# A non-topological proof for the impossibility of $k$-set agreement

Attiya Hagit, Armando Castaneda

# A non-topological proof for the impossibility of $k$-set agreement

Hagit Attiya[1] and Armando Castañeda[2]

[1] Department of Computer Science, Technion, Haifa 32000, Israel.
hagit@cs.technion.ac.il
[2] IRISA-INRIA, Campus de Beaulieu, 35042 Rennes Cedex, France.
armando.castaneda@inria.fr

**Abstract.** In the *$k$-set agreement* task each process proposes a value, and it is required that each correct process has to decide a value which was proposed and at most $k$ distinct values must be decided. Using topological arguments it has been proved that $k$-set agreement is unsolvable in the asynchronous *wait-free* read/write shared memory model, when $k < n$, the number of processes.
This paper presents a simple, non-topological impossibility proof of $k$-set agreement. The proof depends on two simple properties of the *immediate snapshot executions*, a subset of all possible executions, and on the well known graph theory result stating that every graph has an even number of vertices with odd degree (the *handshaking lemma*).
**Keywords:** Set agreement, Shared memory, Wait-freedom.

## 1 Introduction

In a breakthrough result, Fischer, Lynch and Paterson proved [7] that it is impossible to solve consensus in the asynchronous message passing system in which at most one process, which is unknown in advance, can crash. Herlihy [11] and Loui and Abu-Amara [17] extended this impossibility result to the asynchronous wait-free read/write shared memory model, where *wait-free* means that in each execution all processes but one can fail by crashing. Recall that in the *consensus* task, each process proposes a value, and it is required that every correct process decides on a value proposed by some process and no two correct processes decide distinct values.

Later, in order to study the border between solvable and unsolvable tasks in presence of asynchrony and failures, Chaudhuri [6] introduced a natural generalization of consensus, called *$k$-set agreement*; in this task, each process proposes a value and it is required that each correct process decides on a value proposed by a process and at most $k \geq 1$ distinct values are decided. For $k = 1$, $k$-set agreement is exactly consensus, and for $k = n$, the number of processes in the system, $k$-set agreement is trivial, since every process can decide on its proposal. The paper shows that $k$-set agreement can be solved by a $t$-resilient asynchronous algorithm, when $t < k$. An algorithm is *$t$-resilient*, $1 \leq t \leq n - 1$, if it solves the problem even in executions where up to $t$ processes crash. This means that

if the number of failures is strictly smaller than the number of possible decision values, then $k$-set agreement is solvable. Chaudhuri [6] also conjectured that $k$-set agreement is unsolvable if $t \geq k$. For the case $k = 1$, this conjecture matches the impossibility of solving consensus, namely, 1-set agreement, with a single crash failure [11, 17]. Notice that for the the wait-free case, i.e., $t = n - 1$, this conjecture says that only the trivial $n$-set agreement task has a wait-free solution.

Chaudhuri's conjecture was proved by Borowsky and Gafni [3], Herlihy and Shavit [15] and Saks and Zaharoglou [18]. Indeed, these three papers discovered a strong relationship between distributed computing and topology and used this topological approach for proving the conjecture of Chaudhuri. Later, Attiya and Rajsbaum [1] and Herlihy and Rajsbaum [13] presented two other topological impossibility proofs of $k$-set agreement. Although these proofs are not extremely complicated, they use concepts and results that are not widely known by the distributed computing community; the kind of arguments they use vary from combinatorial and algebraic to continuous arguments.

This paper presents a simple, non-topological impossibility proof for $k$-set agreement. This proof does not demand from the reader any previous knowledge of topology at all. Very roughly, the proof considers the *immediate snapshot (IS) executions* [1, 3, 4, 18] of an algorithm, a subset of all possible executions, and constructs a graph, whose vertices are the IS executions, and whose edges connect two IS executions (vertices) only if they satisfy certain indistinguishability conditions. Then, using the well known *handshaking lemma*, stating that every graph has an even number of vertices with odd degree, the proof concludes that there must be at least one execution in which at least $k + 1$ values are decided.

We believe that it is valuable to have several proofs of this important result, since they provide different perspectives on it. In particular, the proof we present in this paper gives an operational insight into the impossibility of set agreement. Such a perspective cannot be easily obtained from the known topological impossibility proofs for $k$-set agreement.

We stress that this paper does not argue that the use of topology for proving the impossibility of $k$-set agreement is "artificial", namely, that topology has been used before as a "trick" for proving the result. The reader has not to understand this paper in that way. The fundamental reason why $k$-set agreement is not solvable is topological, and more precisely, it has to do with Sperner's lemma [10, p. 36]. We strongly encourage the reader to see the connections between the proof we present here and Sperner's lemma. Moreover, we believe that the topological approach to distributed computing must be studied and extended, since it has been extensively and successfully used in the past for proving a variety of results (see, for example, [5, 8, 9, 12, 14, 16]), and it is unlikely that all these results have non-topological proofs.

The paper is organized as follows. Section 2 describes the asynchronous wait-free read/write shared memory model. Section 3 presents the subset of IS executions, which is used for proving the impossibility of $k$-set agreement in Section 4. Finally, Section 5 provides an operational perspective of this proof.

## 2 Model of computation

This section describes the standard asynchronous wait-free read/write shared memory model, considered in this paper, following [1, 2].

