Extension-Based Proofs for Synchronous Message Passing

Yilun Sheng
Institute for Interdisciplinary Information Sciences, Tsinghua University, Beijing, China

Faith Ellen
Department of Computer Science, University of Toronto, Canada

Abstract

There is no wait-free algorithm that solves k-set agreement among $n \geq k+1$ processes in asynchronous systems where processes communicate using only registers. However, proofs of this result for $k \geq 2$ are complicated and involve topological reasoning. To explain why such sophisticated arguments are necessary, Alistarh, Aspnes, Ellen, Gelashvili, and Zhu recently introduced extension-based proofs, which generalize valency arguments, and proved that there are no extension-based proofs of this result.

In the synchronous message passing model, k-set agreement is solvable, but there is a lower bound of t rounds for any k-set agreement algorithm among $n > kt$ processes when at most k processes can crash each round. The proof of this result for $k \geq 2$ is also a complicated topological argument. We define a notion of extension-based proofs for this model and we show there are no extension-based proofs that t rounds are necessary for any k-set agreement algorithm among $n = kt + 1$ processes, for $k \geq 2$ and $t > 2$, when at most k processes can crash each round. In particular, our result shows that no valency argument can prove this lower bound.

2012 ACM Subject Classification Theory of computation → Interactive proof systems; Theory of computation → Complexity theory and logic; Theory of computation → Distributed algorithms; Theory of computation → Distributed computing models

Keywords and phrases Set agreement, lower bounds, valency arguments

Digital Object Identifier 10.4230/LIPIcs.DISC.2021.36

Acknowledgements Faith Ellen would like to thank Sergio Rajsbaum and Leqi Zhu for some preliminary discussions about this problem. Support is gratefully acknowledged from the Natural Science and Engineering Research Council of Canada under grant RGPIN-2020-04178. Part of this work was done while Yilun Sheng was virtually visiting University of Toronto.

1 Introduction

In the *k-set agreement* problem, each process has an input from $\{0, \ldots, k\}$ and each process that does not crash must output a value from among the inputs *(validity)* such that at most k different values are output *(k-agreement)*. In 1993, Borowsky and Gafni [6], Herlihy and Shavit [11], and Saks and Zaharoglou [14] concurrently proved, using sophisticated topological proofs, that there are no wait-free algorithms that solve k-set agreement among $n \geq k + 1$ processes in asynchronous models where processes communicate by reading from and writing to shared registers (or objects that can be built from registers).

Extension-based proofs were recently introduced by Alistarh, Aspnes, Ellen, Gelashvili, and Zhu [2, 3] as a generalization of valency arguments. They proved that, for $k \geq 2$, there are no extension-based proofs of this impossibility result in the iterated immediate snapshot and iterated snapshot models. These are asynchronous shared-memory models that are closely related to the models used by Borowsky and Gafni, Herlihy and Shavit, and Saks and Zaharoglou. This explains the necessity of the topological arguments used to prove this

impossibility result. Subsequently, Attiya, Castañeda, and Rajsbaum [4] presented a simpler, but more restricted, class of proofs and showed that, for $k \geq 2$, this class contains no proof of the impossibility of a uniform algorithm solving k-set agreement among $n \geq k+1$ processes in the iterated immediate snapshot model.

In the synchronous message passing model, Chaudhuri, Herlihy, Lynch, and Tuttle [7] proved that $\lfloor f/k \rfloor + 1$ rounds are necessary and sufficient to solve k-set agreement among $n \geq k+f+1$ processes when at most f processes can crash. Their upper bound is a simple, elegant algorithm, which is described in Section 3. The proof of their lower bound is a complicated topological argument, which constructs a k-dimensional simplex of executions in which adjacent executions are indistinguishable to certain processes. Subsequent papers by Herlihy, Rajsbaum, and Tuttle [9, 10] presented more succinct proofs using additional topological tools, showing that it is impossible to solve k-set agreement among $n \geq k(t+1)+1$ processes in t rounds, assuming at most k processes crash each round. Gafni [8] gave a different proof. He presented a technique that converts any synchronous t round k-set agreement algorithm for $n \geq k(t+1)+1$ processes, assuming at most k processes crash each round, into an asynchronous k-set agreement algorithm for $n \geq k+1$ processes, assuming at most k processes crashes. Since there is no such asynchronous algorithm, he obtained the same lower bound.

A natural question is whether there is a simpler proof of this lower bound, analogous to the valency arguments used to prove that more than t rounds are necessary to solve binary consensus among $n \geq t+2$ processes when at most one process can crash each round [1, 5, 13]. Following the approach by Alistarh, Aspnes, Ellen, Gelashvili, and Zhu [2, 3], we define a version of extension-based proofs for the synchronous message passing model in Section 4. Then, in Section 5, we prove that for $k, t \geq 2$, there is no extension-based proof of the impossibility of solving k-set agreement among $n = k(t+1)+1$ processes in t rounds when at most k processes crash each round.

2 Model

A *synchronous message passing model* consists of a set of n processes, $P = \{p_0, p_1, \ldots, p_{n-1}\}$. Executions of an algorithm proceed in synchronous *rounds*, in which each process (that has not already terminated or crashed) sends messages to other processes, then receives the messages other processes sent to it in that round, updates its state based on the messages it receives, and either outputs a value and terminates or proceeds to the next round.

A process *crashes* during some round if there is some other process to which it is supposed to send a message during that round, but it does not. If a process crashes during a round, it does not send any messages in any subsequent round and it does not output a value. A process is *active* at the beginning of round $r \geq 1$ if it has not terminated or crashed during the first $r-1$ rounds. Every process is active at the beginning of round 1.

Initially, each process p_i has an input x_i that is (part of) its state. A *configuration* is an n-component vector C that describes the states of all processes at the beginning of some round. If process p_i is active, then $C[i]$ is its state at the beginning of this round. If it terminated in a previous round, then $C[i] = (\top, m)$, where m is the value it output. If it crashed in a previous round, then $C[i] = \bot$. An *initial configuration* consists of the state of every process before any computation occurs. A *final configuration* is a configuration in which no process is active, i.e., every process has either terminated or crashed.

An *algorithm* specifies what message each active process wants to send to every other process in each round, as a function of its state at the beginning of the round. A *full information algorithm* is an algorithm in which, in every round, each active process p_i sends its state to every other process and updates its state to be the n-component vector s, where $s[i]$ is its previous state and $s[j]$ is the last message it received from process p_j, for all $j \neq i$. If process p_i has never received a message from process p_j, then $s[j] = \perp$. If process p_i terminates in state s and outputs m, then $s = (\top, m)$. Note that every algorithm can be transformed into a full information algorithm using the same number of rounds: It simply does not pay attention to any extra information it has received when deciding when to terminate and what value to output.

A *one round schedule* is an n-component vector σ, where $\sigma[i] \subseteq P$ consists of p_i and the set of processes to which p_i sent messages during the round, if p_i is active at the beginning of the round, and is $\emptyset$, otherwise. The one round schedule σ is *applicable* to configuration C if $p_i \in \sigma[i]$ for every process p_i that is active in C and $\sigma[i] = \emptyset$ for every process p_i that is not active in C. Note that if C has no active processes, then the only one round schedule applicable to C is the empty schedule σ, where $\sigma[i] = \emptyset$ for every process p_i.

Suppose that C is a configuration of an algorithm and σ is a one round schedule that is applicable to C. Consider the round in which each process p_i that is active in C sends the messages specified by the algorithm when p_i is in state $C[i]$ to the processes in $\sigma[i]$. Let C' be the configuration resulting from this round when performed from configuration C. Then $C\sigma$ denotes the configuration C'. The processes that *crash during* σ are those that have crashed in C', but have not crashed in C. In other words, p_i crashes during σ if $C'[i] = \perp$ and $C[i] \neq \perp$.

Two configurations C and C' are *indistinguishable* to process p_i when $C[i] = C'[i]$. Suppose that σ and σ' are one round schedules applicable to configurations C and C', respectively. If process p_i receives messages from the same set of processes Q during σ and σ' and configurations C and C' are indistinguishable to p_i and each of the processes in Q, then configurations $C\sigma$ and $C'\sigma'$ are indistinguishable to process p_i.

