertex denoted v_σ to each n-simplex $\sigma \in \mathcal{K}$. In addition V contains an "external" vertex u which is not associated with any simplex of $\mathcal{K}$. That is,

$$V = \{v_\sigma, \sigma \in \mathcal{K} \text{ and } dim(\sigma) = n\} \cup \{u\} .$$

For every pair σ, σ' of n-simplexes of $\mathcal{K}$, there is an edge between v_σ and $v_{\sigma'}$ if and only if σ and σ' share a $(n-1)$-dimensional face and this face is labeled B, i.e.,

$$\{v_\sigma, v_{\sigma'}\} \in E \iff dim(\sigma \cap \sigma') = n-1 \text{ and } f(\sigma \cap \sigma') = B .$$

Similarly, for every n-dimensional simplex $\sigma \in \mathcal{K}$, there is an edge between the external vertex u and v_σ if and only if an σ has an $(n-1)$-dimensional face in the boundary of $\mathcal{K}$ labeled B:

$$\{u, v_\sigma\} \in E \iff \exists \tau \subset \sigma, \tau \in \partial\mathcal{K}, dim(\tau) = n-1 \text{ and } f(\tau) = B .$$

Let σ be a n-simplex of $\mathcal{K}$. It follows from the definition of the graph G and the fact that $\mathcal{K}$ is an n-pseudomanifold that:

$$deg(v_\sigma) = \begin{cases} 1 & \text{if } B \subsetneq f(\sigma) ; \\ 2 & \text{if } B = f(\sigma) ; \\ 0 & \text{otherwise .} \end{cases}$$

Indeed, in the first case σ has exactly one $(n-1)$-dimensional face τ labeled B. This face is either in the boundary of $\mathcal{K}$, and thus v_σ share and edge with the external vertex u, or is in the interior of $\mathcal{K}$. In the former case, since $\mathcal{K}$ is a pseudomanifold, τ is a face of exactly one n-simplex $\sigma' \neq \sigma$, and hence $\{v_\sigma, v_{\sigma'}\}$ is an edge of G. If $f(\sigma) = B$, exactly two $(n-1)$-dimensional faces of σ are labeled B. By the same reasoning, it follows that $deg(v_\sigma) = 2$.

The edges of G with endpoint u are induced by the $(n-1)$-dimensional simplexes of $\partial\mathcal{K}$ that are labeled B. Indeed, as $\mathcal{K}$ is a n-pseudomanifold, a $(n-1)$ simplex $\tau \in \partial\mathcal{K}$ is a face of exactly one n-simplex σ of $\mathcal{K}$. Thus, if $f(\tau) = B$, there is an edge between u and v_σ. Therefore, the degree of u is the number of B-labeled $(n-1)$-simplexes of $\partial\mathcal{K}$. By assumption there is a odd number of such simplexes:

$$deg(u) \text{ is odd .}$$

Summarizing, we have shown that

$$\sum_{v \in V} deg(v) = deg(u) + \\ \left|\{\sigma \in \mathcal{K}, dim(\sigma) = n \text{ and } B \subsetneq f(\sigma)\}\right| + 2\left|\{\sigma \in \mathcal{K}, dim(\sigma) = n \text{ and } B = f(\sigma)\}\right| .$$

Since $\sum_{v \in V} deg(v)$ is even, and $deg(u)$ is odd, there must exist an odd number of n-simplexes whose set of labels strictly contains B. Therefore, for some $x \in U \setminus B$, the number of n-simplexes $\sigma \in \mathcal{K}$ such that $f(\sigma) = B \cup \{x\}$ is odd. □