*System and executions.* A *system* consists of $n$ asynchronous sequential processes $p_1, \ldots, p_n$. Each process is a deterministic state machine with a (possibly infinite) set of *local states* $S$ and two subsets of $S$ called *initial states* and *output states*, respectively. The processes communicate by using a shared memory with a finite number of *single-writer multi-reader atomic registers*. No assumption is made regarding the size of the registers, thus we can assume that process $p_i$ has a single register $r_i$ to which it can write its entire state. Process $p_i$ has two atomic operations available to it: $write_i(v)$ that writes the value $v$ into $r_i$, and $read_i(j)$ that returns the current value in $r_j$. A *step* is performed by a single process $p_i$, which executes one of its two available operations, $read_i$ or $write_i$, performs some local computation and then changes its local state.

A *configuration* of the system consists of the local states of the processes and the content of the registers. An *initial configuration* is a configuration in which all local states are initial states and all registers are set to a distinguished value $\perp$. An *output configuration* is a configuration in which all local states are output states.

An *execution* of the system is a, possibly infinite, alternating sequence of configurations and steps $\alpha = C_0, s_0, C_1, s_1, C_2, \ldots$, where $C_0$ is an initial configuration and $C_{\ell+1}$ is the result of applying the step $s_\ell$ to $C_\ell$, for $\ell \geq 0$. The *view* of a process $p_i$ in $\alpha$, denoted $\alpha|p_i$, is the sequence of $p_i$'s local states in configurations $C_0, C_1, \ldots$. The *participating set* of an execution $\alpha$, denoted $\mathrm{ps}(\alpha)$, is the set of processes that take at least one step in the execution.

Two executions $\alpha$ and $\alpha'$ are *indistinguishable* for a set of processes $P$, denoted $\alpha \overset{P}{\sim} \alpha'$, if all processes in $P$ have the same view in both executions, namely, $\forall p_i \in P, \alpha|p_i = \alpha'|p_i$. In the following section we are interested in pairs of executions $\alpha$ and $\alpha'$ that are distinguishable to exactly one process, that is, there is a process $p_i$ such that $\alpha|p_i \neq \alpha'|p_i$ and $\alpha \overset{P}{\sim} \alpha'$, where $P = \{p_1, \ldots, p_n\} \setminus \{p_i\}$. In this case, we write $\alpha \overset{\neg p_i}{\sim} \alpha'$.

*Algorithms.* The state machine of a process $p_i$ models a *local algorithm* $A_i$ that determines $p_i$'s next step. An *algorithm* $A$ is a collection of local algorithms $A_1, \ldots, A_n$.

Each process has two distinguished components, *input* and *output*, that allow the system to model decision tasks. The input component never changes and cannot contain the distinguished value $\perp$. The output component contains initially $\perp$, and once a process reaches a local state in which a non-$\perp$ value is written in it, the output component never changes. When that happens, we say that the process *decides*. The output states are the states with non-$\perp$ output values. If a process decides $v$ in an execution $\alpha$, we say $v$ is *decided* in $\alpha$.

A view of a process $p_i$ in a finite execution $\alpha$ is *final*, if $p_i$ decides in $\alpha$. A final view of $p_i$ in $\alpha$ is *minimal* if none of its prefixes is final. In other words, the minimal final view of $p_i$ in $\alpha$ is the prefix of the view of $p_i$ up to the first configuration in which $p_i$ decides. A finite execution $\alpha$ is *minimal final* if the view of each process in $\text{ps}(\alpha)$ is a minimal final; in particular, each participating process decides in $\alpha$. For a minimal final execution $\alpha$, $\text{dec}(\alpha)$ is the set of all the values that are decided in $\alpha$, and for a process $p_i \in \text{ps}(\alpha)$, $\text{dec}(\alpha, \neg p_i)$ is the set $\text{dec}(\alpha) \setminus \{v\}$, where $v$ is the value that $p_i$ decides in $\alpha$. Note that if two distinct processes $p_i$ and $p_j$ decide on the same value $v$, then $\text{dec}(\alpha, \neg p_i) = \text{dec}(\alpha, \neg p_j)$.

An algorithm $A$ is *wait-free* if in each execution of $A$, every process executes a finite number of steps or decides. Therefore, a process must decide if it executes an infinite number of steps. We only consider wait-free algorithms.

An algorithm is *full-information* if a process writes its entire local state in every write operation. We say that an algorithm is in *standard-form* if processes proceed in a sequence of asynchronous rounds. In a round, every process first executes a write operation and then asynchronously reads all registers. Note that if there is a wait-free algorithm solving some task, then there is a, possibly inefficient, full-information and standard-form wait-free algorithm solving this task. Since efficiency is not an issue in this paper, we only consider full-information and standard-form algorithms.

*$k$-set agreement.* In *$k$-set agreement* [6], $1 \leq k \leq n$, each process $p_i$ proposes a value has to decide on a value, such that the following properties hold.

**Termination:** Each process executes a finite number of steps or decides.
**Validity:** A decided value is a proposed value.
**$k$-Agreement:** At most $k$ different values are decided.

We now define a trivial task $\mathcal{T}$ that will be used in the impossibility proof of $k$-set agreement. Indeed, in Section 4 we shall see that every wait-free algorithm that solves $\mathcal{T}$ possesses a property which implies that there is no algorithm that solves $k$-set agreement for $k < n$.

In the task $\mathcal{T}$ each process $p_i$ proposes a value and each process has to decide a value such that the termination and validity properties of $k$-set agreement are satisfied. Obviously, $\mathcal{T}$ is wait-free solvable: Each process can just decide it proposal, or each process can decide the smallest proposed value it sees in the shared memory. The following lemma is immediate from the definition of $k$-set agreement and $\mathcal{T}$.