An *execution* of an algorithm from configuration C is an (infinite or finite) alternating sequence of configurations and one round schedules $C = C_0, \sigma_1, C_1, \sigma_2, C_2, \ldots$ such that σ_r is applicable to C_{r-1} and $C_r = C_{r-1}\sigma_r$ for all rounds $r \geq 1$. The sequence $\sigma = \sigma_1, \sigma_2, \ldots$ is called a *schedule applicable to* C. When σ has length t, it is called a t *round schedule* and $C\sigma$ denotes the configuration C_t. For any set of processes $Q \subseteq P = \{p_0, \ldots, p_{n-1}\}$, a Q*-only schedule* is a schedule in which only processes in Q send messages (i.e., $\sigma[j] \subseteq \{p_j\}$ for every process $p_j \notin Q$ and every round σ in the schedule). Note that, if configurations C and C' are indistinguishable to every process in Q, then the set of Q-only schedules applicable to C and C' are the same.

A *complete execution* ends with a final configuration. The *round complexity* of an algorithm is the maximum number t such that some process is active at the beginning of round t in some execution of the algorithm from an initial configuration. Every algorithm with round complexity t can be converted into an algorithm in which no process outputs a value before the end of round t: A process that outputs a value in round $r < t$ can, instead, simply send no messages in rounds $r+1$ through t and output the value at the end of round t.

3 The FloodMin Algorithm

The FloodMin Algorithm [7, 12] solves k-set agreement among $n \leq k(t+1)$ processes in t rounds, when at most k processes can crash each round. We present this algorithm because it motivates parts of the construction in Section 5.

In the first round of this algorithm, each process broadcasts its input value and then adopts the smallest value from among its own value and the values it received from other processes. In each successive round, each process broadcasts its adopted value and then adopts the smallest value from among its own value and the values it received from other processes. At the end of round t, each process outputs its adopted value.

Let $\alpha = C_0, \sigma_1, C_1, \sigma_2, \ldots, \sigma_t, C_t$ be an execution of FloodMin starting from an initial configuration, let $adopt(C_0)$ denote the set of input values of processes in configuration C_0, and, for each $r \in \{1, \ldots, t\}$, let $adopt(C_r)$ denote the set of values adopted by processes (that have not crashed) in configuration C_r. The proof that at most k different values are output in α is an easy consequence of the following two facts.

▶ **Lemma 1.** *$adopt(C_r) \subseteq adopt(C_{r-1})$ for all $r \in \{1, \ldots, t\}$.*

▶ **Lemma 2.** *If at most $d-1$ processes crash during round r, then $\#adopt(C_r) \leq d$.*

Since the set of output values, $adopt(C_t)$, is a subset of the set of input values, $adopt(C_0)$, FloodMin satisfies validity. It also satisfies k-agreement for $n \leq k(t+1)$ processes: If there is a round r of α in which fewer than k processes crash, then, by Lemma 2, $\#adopt(C_r) \leq k$ and, hence, by Lemma 1, $\#adopt(C_t) \leq k$. Otherwise, in every round of α, at least k processes crash. Hence, at the end of round t, there are at most $n - kt \leq k$ active processes and, thus, $\#adopt(C_t) \leq k$.

However, when $n = k(t+1)+1$, there are executions of FloodMin in which the set of adopted values has size $k+1$ at the end of round t. For example, consider an initial configuration C'_0 in which $kt+1$ processes have input k and one process has input i, for each $i \in \{0, \ldots, k-1\}$. Inductively, for $1 \leq r \leq t$, there is a configuration C'_r in which $k(t-r)+1$ processes have adopted value k and one process has adopted value i, for each $i \in \{0, \ldots, k-1\}$. It can be obtained from configuration C'_{r-1} by a one round schedule in which the processes with value k don't crash and the processes with other values each sends one message to a different process with value k and then crashes. In configuration C'_t, there are $k+1$ processes, each with a different adopted value. If they terminated at the end of round t, then k-agreement would be violated.

4 Extension-Based Proofs

As in the definition of an extension-based proof in the iterated immediate snapshot or iterated snapshot model [2, 3], an extension-based proof for the synchronous message passing model is an interaction between a prover and an algorithm, which the prover is trying to prove is incorrect. The prover starts with no knowledge about the algorithm except for its initial configurations.

Suppose that the algorithm claims to solve a task $\mathscr{T}$ for n processes within t rounds when up to f processes can crash each round. Without loss of generality, we may assume that no process outputs a value before the end of round t. Let M be the set of possible output values for $\mathscr{T}$. There are $t+1$ phases in the interaction.

In phase 0, the prover asks about the values that can be output in executions from initial configurations. A *query* (C, Q, u, m) consists of an initial configuration C, a set of at least $n-u$ processes Q, an upper bound $u \in \{0, \ldots, f\}$, and a value $m \in M$. The algorithm either:

- responds positively with a Q-only t round schedule σ from C and a process $p_i \in Q$ such that
  - at most u processes crash in each round of σ and
  - p_i has output m in configuration $C\sigma$
- or responds NONE, if there are no such schedule and process.

After a finite number of queries, the prover ends phase 0 by choosing an initial configuration C_0. Then the players proceed to phase 1.

Let $1 \leq r \leq t-1$. At the end of phase $r-1$, the prover has chosen a configuration C_{r-1} that is reachable from an initial configuration by an $r-1$ round schedule in which at most f processes crash each round. During phase r, the prover considers configurations that can be reached from C_{r-1} by one round schedules. It asks about the values that can be output in $t-r$ round executions from such configurations. A *query* $(C_{r-1}, \sigma'_r, u, m)$ consists of an upper bound $u \in \{0, \ldots, f\}$, a value $m \in M$, and a one round schedule σ'_r applicable to C_{r-1} in which at most f processes crash. The algorithm either:

- responds positively with a $t-r$ round schedule σ from $C_{r-1}\sigma'_r$ and a process p_i such that
  - at most u processes crash in any round of σ and
  - p_i has output m in configuration $C_{r-1}\sigma'_r\sigma$
- or responds NONE, if there are no such schedule and process.

After a finite number of queries, the prover ends phase r by choosing a one round schedule σ_r applicable to C_{r-1} in which at most f processes crash and defines $C_r = C_{r-1}\sigma_r$. Then the players proceed to phase $r+1$.

In phase t, the prover asks what values are output by processes at the end of one round executions from configuration C_{t-1}. A *query* $(C_{t-1}, \sigma'_t, p_i)$ consists of a one round schedule σ'_t applicable to C_{t-1}, in which at most f processes crash, and a process p_i, which is not crashed in configuration $C_{t-1}\sigma'_t$. The algorithm responds with the value output by p_i in configuration $C_{t-1}\sigma'_t$.

At the end of phase t, the prover chooses a one round schedule σ_t applicable to C_{t-1} in which at most f processes crash and defines $C_t = C_{t-1}\sigma_t$.

Finally the winner of the interaction is determined. If the outputs in configuration C_t violate the specifications of task $\mathscr{T}$, then the prover wins. If there is contradictory information from the algorithm, then the prover also wins. Otherwise, the prover loses.

An Extension-Based Proof of the Lower Bound for Binary Consensus

Binary consensus is another name for k-set agreement when $k = 1$. As an example, we present an extension-based proof that any algorithm solving binary consensus among $n \geq t+2$ processes requires at least $t+1$ rounds when at most one process crashes each round. It is based on the valency arguments used to prove this result [1, 5, 13].

Suppose that $\mathcal{A}$ is t round algorithm that claims to solve consensus among $n \geq t+2$ processes. We construct a prover that wins against $\mathcal{A}$. Note that, if $\mathcal{A}$ responds negatively to both $(C, Q, u, 0)$ and $(C, Q, u, 1)$ for some initial configuration C, bound $u \in \{0, 1\}$, and set Q of at least $n-u$ processes, then it has provided contradictory information. Specifically, in the t round execution from C in which no processes crash, every process must output 0 or 1. Likewise, if $\mathcal{A}$ responds negatively to both $(C_{r-1}, \sigma'_r, u, 0)$ and $(C_{r-1}, \sigma'_r, u, 1)$, where σ'_r is a one round schedule applicable to C_{r-1}, then it has provided contradictory information. Since providing contradictory information causes $\mathcal{A}$ to lose, we'll suppose that $\mathcal{A}$ never does this.