**Lemma 1.** *Any wait-free algorithm that solves $k$-set agreement for some $k$, $1 \leq k \leq n$, also solves $\mathcal{T}$.*

## 3 Immediate snapshot executions

Consider a full-information and standard-form wait-free algorithm. The immediate snapshot executions of this algorithm form a subset of all its possible

executions. The executions in this set have a structure that makes them easier to analyze.

An *immediate snapshot* (IS) execution [1, 3, 4, 18] is modeled by a sequence of non-empty sets of processes $\alpha = s_1, \ldots, s_l, \ldots$. Processes in $s_l$ first perform, one by one, a write operation and then read all registers. Intuitively, the processes execute a concurrent write followed by a concurrent atomic snapshot of the shared memory. If $p_i \in s_l$, then we say that $p_i$ is *active* in the $l$-th set of $\alpha$.

Since we only consider wait-free algorithms, we can restrict our attention to minimal final IS executions, namely, each process executes computation steps until it decides. Indeed, each process decides in the last round in which it is active. We sometimes write $\alpha$ as a concatenation of sequences of sets, i.e., $\alpha = \alpha_1\ \alpha_2$, where $\alpha_1 = s_1\ s_2\ \ldots\ s_\ell$ and $\alpha_2 = s_{\ell+1}\ s_{\ell+2}\ \ldots\ s_t$.

For example, $\alpha = \{p, q\}\ \{r\}$ denotes an IS execution made of 2 sets. Processes $p$ and $q$ are active in the first set and $r$ is active in the second one. Observe that $p$ and $q$ see each other, but do not see $r$ because it executes steps of computation after $p$ and $q$ read the whole memory.

Although IS execution are well-structured they still have uncertainty, as the following example shows. In addition to the execution $\alpha$ described in the previous paragraph, consider the IS execution $\alpha' = \{p\}\ \{q\}\ \{r\}$. Note that $p$ only sees itself in $\alpha'$ while sees itself and $q$ in $\alpha$. The reader can verify that the views of $q$ and $r$ are the same in $\alpha$ and $\alpha'$. Therefore, $\alpha \overset{\neg p}{\sim} \alpha'$.

For a sequence $\alpha = s_1\ s_2\ \ldots\ s_t$ of sets of processes and a process $p_i$, we write $p_i \notin \alpha$ if $p_i \notin s_\ell$ for every $\ell$, $1 \leq \ell \leq t$, . Alternatively, we say that $p_i$ *does not appear* in $\alpha$. Also, for $p_i$ and $r \geq 1$, we let $\{p_i\}^r$ denote the sequence containing $r$ times the set $\{p_i\}$.

Formally, we say that a process $p_i$ is *unseen* in an IS execution $\alpha = s_1\ s_2\ \ldots\ s_t$ if and only if there exists $\ell$, $1 \leq \ell \leq t-1$, such that $p_i \notin s_x$, $1 \leq x \leq \ell$ and $s_y = \{p_i\}$, $\ell < y \leq t$. Therefore, if $p_i$ is unseen in $\alpha$, then $\alpha = \alpha'\,\{p_i\}^r$, for some $r \geq 1$ and $\alpha'$, such that $p_i \notin \alpha'$. Intuitively, $p_i$ is unseen in $\alpha$ since every step of $p_i$ occurs after all other processes decided. A process $p_i$ is *seen* in an IS execution $\alpha$ if it is not unseen in $\alpha$.

Lemmas 2 and 3 below are from [1]. They capture two properties about the uncertainty in minimal final IS executions. They will be used in the impossibility proof of $k$-set agreement presented in the following section. Lemma 2 is Lemma 3.3 in [1] and Lemma 3 is a restatement of Lemma 3.4 in [1].

**Lemma 2.** *If a process $p_i$ is unseen in a minimal final IS execution $\alpha$, then there is no minimal final IS execution $\alpha'$ such that $\alpha \overset{\neg p_i}{\sim} \alpha'$.*

**Lemma 3.** *Let $\alpha$ be a minimal final IS execution in which a process $p_i$ is seen. Then there is a unique minimal final IS execution $\alpha'$ such that $\alpha \overset{\neg p_i}{\sim} \alpha'$.*

Let $A$ be a wait-free algorithm. For any given collection $C$ of inputs to the processes, let $S$ be the set containing all minimal final IS executions of $A$ in which the processes start with inputs $C$. For a subset $P$ of processes, let $PS_P(S)$ be the set containing all executions in $S$ with participating set $P$. If there is no ambiguity, we write $PS_P$ instead of $PS_P(S)$.

For a proper subset of processes $P$ and a process $p \notin P$, the next lemma shows that there is a one-to-one correspondence between the executions in $PS_P$ and the executions in $PS_{P\cup\{p\}}$ in which $p$ is unseen.

**Lemma 4.** *For every proper subset of processes $P$ and a process $p_i \notin P$, the following properties hold:*

1. *For every execution $\alpha \in PS_P$ there is a unique execution $\alpha' \in PS_{P\cup\{p_i\}}$ such that $p_i$ is unseen in $\alpha'$ and $\alpha$ is equal to the maximal prefix of $\alpha'$ in which $p_i$ does not appear.*
2. *For every execution $\alpha \in PS_{P\cup\{p_i\}}$ such that $p_i$ is unseen in $\alpha$, $PS_P$ contains the maximal prefix $\alpha'$ of $\alpha$ in which $p_i$ does not appear.*

*Proof.* First let us consider an execution $\alpha \in PS_P$. Since $A$ is asynchronous wait-free, $\alpha$ can be extended to a minimal final execution $\alpha'$ with $\mathrm{ps}(\alpha') = P \cup \{p_i\}$, namely, $\alpha' = \alpha \{p_i\}^r$, for some $r \geq 1$. Thus $\alpha' \in PS_{P\cup\{p_i\}}$. Note that $p_i$ is unseen in $\alpha'$ because by hypothesis $p_i \notin \alpha$. In addition, $\alpha$ is the maximal prefix of $\alpha'$ in which $p_i$ does not appear. Moreover, $\alpha'$ is unique because $A$ is deterministic.

Now let us consider an execution $\alpha \in PS_{P\cup\{p_i\}}$ such that $p_i$ is unseen in $\alpha$. We have that $\alpha = \alpha' \{p_i\}^r$, for some $r \geq 1$ and $p_i \notin \alpha'$. Note that $\alpha'$ is the maximal prefix of $\alpha$ in which $p_i$ does not appear. Moreover, $\alpha' \in PS_P$ because, by hypothesis, each process that appears in $\alpha'$ executes steps of computation until it decides, namely, it is a minimal final IS execution with $\mathrm{ps}(\alpha') = P$. □

## 4 Impossibility of $k$-set agreement

This section is devoted to proving the impossibility of the $k$-set agreement task in the asynchronous wait-free read/write shared memory model (Theorem 1). The core of the proof is Lemma 6, roughly showing that every wait-free algorithm solving the task $\mathcal{T}$ (defined in Section 2) maintains an invariant concerning the number of its executions in which $\ell$ distinct processes decide $\ell$ distinct values. Indeed this lemma can be regarded as the operational counterpart of the Sperner's lemma [10, p. 36]. Then, using Lemma 1 and the invariant in Lemma 6, we conclude that $k$-set agreement is not wait-free solvable.

Let $A$ be a wait-free algorithm that solves $\mathcal{T}$. Consider $n$ distinct input values $v_1, \ldots, v_n$ and let $S$ be the set containing all minimal final IS executions of $A$ in which process $p_i$ has input $v_i$, $1 \leq i \leq n$.

Recall that $PS_P$ is the set of all executions in $S$ with participating set $P$. The next simple lemma directly follows from the validity property of $\mathcal{T}$, and it is used in the proof of Lemma 6 below.

**Lemma 5.** *For every subset of processes $P$ and for every execution $\alpha \in PS_P$,* $\mathrm{dec}(\alpha)$ *contains only the inputs of processes in $P$.*

When $P = \{p_1, \ldots, p_t\}$, $1 \leq t \leq n$, we write $PS_t$ instead of $PS_P$.

**Lemma 6.** *For all $t, 1 \leq t \leq n$, $PS_t$ contains exactly an odd number of executions $\alpha_1, \ldots, \alpha_{2q+1}$, $q \geq 0$, such that* $\mathrm{dec}(\alpha_j) = \{v_1, \ldots, v_t\}$, $1 \leq j \leq 2q+1$.

Intuitively, the proof of Lemma 6 proceeds by constructing a graph $G$ using the executions in $PS_t$ as vertices and putting an edge between two executions $\alpha, \alpha' \in PS_t$ if and only if at most one process distinguishes between them and at least $t-1$ distinct values are decided in $\alpha$ and $\alpha'$. The graph $G$ also contains an "imaginary" vertex $v^*$, which is defined in such a way that it has odd degree. This guarantees that $G$ contains at least one vertex with odd degree. All other vertices of $G$ with odd degree, if they exist, correspond to the executions of $PS_t$ in which the values $v_1, \ldots, v_t$ are decided. Let $M$ be the set containing all vertices of $G$ with odd degree except $v^*$. Using the well known handshaking lemma, stating that every graph has an even number of vertices with odd degree,[3] the proof finally concludes $|M \cup \{v^*\}|$ is even, hence $|M|$ is odd, which proves the lemma.

*Proof (of Lemma 6).* We proceed by induction on $t$. For $t = 1$, $PS_1$ only contains $p_1$'s solo execution, that is, $PS_1$ only contains $\alpha = \{p_1\}^r$, for some $r \geq 1$. Obviously, $p_1$ must decide $v_1$ in $\alpha$, which proves the base of the induction. Now suppose that the lemma holds for $t, 1 \leq t \leq n-1$. We prove it holds for $t+1$.

We define a graph $G = (V, E)$ whose vertices are the executions in $PS_{t+1}$, plus an additional vertex $v^*$, that is, $V = PS_{t+1} \cup \{v^*\}$. The edges of $G$ are defined as follows (recall that $v_i$ is the input of process $p_i$).

- For every pair of executions $\alpha, \alpha' \in PS_{t+1}$, $(\alpha, \alpha') \in E$ if and only if there is a process $p_i$ such that $\alpha \overset{\neg p_i}{\sim} \alpha'$ and $\mathrm{dec}(\alpha, \neg p_i) = \{v_1, \ldots, v_t\}$; hence, $\mathrm{dec}(\alpha', \neg p_i) = \{v_1, \ldots, v_t\}$.
- For every execution $\alpha \in PS_{t+1}$, $(v^*, \alpha) \in E$ if and only if $p_{t+1}$ is unseen in $\alpha$ and $\mathrm{dec}(\alpha, \neg p_{t+1}) = \{v_1, \ldots, v_t\}$.