In phase 0, the prover will try to find a bivalent initial configuration. The prover asks the queries $(D_j, P, 1, 0)$ and $(D_j, P, 1, 1)$, for all $0 \leq j \leq n$, where D_j is the initial configuration in which the first j processes, $p_0, \ldots, p_{j-1}$, have input 0 and the rest have input 1. If there exists j such that $\mathcal{A}$ responds positively to both queries, the prover chooses $C_0 = D_j$ and proceeds to phase 1.

Otherwise, for each j, there exists $m_j \in \{0,1\}$ such that $\mathcal{A}$ responded positively to $(D_j, P, 1, m_j)$ and responded negatively to $(D_j, P, 1, 1-m_j)$. If $m_0 = 0$, then the algorithm violates validity, since all processes in configuration D_0 have input 1. Similarly, if $m_n = 1$, the algorithm violates validity, since all processes in configuration D_n have input 0. So, assume that $m_0 = 1$ and $m_n = 0$. Then there exists $0 \leq j < n$ such that $m_j = 1$ and $m_{j+1} = 0$. Next, the prover asks the queries $(D_j, Q, 1, 0)$ and $(D_j, Q, 1, 1)$, where $Q = P - \{p_j\}$. Since $Q \subseteq P$ and $\mathcal{A}$ responded negatively to $(D_j, P, 1, 0)$, $\mathcal{A}$ must respond negatively to $(D_j, Q, 1, 0)$ and positively to $(D_j, Q, 1, 1)$ to avoid providing contradictory information. Let σ and $p_i \in Q$ be its response to $(D_j, Q, 1, 1)$. Since D_j and D_{j+1} are indistinguishable to every process in Q and σ is a Q-only schedule, p_i also outputs 1 in $D_{j+1}\sigma$. However, this contradicts the negative response $\mathcal{A}$ gave to the query $(D_{j+1}, P, 1, 1)$. Thus, the prover wins.

In phase r, for $1 \leq r \leq t-1$, the prover tries to find a bivalent configuration reachable from C_{r-1} by a one round schedule. The prover asks the queries $(C_{r-1}, \sigma'_r, 1, 0)$ and $(C_{r-1}, \sigma'_r, 1, 1)$ for every one round schedule σ'_r applicable to C_{r-1} in which at most 1 process crashes. If there exists σ'_r such that $\mathcal{A}$ responds positively to both, then $C_{r-1}\sigma'_r$ is bivalent. In this case, the prover chooses $C_r = C_{r-1}\sigma'_r$ and proceeds to phase $r+1$.

Otherwise, for each σ'_r, there exists $m(\sigma'_r) \in \{0,1\}$ such that $\mathcal{A}$ responded positively to $(C_{r-1}, \sigma'_r, 1, m(\sigma'_r))$ and responded negatively to $(C_{r-1}, \sigma'_r, 1, 1-m(\sigma'_r))$. Let α denote the one round schedule applicable to C_{r-1} in which no processes crash. Consider the set of all one round schedules σ'_r applicable to C_{r-1} in which one process crashes and $m(\sigma'_r) \neq m(\alpha)$. This set is nonempty since C_{r-1} is bivalent. Among all schedules in this set, let β be one in which the process p_i that crashes sends the largest number of messages to the processes that are active in C_{r-1}. Let Q be the set of processes that are active in $C_{r-1}\beta$. If p_i sends a message to every process in Q during β (i.e., $Q \subseteq \beta[i]$), then $\mathcal{A}$ has provided contradictory information. Specifically, $C_{r-1}\alpha$ and $C_{r-1}\beta$ are indistinguishable to every process in Q, but $m(\alpha) \neq m(\beta)$. So, suppose there is some process $p_\ell \in Q$ to which p_i does not send a message in β. Let β' be the one round schedule applicable to C_{r-1} that is the same as β except that p_i also sends a message to p_ℓ. Then, by definition of β, it follows that $m(\beta') \neq m(\beta)$. In this case, $\mathcal{A}$ has provided contradictory information, since $C_{r-1}\beta'$ and $C_{r-1}\beta$ are indistinguishable to all processes in $Q - \{p_\ell\}$. Thus, the prover wins.

In phase t, the prover asks the queries $(C_{t-1}, \sigma'_t, p_i)$ for all one round schedules σ'_t applicable to C_{t-1} in which at most 1 process crashes and all processes p_i which have not crashed in $C_{t-1}\sigma'_t$. If $\mathcal{A}$ responded with both 0 and 1 as output values in $C_{t-1}\sigma'_t$, for some one round schedule σ'_t, then the prover wins, since $\mathcal{A}$ has violated 1-agreement.

Otherwise, for each σ'_t, there exists $m(\sigma'_t) \in \{0,1\}$ such that $\mathcal{A}$ responded with $m(\sigma'_t)$ to the queries $(C_{t-1}, \sigma'_t, p_i)$ for all processes p_i which have not crashed. Let α denote the one round schedule applicable to C_{t-1} in which no processes crash. Consider the set of all one round schedules σ'_t applicable to C_{t-1} in which one process crashes and $m(\sigma'_t) \neq m(\alpha)$. This set is nonempty since C_{t-1} is bivalent. Among all schedules in this set, let β be one in which the process p_i that crashes sends the largest number of messages to the processes that are active in C_{t-1}. Let Q be the set of processes excluding p_i that are active in C_{t-1}. If p_i sends a message to every process in Q during β (i.e., $Q \subseteq \beta[i]$), then $\mathcal{A}$ has provided contradictory information. Specifically, $C_{t-1}\alpha$ and $C_{t-1}\beta$ are indistinguishable to every process in Q, but $m(\alpha) \neq m(\beta)$. So, suppose there is some process $p_\ell \in Q$ to which p_i does not send a message in β. Let β' be the one round schedule applicable to C_{r-1} that is the same as β except that p_i also sends a message to p_ℓ. Then, by definition of β, it follows that $m(\beta') \neq m(\beta)$. In this case, $\mathcal{A}$ has provided contradictory information, since $C_{t-1}\beta'$ and $C_{t-1}\beta$ are indistinguishable to all processes in $Q - \{p_\ell\}$. Thus, the prover wins.

5 Why Extension-Based Lower Bounds Fail

This section is devoted to proving the main result of the paper.

▶ **Theorem 3.** *For $k, t \geq 2$, there is no extension-based proof of a lower bound of $t+1$ rounds for solving k-set agreement among $n = k(t+1)+1$ processes when each process has an input in $\{0, \ldots, k\}$ and at most k processes can crash each round.*

To prove this result, we construct a t round adversarial algorithm $\mathcal{A}$ that is able to win against every extension-based prover.

We begin by presenting some terminology and notation used to describe $\mathcal{A}$. For any configuration C and any process p_i that is not crashed in C, let $adopt(C, p_i)$ be the smallest input value process p_i saw in the execution leading to configuration C. If C is an initial configuration, then $adopt(C, p_i) = x_i$, the input value of process p_i. If $C' = C\sigma$, where σ is a one round schedule, and p_i is not crashed in C', then $\{adopt(C, p_j) \mid p_i \in \sigma[j]\}$ is the *set of adopted values that p_i saw during σ*, either because the value was adopted by p_i in C or the value was adopted by some other process that sent a message to p_i during σ. As in FloodMin, $adopt(C', p_i) = \min\{adopt(C, p_j) \mid p_i \in \sigma[j]\}$. For any configuration C, any set of processes Q that are active in C, and any possible value $m \in M = \{0, \ldots, k\}$, let

$$adopt(C, Q) = \{adopt(C, q) \mid q \in Q\}$$

be the *set of values adopted in C by processes in Q*.

The following one round schedules will be useful for defining our adversarial algorithm. Let C be any configuration, let Q be any subset of the active processes in C, and let $m \in M$ be any possible output value.

- In $\alpha(C, Q)$, processes in Q do not crash and all other active processes crash without sending any messages:

$$\alpha(C,Q)[i] = \begin{cases} \{p_0, \ldots, p_n\} & \text{if } p_i \in Q \\ \{p_i\} & \text{if } p_i \text{ is active in } C, \text{ but } p_i \notin Q \\ \phi & \text{if } p_i \text{ is not active in } C. \end{cases}$$

- In $\beta(C, Q, m)$, processes in Q that have adopted a value other than m in C do not crash and all other active processes crash without sending any messages:

$$\beta(C,Q,m)[i] = \begin{cases} \{p_0, \ldots, p_n\} & \text{if } p_i \in Q \text{ and } adopt(C, p_i) \neq m \\ \{p_i\} & \text{if } p_i \text{ is active in } C, \text{ but } p_i \notin Q \text{ or } adopt(C, p_i) = m \\ \phi & \text{if } p_i \text{ is not active in } C. \end{cases}$$

  If $m \notin adopt(C, Q)$, then $\beta(C, Q, m)$ is the same as $\alpha(C, Q)$. Otherwise, it is like $\alpha(C, Q)$, except that the processes in Q that have adopted m in C crash before sending any messages.

- In $\beta(C, Q, <m)$, processes in Q that have adopted a value greater than or equal to m in C do not crash and all other active processes crash without sending any messages:

$$\beta(C,Q,<m)[i] = \begin{cases} \{p_0, \ldots, p_n\} & \text{if } p_i \in Q \text{ and } adopt(C, p_i) \geq m \\ \{p_i\} & \text{if } p_i \text{ is active in } C, \text{ but } p_i \notin Q \text{ or } adopt(C, p_i) < m \\ \phi & \text{if } p_i \text{ is not active in } C. \end{cases}$$

  If $adopt(C, Q) \subseteq \{m, m+1, \ldots, k\}$, then $\beta(C, Q, m)$ is the same as $\alpha(C, Q)$. Otherwise, it is like $\alpha(C, Q)$, except that the processes in Q that have adopted values less than m in C crash before sending any messages.

If C is a configuration at the end of round $r < t-1$ and γ is a one round schedule starting from C, then γ^* is the $t-r$ round schedule where the first round is γ and no processes crash in the remaining $t-r-1$ rounds.

Algorithm $\mathcal{B}(m^*)$

Before defining the adversarial algorithm $\mathcal{A}$, we consider $k+1$ different bad algorithms, $\mathcal{B}(m^*)$, one for each value $m^* \in M$. Except for what they output, processes in $\mathcal{B}(m^*)$ behave as in FloodMin, with each active process repeatedly trying to send its adopted value to all other processes and adopting the smallest value it saw. At the end of round t, every process p_i that has not crashed will output a value (but not necessarily the smallest value) that it saw in the last round. Specifically, let a_i be the value that p_i adopted at the end of round $t-1$. Then p_i outputs a_i, except in two special cases.

- During round t, if p_i received at least $k+1$ messages, no message with value a_i, and at least one message with every other value, then it outputs the smallest value it saw at least twice.
- If $a_i = m^*$ and p_i saw each value in M exactly once during round t, then it outputs $(m^*+1) \bmod (k+1)$.
- Otherwise, p_i outputs a_i.

Note that, if p_i saw each value in M exactly once, then it received exactly k messages, so these two cases are mutually exclusive.

Since the set of adopted values at each round is a subset of the input values, every output value is an input value. Hence $\mathcal{B}(m^*)$ satisfies validity.

Consider any configuration C'_{t-1} reachable from an initial configuration by a $t-1$ round schedule in which at most k processes crash each round, any one round schedule σ'_t applicable to C'_{t-1} in which at most k processes crash, and any process p_i that has not crashed in $C'_t = C'_{t-1}\sigma'_t$. The following three observations are consequences of the definition of $\mathcal{B}(m^*)$. They are true because neither of the special cases are applicable.

▶ **Observation 4.** *In algorithm $\mathcal{B}(m^*)$, if process p_i saw at most k different values during σ'_t, then p_i outputs $adopt(C'_{t-1}, p_i)$ in configuration C'_t.*

▶ **Observation 5.** *In algorithm $\mathcal{B}(m^*)$, if process p_i saw the value it adopted in C'_{t-1} at least twice during σ'_t, then p_i outputs $adopt(C'_{t-1}, p_i)$ in configuration C'_t.*

▶ **Observation 6.** *Let $m \in M$. In algorithm $\mathcal{B}(m^*)$, if process p_i did not see m during round σ'_t, then p_i does not output m in configuration C'_t.*

Here is another useful property of this algorithm.

▶ **Lemma 7.** *Let $m \in M$. In algorithm $\mathcal{B}(m^*)$, if process p_i saw m at most once, saw every other value in M at least once, and received at least $k+1$ messages during σ'_t, then p_i does not output m in configuration C'_t.*

Proof. If p_i did not see m during σ'_t, then, by Observation 6, it does not output m in C'_t. So, suppose that p_i saw m exactly once during σ'_t. Also suppose that, during σ'_t, p_i saw every other value in M at least once and it received at least $k+1$ messages.

If p_i saw the value it adopted in C'_{t-1} at least twice during σ'_t, then, by Observation 5, it outputs $adopt(C'_{t-1}, \sigma'_t) \neq m$ in configuration C'_t. Therefore, suppose that p_i saw the value it adopted in C'_{t-1} only once during σ'_t. Then, during σ'_t, it received no message with this value and received messages with every other value. Since the conditions of the first special case are satisfied, p_i outputs the smallest value it saw at least twice, which is not m. ◀

If at most k different values were adopted at the end of round $t-1$, it doesn't matter which of these values a process outputs. So suppose that each of the $k+1$ values in M was adopted by some process at the end of round $t-1$. By Lemmas 1 and 2, this can only occur if k processes crashed in each of the first $t-1$ rounds. Hence, there are $2k+1$ active processes at the end of round $t-1$. At least one value in M was adopted by at least two active processes. Furthermore, at most k values in M were adopted by at least two active processes. Thus, a process can output a value that it sees at least twice during round t. It is also possible that k processes crash at the beginning of round t and each of the remaining $k+1$ processes sees each of the $k+1$ values in M exactly once during round t. In $\mathcal{B}(m^*)$, this symmetry is broken by having the process that had adopted m^* at the end of round $t-1$ output a different value. If m^* is a value that was adopted by at most one process at the end of round $t-1$, then these two approaches don't interfere with one another. However, for each $m^* \in M$, there is some final configuration of algorithm $\mathcal{B}(m^*)$ in which all $k+1$ values in M are output. We explicitly construct such a configuration in the appendix.

The Adversarial Algorithm

During phases 0 through $t-1$, the adversarial algorithm $\mathcal{A}$ responds to each query in a way that is consistent with all these bad algorithms. At the end of phase $t-1$, the adversarial algorithm chooses one of these bad algorithms and responds as if it is that algorithm during phase t. Its choice depends on the prover's choice of configuration C_{t-1}. The chosen bad algorithm has the property that it does not violate *k-agreement* in any final configuration reachable from C_{t-1}. Next, we give detailed specifications for how the adversarial algorithm responds to queries in each phase. Then we show that at most k different values are output in the configuration chosen by the prover at the end of phase t. Finally, to show that $\mathcal{A}$ never answers queries inconsistently, we prove that $\mathcal{A}$ responded to every query in a way that is consistent with the bad algorithm it chose.

Phase 0

Consider any query (C, Q, u, m) made by the prover during phase 0, where C is an initial configuration, Q is a set of processes, $u \in \{0, \ldots, k\}$ is an upper bound on the number of crashes per round, and $m \in M$ is a value.

- If m is the input of some process in Q (i.e., $m \in \mathit{adopt}(C, Q)$) and there are at most u processes that are not in Q or have inputs less than m, then $\mathcal{A}$ responds with the schedule $\beta^*(C, Q, <m)$ (in which these processes crash immediately) and the process $p_i \in Q$ with smallest index that has input m in configuration C.
- Otherwise, $\mathcal{A}$ responds with NONE.

The following observations are useful consequences of this specification.

▶ **Observation 8.** *If $\mathcal{A}$ responds to the query (C, Q, u, m) in phase 0 with $\beta^*(C, Q, <m)$ and p_i, then m is the only value p_i saw in the last round of $\beta^*(C, Q, <m)$.*

▶ **Observation 9.** *If $\mathcal{A}$ responds to the query (C, Q, u, m) in phase 0 with* NONE*, then, for every configuration C'_{t-1} reachable from C by a $(t-1)$-round Q-only schedule in which at most u processes crash each round, $m \notin \mathit{adopt}(C'_{t-1}, Q'_{t-1})$, where Q'_{t-1} is the set of active processes in C'_{t-1}.*

Phases $1, \ldots, t-2$

In phase r, where $1 \leq r \leq t-2$, the adversarial algorithm $\mathcal{A}$ behaves similarly. Consider any query $(C_{r-1}, \sigma'_r, u, m)$ made by the prover during phase r, where σ'_r is a one round schedule applicable to C_{r-1} in which at most k processes crash, $u \in \{0, \ldots, k\}$ is an upper bound on the number of crashes per round, and $m \in M$ is a value. Let $C'_r = C_{r-1}\sigma'_r$ and let Q'_r be the set of active processes in C'_r.