Fig. 1 depicts an example of the graph $G$ constructed in the inductive step $t = 2$ for a one-round algorithm for three processes $p$, $q$ and $r$. In the algorithm, a process decides its proposal if it sees less than 2 processes. Otherwise, if it sees that its proposal is not the largest one, then decides the smallest proposal, and it decides the second smallest proposal in any other case. In Fig. 1 the inputs for $p$, $q$ and $r$ are 1, 2 and 3, respectively, and process $r$ corresponds to $p_{t+1}$. In each execution, above each process it appears the value that the process decides in that execution. Observe that there is an edge between executions $\alpha = \{p\}\,\{q, r\}$ and $\alpha' = \{p, q, r\}$ because, first, $\alpha \overset{\neg p}{\sim} \alpha'$ ($p$ sees no other process than itself in $\alpha$ while it sees $q$ and $r$ in $\alpha'$), and second, $q$ and $r$ decide 1 and 2, respectively, in $\alpha$. Also there is an edge between $v^*$ and $\alpha = \{p, q\}\,\{r\}$ because $r$ is unseen in $\alpha$ and $p$ and $q$ decide 1 and 2. Finally, note that there is no edge between $\alpha = \{p, q, r\}$ and $\alpha' = \{r\}\,\{p, q\}$ because although $\alpha \overset{\neg r}{\sim} \alpha'$, both $p$ and $q$ decide 1 in $\alpha$.

For the rest of the proof, let $\deg(v)$ denote the degree of a vertex $v$ of $G$. Below, we prove the following properties about the degrees of the vertices in $G$.

1. $\deg(v^*)$ is odd.

[3] This result is a consequence of Euler's observation that for every graph $G = (V, E)$, $\sum_{v \in V} \deg(v) = 2\,|E|$.

2. For all $\alpha \in PS_{t+1} = V \setminus \{v^*\}$:
   (a) If $\mathrm{dec}(\alpha) = \{v_1, \ldots, v_{t+1}\}$, then $\deg(\alpha) = 1$.
   (b) If $\mathrm{dec}(\alpha) = \{v_1, \ldots, v_t\}$, then $\deg(\alpha) = 2$.
   (c) Otherwise, $\deg(\alpha) = 0$.

These properties, can be use to derive the induction step, together with the next well-known result of graph theory.

**Handshaking Lemma.** *Every graph has an even number of vertices with odd degree.*

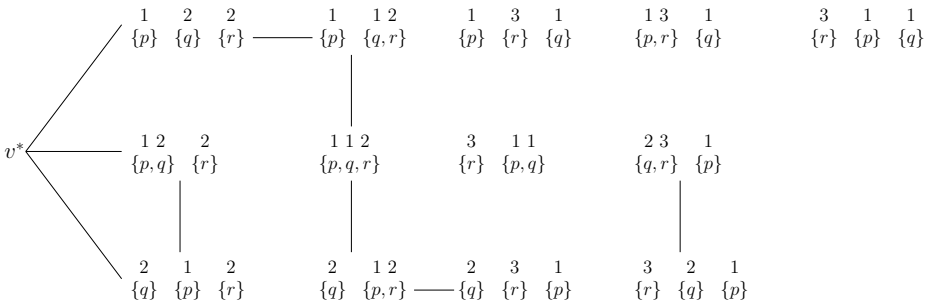


**Fig. 1.** Graph associated with a 3-process 1-round algorithm.

The vertices of $G$ with odd degree are exactly $M \cup \{v^*\}$, where $M$ is the set that contains every execution $\alpha \in PS_{t+1}$ such that $\mathrm{dec}(\alpha) = \{v_1, \ldots, v_{t+1}\}$. Thus, by the Handshaking Lemma, $|M \cup \{v^*\}|$ is even, hence $|M|$ is odd. We now prove the properties of the degrees.

$\deg(v^*)$ *is odd.* Consider an execution $\alpha \in PS_t$. By Lemma 4(1), there is a unique execution $\alpha' \in PS_{t+1}$ such that $p_{t+1}$ is unseen in $\alpha'$ and $\alpha$ is equal to the maximal prefix of $\alpha'$ in which $p_{t+1}$ does not appear. Conversely, by Lemma 4(2), for every execution $\alpha \in PS_{t+1}$ such that $p_{t+1}$ is unseen in $\alpha$, $PS_t$ contains the maximal prefix $\alpha'$ of $\alpha$ in which $p_{t+1}$ does not appear.

By the induction hypothesis, we have that $PS_t$ contains exactly an odd number of executions $\alpha_1, \ldots, \alpha_{2q+1}$, $q \geq 0$, such that $\mathrm{dec}(\alpha_j) = \{v_1, \ldots, v_t\}$, $1 \leq j \leq 2q+1$. Thus for every $\alpha_j$, $1 \leq j \leq 2q+1$, there is a unique execution $\alpha'_j \in PS_{t+1}$ such that $p_{t+1}$ is unseen in $\alpha'_j$ and $\alpha_j$ is a prefix of $\alpha'_j$. Hence, $(v^*, \alpha'_j) \in E$, since $\mathrm{dec}(\alpha'_j, \neg p_{t+1}) = \{v_1, \ldots, v_t\}$ because $\mathrm{dec}(\alpha_j) = \{v_1, \ldots, v_t\}$. Moreover, these are all the edges adjacent to $v^*$ because, as explained above, there is a one-to-one correspondence between the executions of $PS_t$ and the executions of $PS_{t+1}$ in which $p_{t+1}$ is unseen. Therefore, $\deg(v^*)$ is odd.

*For every execution* $\alpha \in PS_{t+1}$, *if* $\text{dec}(\alpha) = \{v_1, \ldots, v_{t+1}\}$, *then* $\deg(\alpha) = 1$. Since $|\text{dec}(\alpha)| = |\{v_1, \ldots, v_{t+1}\}| = t+1$ and $|\text{ps}(\alpha)| = |\{p_1, \ldots, p_{t+1}\}| = t+1$, it follows that for every $v \in \{v_1, \ldots, v_{t+1}\}$, there is exactly one process in $\alpha$ that decides $v$. Let $p$ denote the process that decides $v_{t+1}$ in $\alpha$. Thus, $\text{dec}(\alpha, \neg p) = \{v_1, \ldots, v_t\}$. We identify two subcases: $p$ is unseen in $\alpha$ or $p$ is seen in $\alpha$.

If $p$ is unseen in $\alpha$, then, by Lemma 4(2), $PS_P$ contains the maximal prefix $\alpha'$ of $\alpha$ in which $p$ does not appear, where $P = \{p_1, \ldots, p_{t+1}\} \setminus \{p\}$. Observe that $\text{dec}(\alpha') = \{v_1, \ldots, v_t\}$. Lemma 5 and the assumption that each $p_i$ has a distinct input $v_i$, imply that $P = \{p_1, \ldots, p_{t+1}\} \setminus \{p\} = \{p_1, \ldots, p_t\}$, hence $p = p_{t+1}$. Therefore, $p_{t+1}$ is unseen in $\alpha$ and $\text{dec}(\alpha, \neg p_{t+1}) = \{v_1, \ldots, v_t\}$, because as explained above $\text{dec}(\alpha, \neg p) = \{v_1, \ldots, v_t\}$. By definition of $G$, $(v^*, \alpha) \in E$.

We claim that $(v^*, \alpha)$ is the only edge that is adjacent to $\alpha$. First, there is no execution $\alpha' \in PS_{t+1}$ such that $\alpha \overset{\neg q}{\sim} \alpha'$ with $q = p_{t+1}$, by Lemma 2. Second, for every process $q \in \{p_1, \ldots, p_t\}$, $\text{dec}(\alpha, \neg q) \neq \{v_1, \ldots, v_t\}$ because for each $v \in \{v_1, \ldots, v_{t+1}\}$ there is exactly one process in $\alpha$ that decides $v$, and $q$ decides on a value in $\{v_1, \ldots, v_t\}$. These two observations imply that there is no $\alpha' \in PS_{t+1}$ such that $(\alpha, \alpha') \in E$, and hence $\deg(\alpha) = 1$.

For the second subcase, namely, $p$ is seen in $\alpha$, there is only one execution $\alpha' \in PS_{t+1}$ such that $\alpha \overset{\neg q}{\sim} \alpha'$, by Lemma 3. Then $(\alpha, \alpha') \in E$, because $\text{dec}(\alpha, \neg p) = \{v_1, \ldots, v_t\}$.

The edge $(\alpha, \alpha')$ is the only edge that is adjacent to $\alpha$: We have that for every $q \in \{p_1, \ldots, p_t\}$, $\text{dec}(\alpha, \neg q) \neq \{v_1, \ldots, v_t\}$ because for each $v \in \{v_1, \ldots, v_{t+1}\}$ there is exactly one process in $\alpha$ that decides $v$, and $q$ decides a value of $\{v_1, \ldots, v_t\}$. Therefore, there does not exist a $\alpha'' \in PS_{t+1}$ such that $\alpha'' \neq \alpha'$ and $(\alpha, \alpha'') \in E$. Hence we get $\deg(\alpha) = 1$.

*For every* $\alpha \in PS_{t+1}$, *if* $\text{dec}(\alpha) = \{v_1, \ldots, v_t\}$, *then* $\deg(\alpha) = 2$. Since $|\text{dec}(\alpha)| = |\{v_1, \ldots, v_t\}| = t$ and $|\text{ps}(\alpha)| = |\{p_1, \ldots, p_{t+1}\}| = t+1$, we get that there must be a value $\bar{v} \in \{v_1, \ldots, v_t\}$ such that there are two distinct processes $q_1, q_2 \in \{p_1, \ldots, p_{t+1}\}$ that decide $\bar{v}$ in $\alpha$. Therefore, for each $i \in \{1, 2\}$, $\text{dec}(\alpha, \neg q_i) = \{v_1, \ldots, v_t\}$. Also observe that $\bar{v}$ is the unique value of $\{v_1, \ldots, v_t\}$ that has that property.

We identify three subcases: $q_1$ is unseen and $q_2$ is seen in $\alpha$, $q_1$ is seen and $q_2$ is unseen in $\alpha$, and both $q_1$ and $q_2$ are seen in $\alpha$. Note that it is impossible that both $q_1$ and $q_2$ are unseen.

Consider first the subcase $q_1$ is unseen and $q_2$ is seen in $\alpha$. The argument is similar to the one in the previous case. If $q_1$ is unseen in $\alpha$, then, by Lemma 4(2), $PS_P$ contains the maximal prefix $\alpha'$ of $\alpha$ in which $q_1$ does not appear, where $P = \{p_1, \ldots, p_{t+1}\} \setminus \{q_1\}$. Observe that $\text{dec}(\alpha') = \{v_1, \ldots, v_t\}$. Thus, by Lemma 5 and the assumption that each $p_i$ has a distinct input $v_i$, we get $P = \{p_1, \ldots, p_{t+1}\} \setminus \{q_1\} = \{p_1, \ldots, p_t\}$, and hence $q_1 = p_{t+1}$. Therefore, $p_{t+1}$ is unseen in $\alpha$ and $\text{dec}(\alpha, \neg p_{t+1}) = \{v_1, \ldots, v_t\}$, because $\text{dec}(\alpha, \neg q_1) = \{v_1, \ldots, v_t\}$. Hence $(v^*, \alpha) \in E$.

For $q_2$, by Lemma 3, there is an execution $\alpha' \in PS_{t+1}$ such that $\alpha \overset{\neg q_2}{\sim} \alpha'$. Then $(\alpha, \alpha') \in E$, because $\text{dec}(\alpha, \neg q_2) = \{v_1, \ldots, v_t\}$.