- If m is the adopted value of some process in Q'_r (i.e., $m \in \mathit{adopt}(C'_r, Q'_r)$) and there are at most u processes in Q'_r with adopted value less than m, then $\mathcal{A}$ responds with the schedule $\beta^*(C'_r, Q'_r, {<}m)$ (in which these processes crash immediately) and the process $p_i \in Q'_r$ with smallest index that has adopted m in C'_r.
- Otherwise, $\mathcal{A}$ responds with NONE.

The following two observations are analogous to Observation 8 and Observation 9.

▶ **Observation 10.** *If $\mathcal{A}$ responds to the query $(C_{r-1}, \sigma'_r, u, m)$ in phase $1 \leq r \leq t-2$ with $\beta^*(C'_r, Q'_r, {<}m)$ and p_i, where Q'_r is the set of active processes in $C'_r = C'_{r-1}\sigma'_r$, then m is the only value p_i saw in the last round of $\beta^*(C'_r, Q'_r, {<}m)$.*

▶ **Observation 11.** *If $\mathcal{A}$ responds to the query $(C_{r-1}, \sigma'_r, u, m)$ in phase $1 \leq r \leq t-2$ with* NONE*, then, for every configuration C'_{t-1} reachable from $C_{r-1}\sigma'_r$ by a $(t-1-r)$-round schedule in which at most u processes crash each round, $m \notin \mathit{adopt}(C'_{t-1}, Q'_{t-1})$, where Q'_{t-1} is the set of active processes in C'_{t-1}.*

Note that, during phases 0 through $t-2$, the adversarial algorithm responds in a way that is consistent with FloodMin.

Phase $t-1$

In phase $t-1$, the adversarial algorithm's strategy is different. It depends on the query and the prover's choice for C_{t-2}. Consider any query $(C_{t-2}, \sigma'_{t-1}, u, m)$ made by the prover during phase $t-1$, where σ'_{t-1} is a one round schedule applicable to C_{t-2} in which at most k processes crash, $u \in \{0, \ldots, k\}$ is an upper bound on the number of crashes per round, and $m \in M$ is a value. Let $C'_{t-1} = C_{t-2}\sigma'_{t-1}$ and let Q'_{t-1} be the set of active processes in C'_{t-1}. The response chosen by $\mathcal{A}$ depends on the values adopted by these processes in configuration C'_{t-1}.

- If $m \notin \mathit{adopt}(C'_{t-1}, Q'_{t-1})$, then $\mathcal{A}$ responds NONE.

So, suppose $m \in \mathit{adopt}(C'_{t-1}, Q'_{t-1})$. Let $i = \min\{j \mid p_j \in Q'_{t-1} \text{ and } \mathit{adopt}(C'_{t-1}, p_j) = m\}$ be the smallest index of a process that adopted m. Let $m' \in M - \{m\}$ be the smallest value other than m that was adopted in C'_{t-1} by the fewest number of processes. In particular, it is possible that $m' \notin \mathit{adopt}(C'_{t-1}, Q'_{t-1})$.

- If at most u processes adopted m', then $\mathcal{A}$ responds with the one round schedule $\beta(C'_{t-1}, Q'_{t-1}, m')$ (in which these processes crash immediately) and process p_i.
- If more than u processes adopted m' and at least 2 processes adopted m, then $\mathcal{A}$ responds with the one round failure-free schedule $\alpha(C'_{t-1}, Q'_{t-1})$ and process p_i.
- Otherwise, $\mathcal{A}$ responds NONE.

The information communicated by positive and negative responses in phase $t-1$ is different than in the previous phases.

▶ **Lemma 12.** *If $\mathcal{A}$ responds to the query $(C_{t-2}, \sigma'_{t-1}, u, m)$ in phase $t-1$ with σ and p_i, then p_i had adopted value m in $C_{t-2}\sigma'_{t-1}$ and either p_i saw at most k different values during σ or p_i saw m at least twice during σ.*

Proof. Suppose that $\mathcal{A}$ responds to $(C_{t-2}, \sigma'_{t-1}, u, m)$ in phase $t-1$ with σ and p_i. By construction, $adopt(C_{t-2}\sigma'_{t-1}, p_i) = m$. Let m' be the smallest value other than m that was adopted in $C'_{t-1} = C_{t-2}\sigma'_{t-1}$ by the fewest number of processes in Q'_{t-1}.

If at most u processes adopted m' in C'_{t-1}, none of these processes sent any messages in $\sigma = \beta(C'_{t-1}, Q'_{t-1}, m')$. Then, during σ, p_i did not see m', so it saw at most k different values.

Otherwise, more than u processes adopted m' in C'_{t-1}, at least 2 processes adopted m in C'_{t-1}, and $\sigma = \alpha(C'_{t-1}, Q'_{t-1})$. By definition, no processes crash in σ, so p_i saw m at least twice during σ. ◀

▶ **Lemma 13.** *If $\mathcal{A}$ responds to the query $(C_{t-2}, \sigma'_{t-1}, u, m)$ in phase $t-1$ with* NONE*, then*

- *$m \notin adopt(C'_{t-1}, Q'_{t-1})$ or*
- *exactly one process in Q'_{t-1} adopted m in C'_{t-1} and more than u, but at most 2, processes in Q'_{t-1} adopted m' in C'_{t-1},*

where Q'_{t-1} is the set of active processes in C'_{t-1} and $m' \in M - \{m\}$ is the smallest value other than m that was adopted by the fewest number of processes in C'_{t-1}.

Proof. Suppose that $\mathcal{A}$ responds to $(C_{t-2}, \sigma'_{t-1}, u, m)$ in phase $t-1$ with NONE and $m \in adopt(C'_{t-1}, Q'_{t-1})$. Then, from the specifications of $\mathcal{A}$, more than u processes in Q'_{t-1} adopted m' and less than 2 processes in Q'_{t-1} adopted m in configuration C'_{t-1}. Since $m \in adopt(C'_{t-1}, Q'_{t-1})$, exactly one process in Q'_{t-1} adopted m in C'_{t-1}.

By definition of m', each value $m'' \in M - \{m\}$ was adopted in C'_{t-1} by at least as many processes in Q'_{t-1} as m' was. Since m' was adopted by more than $u \geq 0$ processes in Q'_{t-1}, it follows that $m'' \in adopt(C'_{t-1}, Q'_{t-1})$. Hence $\#adopt(C'_{t-1}, Q'_{t-1}) = k+1$. By Lemma 2, at least k processes crashed in each of the $t-1$ rounds of the execution from C_0 to C'_{t-1}. At most k processes can crash each round, so exactly k processes crashed in each of these rounds and $\#Q'_{t-1} = n - k(t-1) = 2k+1$. Since only one process in Q'_{t-1} has adopted m, the other $2k$ processes in Q'_{t-1} have each adopted one of the k values in $M - \{m\}$. Thus, m', which was adopted by the fewest number of these processes, was adopted by at most 2 of these processes. ◀

Phase t

Let Q_{t-1} be the set of active processes in C_{t-1} and let $m^* \in M$ be the smallest value that was adopted in C_{t-1} by the fewest number of processes in Q_{t-1}. In particular, if $\#adopt(C_{t-1}, Q_{t-1}) \leq k$, then m^* was adopted by no process in Q_{t-1}. However, if $\#adopt(C_{t-1}, Q_{t-1}) = k+1$, then, by Lemma 2, at least k processes crashed in each of the $t-1$ rounds of the execution from C_0 to C_{t-1}. At most k processes can crash each round, so exactly k processes crashed in each of these rounds and $\#Q_{t-1} = n - k(t-1) = 2k+1$. In this case, the number of processes in Q_{t-1} that adopted m^* is at most $\lfloor \#Q_{t-1}/(k+1) \rfloor = \lfloor (2k+1)/(k+1) \rfloor = 1$.