We claim that $(v^*, \alpha)$ and $(\alpha, \alpha')$ are the unique edges adjacent to $\alpha$. First, for $q_1$, there is no execution $\alpha'' \in PS_{t+1}$ such that $\alpha \overset{\neg q_1}{\sim} \alpha''$, by Lemma 2. Second, for $q_2$, there is no $\alpha'' \neq \alpha'$ such that $\alpha \overset{\neg q_2}{\sim} \alpha''$, by Lemma 3. And third, for every process $q \in \{p_1, \ldots, p_{t+1}\}$ distinct from $q_1$ and $q_2$, $\mathrm{dec}(\alpha, \neg q) \neq \{v_1, \ldots, v_t\}$ because $q$ decides on a value in $\{v_1, \ldots, v_t\}$ and, as mentioned above, there is no other process in $\{p_1, \ldots, p_{t+1}\}$ that decides the same value as $q$. Therefore, $\deg(\alpha) = 2$.

The subcase in which $q_1$ is seen and $q_2$ is unseen in $\alpha$ is symmetric to the previous one.

For the third subcase, that is, $q_1$ and $q_2$ are seen in $\alpha$, for each $i \in \{1, 2\}$ there is an execution $\alpha'_i \in PS_{t+1}$ such that $\alpha \overset{\neg q_i}{\sim} \alpha'_i$, by Lemma 3. Then $(\alpha, \alpha'_i) \in E$, because $\mathrm{dec}(\alpha, \neg q_i) = \{v_1, \ldots, v_t\}$. As in the first subcase, it can be easily proved that there is no extra edge that is adjacent to $\alpha$. Hence $\deg(\alpha) = 2$.

*Otherwise,* $\deg(\alpha) = 0$. We have two subcases: $|\mathrm{dec}(\alpha)| < t$, or $|\mathrm{dec}(\alpha)| = t$ and $\mathrm{dec}(\alpha) \neq \{v_1, \ldots, v_t\}$. In both subcases, for every process $p \in \{p_1, \ldots, p_{t+1}\}$, $\mathrm{dec}(\alpha, \neg p) \neq \{v_1, \ldots, v_t\}$, hence it follows from the definition of $G$ that $\deg(\alpha) = 0$. □

**Theorem 1.** *There is no wait-free algorithm that solves $k$-set agreement for $k < n$.*

*Proof.* Suppose that there is a wait-free algorithm $A$ that solves $k$-set agreement, for some $k$, $1 \leq k \leq n-1$. By Lemma 1, $A$ solves $\mathcal{T}$. Consider $n$ distinct input values $v_1, \ldots, v_n$ and let $S$ be the set containing all minimal final IS executions of $A$ in which process $p_i$ has input $v_i$, $1 \leq i \leq n$. Consider the set $PS_{k+1}(S)$, namely, the set containing all executions in $S$ with participating set $p_1, \ldots, p_{k+1}$. By Lemma 6, $PS_{k+1}(S)$ contains an execution $\alpha$ such that $|\mathrm{dec}(\alpha)| = k+1$. But this contradicts the fact that $A$ solves $k$-set agreement, since the $k$-agreement property means that in all executions $\alpha \in PS_{k+1}(S)$, $|\mathrm{dec}(\alpha)| \leq k$. □

## 5 An operational perspective

From an operational perspective we can think of the proof of Lemma 6, and hence the impossibility proof of $k$-set agreement, in the following way.

The induction hypothesis of the proof states that there is an odd number of minimal final IS executions $\alpha_1, \ldots, \alpha_{2q+1}$, $q \geq 0$, such that for each $\alpha \in S = \{\alpha_1, \ldots, \alpha_{2q+1}\}$, $\mathrm{ps}(\alpha) = \{p_1, \ldots, p_k\}$ and $\mathrm{dec}(\alpha) = D = \{v_1, \ldots, v_k\}$, where $v_i$ is the input of $p_i$, $1 \leq i \leq k$. Namely, $k$ processes decide $k$ distinct values in $\alpha$. As explained in the proof, each $\alpha \in E$ can be extended to a minimal final IS execution $\gamma_1$ such that $\mathrm{ps}(\gamma_1) = \{p_1, \ldots, p_{k+1}\}$ and $p_{k+1}$ is unseen in $\gamma_1$. Thus, $\alpha$ is the maximal prefix of $\gamma_1$ in which $p_{k+1}$ does not appear, hence $p_{k+1}$ decides after all processes $p_1, \ldots, p_k$ decide in $\gamma_1$. Moreover, $\mathrm{dec}(\gamma_1, \neg p_{k+1}) = D$ because $\mathrm{dec}(\alpha) = D$. Let us fix $\alpha$ and $\gamma_1$.

Then, taking advantage of the uncertainty related to IS execution, what the proof essentially does is to produce a path $P$ in the graph $G$ that starts in $\gamma_1$.

If $|\mathrm{dec}(\gamma_1)| = k+1$, then $P$ only contains $\gamma_1$. Otherwise, $|\mathrm{dec}(\gamma_1)| = |D| = k$ and the proof looks for an execution $\gamma_2$ such that there is a process $p$ such that $\gamma_1 \overset{\neg p}{\sim} \gamma_2$, and hence $\mathrm{dec}(\gamma_2, \neg p) = D$. If there is no such execution $\gamma_2$ then $P$ only contains $\gamma_1$, otherwise $\gamma_2$ is appended to $P$ and the procedure continues by considering $\gamma_2$ instead of $\gamma_1$.

In the end we get a path $P = \gamma_1, \gamma_2, \ldots, \gamma_s$, such that $s \geq 1$ and for each $i$, $1 \leq i \leq s-1$, there is a process $p$ such that $\gamma_i \overset{\neg p}{\sim} \gamma_{i+1}$ and $\mathrm{dec}(\gamma_i, \neg p) = D$. What is important to notice is that the last execution (vertex) $\gamma_s$ of $P$ holds either $|\mathrm{dec}(\gamma_s)| = k+1$, or $|\mathrm{dec}(\gamma_s)| = |D| = k$ and $p_{k+1}$ is unseen in $\gamma_s$. In the second case we have that the maximal prefix $\alpha'$ of $\gamma_s$ in which $p_{k+1}$ does not appear, belongs to $S$. Therefore, in some sense, the path $P$ "matches" $\alpha$ (the maximal prefix of $\gamma_1$ in which $p_{k+1}$ does not appear) to $\alpha'$. For example, in the graph in Fig. 1, the sequence of IS executions $\{p\ q\}\,\{r\}$, $\{q\}\,\{p\}\,\{r\}$ matches $\{p\ q\}$ to $\{q\}\,\{p\}$. However, only an even number of execution of $S$ can be matched in this way, and since $|S|$ is odd, we get that there must be a path that matches an $\alpha \in S$ to an execution in which $k+1$ values are decided. In Fig. 1, the path that starts at $\{p\}\,\{q\}\,\{r\}$ and ends at $\{q\}\,\{r\}\,\{p\}$, matches $\{p\}\,\{q\}$ to $\{q\}\,\{r\}\,\{p\}$.

Finally, an execution $\gamma$ with $|\mathrm{dec}(\gamma)| = k+1$, that is not matched to an execution in $S$, is matched to an execution $\gamma'$ with $|\mathrm{dec}(\gamma')| = k+1$. In Fig. 1, $\{q\ r\}\,\{p\}$ is matched to $\{r\}\,\{q\}\,\{p\}$.

*Acknowledgments:* The second author would like to thank Michel Raynal and Julien Stainer for useful discussions and comments on earlier versions of this paper.

## References

1. Attiya H. and Rajsbaum S., The Combinatorial Structure of Wait-Free Solvable Tasks. *SIAM Journal on Computing*, 31(4):1286-1313, 2002.
2. Attiya H. and Welch J., Distributed Computing: Fundamentals, Simulations and Advanced Topics. *McGraw-Hill* 1998.
3. Borowsky E. and Gafni E., Generalized FLP Impossibility Result for $t$-Resilient Asynchronous Computations. *Proc. 25th ACM Symposium on Theory of Computing (STOC'93)*, ACM Press, pp. 91-100, 1993.
4. Borowsky E. and Gafni E., Immediate Atomic Snapshots and Fast Renaming. *Proc. 12th ACM Symposium on Principles of Distributed Computing (PODC'93)*, pp. 41-51, 1993.
5. Castañeda A. and Rajsbaum S., New Combinatorial Topology Upper and Lower Bounds for Renaming. *Proc. 27th ACM Symposium on Principles of Distributed Computing (PODC'08)*, ACM Press, pp. 295-304, 2008.
6. Chaudhuri S., More Choices Allow More Faults: Set Consensus Problems in Totally Asynchronous Systems. *Information and Computation*, 105(1):132-158, 1993.
7. Fischer M.J., Lynch N.A. and Paterson M.S., Impossibility of Distributed Consensus with One Faulty Process. *Journal of the ACM*, 32(2):374-382, 1985.
8. E. Gafni and E. Koutsoupias. Three-Processor Tasks Are Undecidable. *SIAM Journal on Computing* 28(3): 970–983 (1999).

9. Gafni E., Rajsbaum S. and Herlihy M.P., Subconsensus Tasks: Renaming is Weaker Than Set Agreement. *Proc. 20th Int'l Symposium on Distributed Computing (DISC'06)*, Springer Verlag LNCS #4167, pp.329-338, 2006.
10. Henle M., A Combinatorial Introduction to Topology. *Dover* 1994.
11. Herlihy M., Wait-free synchronization. *Transactions on Programming Languages and Systems*, 13(1):124–149, 1991.
12. Herlihy M.P. and Rajsbaum S., Set Consensus Using Arbitrary Objects (Preliminary Version). *Proc. 13th Annual ACM Symposium on Principles on Distributed Computing (PODC'94)*, ACM Press, pp. 324-333, 1994.
13. Herlihy M.P. and Rajsbaum S., Algebraic Spans. *Mathematical Structures in Computer Science*, 10(4):549-573, 2000.
14. Herlihy M.P. and Rajsbaum S., A Classification of Wait-free Loop Agreement Tasks. *Theoretical Computer Science,* 291(1): 55-77, 2003.
15. Herlihy M.P. and Shavit N., The Topological Structure of Asynchronous Computability. *Journal of the ACM*, 46(6):858-923, 1999.
16. G. Hoest and N. Shavit, Toward a Topological Characterization of Asynchronous Complexity. *SIAM Journal on Computing* 36(2): 457–497 (2006).
17. Loui M. and Abu-Amara H., Memory requirements for agreement among unreliable asynchronous processes. *F. P. Preparata, editor, Advances in Computing Research*, volume 4, pages 163–183. JAI Press, Greenwich, CT, 1987.
18. Saks M. and Zaharoglou F., Wait-Free k-Set Agreement Is Impossible: The Topology of Public Knowledge. *SIAM Journal on Computing*, 29(5):1449-1483, 2000.