Consider any query $(C_{t-1}, \sigma'_t, p_i)$ made by the prover during phase t, where σ'_t is a one round schedule applicable to C_{t-1} in which at most k processes crash and p_i is a process that has not crashed in $C_{t-1}\sigma'_t$. Then $\mathcal{A}$ responds with $adopt(C_{t-1}, p_i)$, except in two special cases.

- If p_i received at least $k+1$ messages, no message with value $adopt(C_{t-1}, p_i)$, and at least one message with every other value, then $\mathcal{A}$ responds with the smallest value p_i saw at least twice during σ'_t.
- If $adopt(C_{t-1}, p_i) = m^*$ and p_i saw each value in M exactly once during σ'_t then $\mathcal{A}$ responds with $(m^*+1) \bmod (k+1)$.

- Otherwise, $\mathcal{A}$ responds with $adopt(C_{t-1}, p_i)$.

Note that, during σ'_t, if p_i saw each value in M exactly once, then it received exactly k messages, so the two special cases are mutually exclusive.

When $\mathcal{A}$ responds to a query with value m^*, the execution has special properties.

▶ **Lemma 14.** *If $\mathcal{A}$ responds to the query $(C_{t-1}, \sigma'_t, p_i)$ in phase t with m^*, then $adopt(C_{t-1}, p_i) = m^*$ and p_i saw at most k different values during σ'_t.*

Proof. Since at most one process in Q_{t-1} adopted m^* in C_{t-1}, process p_i saw m^* at most once during σ'_t, so $\mathcal{A}$ does not respond with m^* as a result of the first special case. Since $m^* \neq (m^* + 1) \bmod k$, $\mathcal{A}$ does not respond with m^* as a result of the second special case. Thus, $adopt(C_{t-1}, p_i) = m^*$.

To obtain a contradiction, suppose that p_i saw all $k+1$ values during σ'_t. Since the second special case does not hold, there is a value $m \in M$ that p_i saw at least twice during σ'_t. Hence, p_i received at least $k+1$ messages during σ'_t. Since p_i is the only process in Q_{t-1} that adopted m^* in C_{t-1}, it did not receive a message with value $m^* = adopt(C_{t-1}, p_i)$, but it did receive at least one message with every other value. But then the first special case holds, which is a contradiction. Hence, p_i saw at most k different values during σ'_t. ◀

The following observation is a consequence of the specifications, since neither of the special cases is applicable.

▶ **Observation 15.** *If p_i saw the value it adopted in C_{t-1} at least twice during σ'_t, then $\mathcal{A}$ responds to the query $(C_{t-1}, \sigma'_t, p_i)$ in phase t with $adopt(C_{t-1}, p_i)$.*

Agreement

Let C_t be the the final configuration chosen by the prover at the end of phase t and let σ_t be the one round schedule such that $C_t = C_{t-1}\sigma_t$. We show that $\mathcal{A}$ does not lose by violating k-agreement in C_t.

▶ **Lemma 16.** *Consider the set of processes Q_t that output values in C_t. $\mathcal{A}$ responds to the queries (C_{t-1}, σ_t, p_i) for $p_i \in Q_t$ with at most k different values.*

Proof. Suppose not. For each $m \in M$, let $q_m \in Q_t$ be the process with smallest index that outputs m. By Lemma 14, process q_{m^*} saw at most k different values during σ_t. During σ_t, process q_{m^*} received a message from every process in Q_t, so $\#adopt(C_{t-1}, Q_t)$ is bounded above by the number of different values q_{m^*} saw during σ_t. Thus $adopt(C_{t-1}, Q_t) \subsetneq M$. Consider the sequence $m_0, m_1, \ldots, m_{k+1}$, where

- $m_0 = \min(M - adopt(C_{t-1}, Q_t))$ is the smallest value not in $adopt(C_{t-1}, Q_t)$ and
- $m_i = adopt(C_{t-1}, q_{m_{i-1}}) \in M$ is the value adopted by $q_{m_{i-1}}$ in C_{t-1}, for all $i \geq 1$.

Since $\#M = k+1$, the sequence contains at least one duplicate. Let j be the smallest positive integer such that $m_j = m_i$ for some $i < j$.

Note that $m_i \neq m_0$, since $m_0 \notin adopt(C_{t-1}, Q_t)$, but $m_j = adopt(C_{t-1}, q_{m_{j-1}}) \in adopt(C_{t-1}, Q_t)$. By the minimality of j, $m_{j-1} \neq m_{i-1}$, so $q_{m_{j-1}}$ and $q_{m_{i-1}}$ output different values in C_t, but they adopted the same value $m_j = m_i$ in C_{t-1}. During σ_t, processes $q_{m_{i-1}}$ and $q_{m_{j-1}}$ receive a message with this value from one another. Therefore both $q_{m_{j-1}}$ and $q_{m_{i-1}}$ saw m_j at least twice during σ_t. By Observation 15, both these processes output the values they had adopted in C_{t-1}. Hence $m_{j-1} = adopt(C_{t-1}, q_{m_{j-1}}) = m_j = m_i = adopt(C_{t-1}, q_{m_{i-1}}) = m_{i-1}$, which is a contradiction. ◀

Consistency

To show that the responses of the adversarial algorithm $\mathcal{A}$ to the queries do not contradict one another, we show that they are all consistent with algorithm $\mathcal{B}(m^*)$, where m^* is the smallest value adopted by the fewest number of processes in C_{t-1}, the configuration chosen by the prover at the end of phase $t-1$. Note that the choice of m^* depends on the choices made by the prover.

First, we show that, whenever $\mathcal{A}$ responded positively to a query, $\mathcal{B}(m^*)$ can give the same response. The queries made in each phase are considered separately.

▶ **Lemma 17.** *If $\mathcal{A}$ responded to a query with a schedule and a process (in phases 0 to $t-1$) or with a value (in phase t), then that response is consistent with algorithm $\mathcal{B}(m^*)$.*

Proof. Suppose the prover asked query (C, Q, u, m) in phase 0 and $\mathcal{A}$ responded with $\beta^*(C, Q, <m)$ and p_i. Let σ'_t be the last round of this schedule and let C'_{t-1} be the second last configuration in the execution of $\beta^*(C, Q, <m)$ from C. Then $C\beta^*(C, Q, <m) = C'_{t-1}\sigma'_t$. By Observation 8, m was the only value p_i saw in σ'_t. In particular, $adopt(C'_{t-1}, p_i) = m$. Then, by Observation 4, in algorithm $\mathcal{B}(m^*)$, process p_i outputs $adopt(C'_{t-1}, p_i) = m$ in configuration $C'_{t-1}\sigma'_t$.

Suppose the prover asked query $(C_{r-1}, \sigma'_r, u, m)$ in phase r, where $1 \leq r \leq t-2$, and $\mathcal{A}$ responded with $\beta^*(C'_r, Q'_r, <m)$ and p_i. Then $C'_r = C_{r-1}\sigma'_r$ and Q'_r is the set of active processes in C'_r. Let σ'_t be the last round of the schedule $\beta^*(C'_r, Q'_r, <m)$ and let C'_{t-1} be the second last configuration in the execution of $\beta^*(C'_r, Q'_r, <m)$ from C'_r. By Observation 10, m is the only value p_i saw in σ'_t. This implies that $adopt(C'_{t-1}, p_i) = m$ and, hence, by Observation 4, in algorithm $\mathcal{B}(m^*)$, p_i outputs m in configuration $C'_{t-1}\sigma'_t$.

Suppose the prover asked query $(C_{t-2}, \sigma'_{t-1}, u, m)$ in phase $t-1$. Let $C'_{t-1} = C_{t-2}\sigma'_{t-1}$, let Q'_{t-1} be the set of active processes in C'_{t-1}, and let $m' \in M - \{m\}$ be the smallest value other than m that was adopted by the fewest number of processes in C'_{t-1}.

If $\mathcal{A}$ responded with $\beta(C'_{t-1}, Q'_{t-1}, m')$ and p_i, then at most u processes in Q'_{t-1} adopted m' in configuration C'_{t-1} and $adopt(C'_{t-1}, p_i) = m$. Since all processes in Q'_{t-1} that adopted value m' crash without sending any messages in $\beta(C'_{t-1}, Q'_{t-1}, m')$, it follows that, during $\beta(C'_{t-1}, Q'_{t-1}, m')$, process p_i did not see m' and, hence, saw at most k different values. By Observation 4, in algorithm $\mathcal{B}(m^*)$, process p_i outputs $adopt(C'_{t-1}, p_i) = m$ in configuration $C'_{t-1}\sigma'_t$.

If $\mathcal{A}$ responded with $\alpha(C'_{t-1}, Q'_{t-1})$ and p_i, then at least two processes adopted m and $adopt(C'_{t-1}, p_i) = m$. Since no processes crash in $\alpha(C'_{t-1}, Q'_{t-1})$, process p_i received a message from every other process in Q'_{t-1}, so p_i saw m at least twice during $\alpha(C'_{t-1}, Q'_{t-1})$. By Observation 5, in algorithm $\mathcal{B}(m^*)$, process p_i outputs $adopt(C'_{t-1}, p_i) = m$ in configuration $C'_{t-1}\sigma'_t$.

Finally, suppose the prover asked query $(C_{t-1}, \sigma'_t, p_i)$ in phase t and $\mathcal{A}$ responded with m. Since the same three cases also occur in the specification of $\mathcal{B}(m^*)$, p_i outputs m in configuration $C_{t-1}\sigma'_t$ of algorithm $\mathcal{B}(m^*)$. ◀

Next, we show that whenever $\mathcal{A}$ responded negatively to a query, $\mathcal{B}(m^*)$ also responds negatively. Again, we consider the queries made in each phase separately.

▶ **Lemma 18.** *If $\mathcal{A}$ responded to a query with* NONE*, then algorithm $\mathcal{B}(m^*)$ responds to the query with* NONE*.*

Proof. Suppose the prover asked query (C, Q, u, m) in phase 0 and $\mathcal{A}$ responded with NONE. Then, by Observation 9, for every configuration C'_{t-1} reachable from C by a $(t-1)$-round Q-only schedule in which at most u processes crash each round, $m \notin adopt(C'_{t-1}, Q'_{t-1})$, where Q'_{t-1} is the set of active processes in C'_{t-1}. Therefore, for every one round schedule σ'_t applicable to C'_{t-1} in which at most u processes crash, every process that is not crashed in $C'_{t-1}\sigma'_t$ did not see m during σ'_t. By Observation 6, in algorithm $\mathcal{B}(m^*)$, p_i does not output m in configuration $C'_{t-1}\sigma'_t$. Hence, $\mathcal{B}(m^*)$ also responds with NONE to the query (C, Q, u, m) in phase 0.

Suppose the prover asked query $(C_{r-1}, \sigma'_r, u, m)$ in phase r, where $1 \leq r \leq t-2$, and $\mathcal{A}$ responded with NONE. Then, by Observation 11, for every configuration C'_{t-1} reachable from $C_{r-1}\sigma'_r$ by a $(t-1-r)$-round schedule in which at most u processes crash each round, $m \notin adopt(C'_{t-1}, Q'_{t-1})$, where Q'_{t-1} is the set of active processes in C'_{t-1}. Therefore, for every one round schedule σ'_t applicable to C'_{t-1} in which at most u processes crash, every process that is not crashed in $C'_{t-1}\sigma'_t$ did not see m during σ'_t. By Observation 6, in algorithm $\mathcal{B}(m^*)$, p_i does not output m in configuration $C'_{t-1}\sigma'_t$. Hence, $\mathcal{B}(m^*)$ also responds with NONE to the query $(C_{r-1}, \sigma'_r, u, m)$ in phase r.

Suppose the prover asked query $(C_{t-2}, \sigma'_{t-1}, u, m)$ in phase $t-1$ and $\mathcal{A}$ responded with NONE. Let $C'_{t-1} = C_{t-2}\sigma'_{t-1}$ and let Q'_{t-1} be the set of active processes in C'_{t-1}. If $m \notin adopt(C'_{t-1}, Q'_{t-1})$, then, for every one round schedule σ'_t applicable to C'_{t-1} in which at most u processes crash, every process that is not crashed in $C'_{t-1}\sigma'_t$ did not see m during σ'_t. By Observation 6, in algorithm $\mathcal{B}(m^*)$, p_i does not output m in configuration $C'_{t-1}\sigma'_t$. Hence $\mathcal{B}(m^*)$ also responds with NONE to the query $(C_{t-1}, \sigma'_t, u, m)$ in phase $t-1$.

Therefore, suppose that $m \in adopt(C'_{t-1}, Q'_{t-1})$. By Lemma 13, exactly one process in Q'_{t-1} adopted m in configuration C'_{t-1} and more than u, but at most 2, processes in Q'_{t-1} adopted m' in configuration C'_{t-1}, where $m' \in M - \{m\}$ is the smallest value other than m that was adopted by the fewest number of processes in C'_{t-1}. Let σ'_t be an arbitrary one round schedule applicable to C'_{t-1} in which at most u processes crash and let p_i be a process that is not crashed in $C'_{t-1}\sigma'_t$. Since only one process in Q'_{t-1} adopted m in configuration C'_{t-1}, process p_i saw m at most once during σ'_t.

By definition of m', each value $m'' \in M - \{m\}$ was adopted in C'_{t-1} by at least as many processes in Q'_{t-1} as m' was. Since m' was adopted by more than $u \geq 0$ processes in Q'_{t-1} and at most u processes crash in σ'_t, process p_i saw m'' during σ'_t.

At most k processes crash in each of the first $t-1$ rounds and at most $u < 2$ processes crash in σ'_t, so p_i receives at least $n - 1 - k(t-1) - 1 = 2k - 1$ messages. Note that $2k - 1 \geq k + 1$ since $k \geq 2$. Hence, by Lemma 7, in algorithm $\mathcal{B}(m^*)$, process p_i does not output m in configuration $C'_{t-1}\sigma'_t$. Therefore, $\mathcal{B}(m^*)$ also responds with NONE to the query $(C_{t-1}, \sigma'_t, u, m)$ in phase $t-1$. ◀

6 Conclusions

In this paper, we define the class of extension-based proofs for synchronous message passing models and study the power of such proofs. On one hand, we give an an extension-based proof of the t round lower bound for solving binary consensus among $n \geq t+1$ processes when at most one process can crash each round. On the other hand, we show that, for $k \geq 2$ and $t > 2$, there is no extension-based proof of the t round lower bound for solving k-set agreement among $n = kt + 1$ processes when at most k processes can crash each round.

There are a number of problems that remain open. First, is there an extension-based proof of the t round lower bound for solving k-set agreement among $n > kt + 1$ processes if at most k processes can crash each round, for $k \geq 2$? If so, what is the smallest value of n for

which such a proof exists? A related problem is whether there is an extension-based proof of the t round lower bound for solving k-set agreement among among $n = kt + 1$ processes when any number of processes can crash in each round.

There is a simple 1 round lower bound for solving k-set agreement among $n \geq k + 1$ processes. Without any communication, every process must output its input value to ensure validity. Hence, in any initial configuration in which all $k + 1$ values in M appear as inputs, either validity or k-agreement is violated.

Is there an extension-based proof of the 2 round lower bound for solving k-set agreement among $n = 2k + 1$ processes? The proof of Theorem 3 does not work in this case, since Observation 8 and Observation 9 do not always hold. The reason it may be difficult to extend the result to include this case is that there are more queries the prover can ask in phase $t - 1$ when $t = 1$ than when $t > 1$. In phase 0, for any bound u, value m, and initial configuration C'_0, the prover can specify a set Q of at most $n - u$ processes and ask whether there is a one round Q-only schedule in which at most u processes crash and some process outputs m when applied to configuration C'_0. However, when $t > 1$, for any bound u, value m, and configuration $C'_{t-1} = C_{t-2}\sigma'_{t-1}$, in phase $t - 1$, the prover can only ask whether there is a one round schedule in which at most u processes crash and some process outputs m when applied to configuration C'_{t-1}. In particular, for $u > 0$, the prover cannot specify a subset Q of processes that must crash at the beginning of the one round schedule. This restriction gives the adversary more flexibility when choosing what to answer, which we take advantage of in our proof.

Our choice of allowable queries in the definition of extension-based proofs was influenced by the valency argument showing the lower bound for consensus. Specifically, to make extension-based proofs interesting in the synchronous message passing model, this valency argument should be able to be expressed as an extension-based proof. Although it suffices to only use queries in which there is no additional restriction on the number of processes that can crash each round (i.e., $u = k$), also allowing queries with smaller values of u makes the prover stronger and, hence, makes Theorem 3 better. Other definitions for extension-based proofs are certainly possible. It would be interesting to see if a more restricted class of queries allows Theorem 3 to be extended to $n > k(t + 1) + 1$ or $t = 1$, or makes its proof significantly easier.

Finally, we would like to show that there are other problems for which extension-based proofs cannot be used to obtain known lower bounds on the number of rounds necessary to solve them.

References

1. Marcos Kawazoe Aguilera and Sam Toueg. A simple bivalency proof that t-resilient consensus requires $t + 1$ rounds. *Inf. Process. Lett.*, 71(3-4):155–158, 1999.
2. Dan Alistarh, James Aspnes, Faith Ellen, Rati Gelashvili, and Leqi Zhu. Why extension-based proofs fail. In *Proceedings of the 51st Annual ACM Symposium on Theory of Computing (STOC)*, pages 986–996, 2019.
3. Dan Alistarh, James Aspnes, Faith Ellen, Rati Gelashvili, and Leqi Zhu. Brief announcement: Why extension-based proofs fail. In *Proceedings of the 39th ACM Symposium on Principles of Distributed Computing (PODC)*, pages 54–56, 2020.
4. Hagit Attiya, Armando Castañeda, and Sergio Rajsbaum. Locally solvable tasks and the limitations of valency arguments. In *Proceedings of the 24th International Conference on Principles of Distributed Systems (OPODIS)*, volume 184 of *LIPIcs*, pages 18:1–18:16, 2020.
5. Hagit Attiya and Faith Ellen. *Impossibility Results for Distributed Computing*. Synthesis Lectures on Distributed Computing Theory. Morgan & Claypool Publishers, 2014.

6 Elizabeth Borowsky and Eli Gafni. Generalized FLP impossibility result for t-resilient asynchronous computations. In *Proceedings of the 25th ACM Symposium on Theory of Computing (STOC)*, pages 91–100, 1993.

7 Soma Chaudhuri, Maurice Herlihy, Nancy A. Lynch, and Mark R. Tuttle. Tight bounds for k-set agreement. *J. ACM*, 47(5):912–943, 2000.

8 Eli Gafni. Round-by-round fault detectors: Unifying synchrony and asynchrony (extended abstract). In *Proceedings of the Seventeenth Annual ACM Symposium on Principles of Distributed Computing (PODC)*, pages 143–152, 1998.

9 Maurice Herlihy, Sergio Rajsbaum, and Mark R. Tuttle. An overview of synchronous message-passing and topology. *Electron. Notes Theor. Comput. Sci.*, 39(2):1–17, 2000.

10 Maurice Herlihy, Sergio Rajsbaum, and Mark R. Tuttle. An axiomatic approach to computing the connectivity of synchronous and asynchronous systems. *Electron. Notes Theor. Comput. Sci.*, 230:79–102, 2009.

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

12 Nancy A. Lynch. *Distributed Algorithms*. Morgan Kaufmann, 1996.

13 Yoram Moses and Sergio Rajsbaum. A layered analysis of consensus. *SIAM J. Comput.*, 31(4):989–1021, 2002.

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

A Why Algorithm $\mathcal{B}(m^*)$ Violates k-Agreement

To show that algorithm $\mathcal{B}(m^*)$ violates *k-agreement*, we construct a final configuration C'_t of $\mathcal{B}(m^*)$ in which each of the $k+1$ values in M is output by some process. As in the lower bound for FloodMin, presented at the end of Section 3, we start from an initial configuration in which there are $k+1$ different input values and construct an execution in which k processes crash each round and the set of adopted values has size $k+1$ after each of the first $t-1$ rounds.

Let C'_0 be an initial configuration in which process p_i has input $m \in \{0,\ldots,k-1\}$ for $i = k(t+1)-m$ and has input k for $0 \le i \le kt$. In other words, the first $n-k = kt+1$ process have input k and the last k processes have inputs $k-1$ to 0. Inductively, we construct a configuration C'_r at round r, for $1 \le r \le t-2$, in which process p_i

- adopted $m \in \{0,\ldots,k-1\}$ for $i = k(t+1-r)-m$
- adopted k for $0 \le i \le k(t-r)$, and
- is crashed for $k(t+1-r) < i \le n-1$.

In other words, the first $n-k(r+1) = k(t-r)+1$ process adopted k, the next k processes adopted $k-1$ to 0, and the last kr processes are crashed.

For $1 \le r \le t-2$, consider the one round schedule σ'_r applicable to C'_{r-1} in which

$$\sigma'_r[i] = \begin{cases} \{p_i, p_{i-k}\} & \text{for } k(t+1-r) < i \le k(t+2-r), \\ P & \text{for } 0 \le i \le k(t+1-r), \text{ and,} \\ \phi & \text{for } k(t+2-r) < i < n. \end{cases}$$

In other words, the first $n-kr = k(t-r-1)+1$ processes do not crash and the next k processes each crash after sending a single message, to the process whose index is k less than its own index. Let $C'_r = C'_{r-1}\sigma'_r$. Note that, for $0 \le i \le k(t-r)$, the only messages process p_i received during σ'_r were from the first $k(t-r-1)+1$ processes, all of which had adopted value k in C'_{r-1}, so p_i keeps k as its adopted value in C'_r. However, for $k(t-r) < i \le k(t-r+1)$, process p_i also received a message from process p_{i+k}, which had adopted value $m = k(t+1-(r-1))-(i+k) = k(t+1-r)-i$ in C'_{r-1}, so p_i adopts value m in C'_r.

Consider the one round schedule σ'_{t-1} applicable to C'_{t-2} in which

$$\sigma'_{t-1}[i] = \begin{cases} \{p_i, p_{i-k}, p_0\} & \text{if } m^* \neq k \text{ and } i = 3k - m^*, \\ \{p_i, p_{i-k}\} & \text{for } 2k < i < 3k - m^* \text{ and } 3k - m^* < i \leq 3k, \\ P & \text{for } 0 \leq i \leq 2k, \text{ and,} \\ \phi & \text{for } 3k < i \leq n - 1. \end{cases}$$

In other words, the first $n - k(t-1) = 2k + 1$ processes do not crash, and the next k processes crash after sending one or two messages. These k processes each send a message to the process whose index is k less than its own index. In addition, if $m^* \neq k$, process p_{3k-m^*} also sends a message to process p_0. Let $C'_{t-1} = C'_{t-2}\sigma'_{t-1}$. Note that, for $0 < i \leq k$, the only messages process p_i received during σ'_{t-1} were from the first $2k + 1$ processes, all of which had adopted value k in C'_{t-2}, so p_i keeps k as its adopted value in C'_{t-1}. For $k < i \leq 2k$, process p_i also received a message from process p_{i+k}, which had adopted value $m = k(t + 1 - (r - 1)) - (i + k) = k(t + 1 - r) - i$ in C'_{t-2}, so p_i adopts value m in C'_{t-1}. If $m^* = k$, process p_0 adopts value m^*. If $m^* \neq k$, process p_0 also received a message from p_{3k-m^*}, which had adopted value m^* in C'_{t-2}, so p_0 also adopts value m^*.

Finally, consider the one round schedule σ'_t in which

$$\sigma'_t[i] = \begin{cases} \{p_0, p_{2k-m^*}\} & \text{for } i = 0, \\ \{p_i\} & \text{for } 1 \leq i \leq k - 1, \\ P & \text{for } k \leq i \leq 2k, \text{ and} \\ \phi & \text{for } 2k + 1 \leq i < n - 1. \end{cases}$$

Let $C'_t = C'_{t-1}\sigma'_t$. Note that, for $k \leq i \leq 2k$, process p_i sent the value $2k - i$ it adopted in C'_{t-1} to every other process during σ'_t. Thus, each process p_i saw each value in M at least once during σ'_t. If $i \neq 2k - m^*$, then p_i received no other messages during σ'_t, so it outputs $2k - i \neq m^*$ by the specifications of $\mathcal{B}(m^*)$. If $i = 2k - m^*$, then process p_i also received a message with value m^* from process p_0 during σ'_t, so it outputs $2k - i = m^*$ by the specifications of $\mathcal{B}(m^*)$. Hence, $k + 1$ different values are output in configuration C'_t.