Formally verified asymptotic consensus in robust networks

Mohit Tekriwal, Avi Tachna-Fram, Jean-Baptiste Jeannin, Manos Kapritsos, and Dimitra Panagou

University of Michigan, Ann Arbor, MI 48109, USA
{tmohit, avitf, jeannin, manosk, dpanagou}@umich.edu

Abstract. Distributed architectures are used to improve performance and reliability of various systems. Examples include drone swarms and load-balancing servers. An important capability of a distributed architecture is the ability to reach consensus among all its nodes. Several consensus algorithms have been proposed, and many of these algorithms come with intricate proofs of correctness, that are not mechanically checked. In the controls community, algorithms often achieve consensus *asymptotically*, e.g., for problems such as the design of human control systems, or the analysis of natural systems like bird flocking. This is in contrast to exact consensus algorithm such as Paxos, which have received much more recent attention in the formal methods community.
This paper presents the first formal proof of an asymptotic consensus algorithm, and addresses various challenges in its formalization. Using the Coq proof assistant, we verify the correctness of a widely used consensus algorithm in the distributed controls community, the *Weighted-Mean Subsequence Reduced (W-MSR) algorithm*. We formalize the necessary and sufficient conditions required to achieve resilient asymptotic consensus under the assumed attacker model. During the formalization, we clarify several imprecisions in the paper proof, including an imprecision on quantifiers in the main theorem.

Keywords: Resilient asymptotic consensus · W–MSR algorithm · Network robustness.

1 Introduction

To enhance reliability, robustness and performance, many modern systems use a distributed architecture, composed of multiple nodes communicating with each other. Examples range from coordinated control of multi-robot systems such as swarms of mobile and aerial robots, to load-balancing among servers answering many queries per second. A fully decentralized system, where decisions are made collectively by the nodes rather than by one master node, greatly improves reliability by ensuring there is no single point of failure in the system. A distributed architecture also provides greater performance (depending on the context, in terms of load capacity, reduced latency, smaller communication overhead, etc.)

than any single node could ever achieve. Distributed architectures are supported by distributed algorithms, which particularly focus on carefully handling situations where some nodes become faulty, stop responding, or become malicious.

One central aspect of distributed algorithms is the ability to achieve *consensus.* Consensus is said to be achieved in a network if all normal (correct) nodes agree on a certain value, where a node is *normal* if it is not faulty [34]. The value agreed upon by all nodes can be a reference point for the next position of a swarm, or the sequence of commands executed by a set of replicas in State Machine Replication [44]. Consensus has been studied extensively in different communities. In the distributed computer systems communities, some prominent algorithms achieving consensus are Paxos [29], MultiPaxos [47], Raft [36], and Practical Byzantine Fault Tolerance (PBFT) [6]. However, these algorithms deal with the problem of exact consensus. There are many scenarios where exact consensus is not achievable, ranging from the design of human controlled systems to analysis of natural systems like bird flocking. These problems have to be solved under harsh environmental restrictions such as restricted communication abilities and presence of communication uncertainty. Therefore, these problems warrant the study of *asymptotic consensus* problems, which unlike exact consensus, do not require strong assumptions on the underlying network [16].

This paper presents the first formal proof of an asymptotic consensus algorithm, by formalizing the Weighted-Mean Subsequence Reduced (W-MSR) algorithm [30, 50]. The problem of asymptotic consensus is of much importance to the distributed robotics and controls community, who have studied algorithms like the Mean Subsequence Reduced (MSR) algorithm [27] and its recent extension W-MSR. These algorithms are designed to achieve asymptotic consensus in partially connected groups of nodes, but have not been formally verified. Formal verification of consensus algorithms is important as has been emphasized by the distributed computer systems community, who have long invested in producing mechanically checked proofs of its consensus protocols. The controls community, however, lags behind in this direction. In recent years, the distributed systems community has embraced formal methods to provide *mechanically-checked* proofs of its consensus protocols and their implementations, using a wide range of techniques from interactive and automated theorem proving [48, 25, 8, 5, 18, 9, 31] to automatic generation of inductive invariants [33, 21, 49, 20]. In the distributed robotics and controls community however, researchers usually prove their consensus protocols with paper proofs, using mathematical analysis based on Lyapunov theory and its extensions, without computer-checked formalizations. As we show in this paper, our formalization of asymptotic consensus for the W-MSR algorithm [30] reveals imprecisions in the placement of quantifiers in the main theorem and several missing pieces in the proof, thereby highlighting the importance of machine-checked proofs. Thus a significant contribution of our work is providing the first mechanically checked formalism of the asymptotic consensus and its application to the W-MSR algorithm, widely used in the controls community. We have chosen to formalize this algorithm since it is a widely-used algorithm for resilient consensus [42, 41, 46]. From the perspective of practical applications, enabling resilient consensus in the presence of misbehaving or faulty nodes is desirable for many applications in autonomous systems and robotics, e.g., for coordinated control of multi-robot systems.

The MSR and W-MSR algorithms are very different from exact consensus algorithms such as MultiPaxos, Raft or PBFT. As such our formal verification of the correctness of W-MSR uses different techniques than previous proofs of exact consensus algorithms. The first major difference is that MSR and W-MSR guarantee *asymptotic* consensus rather than finite-time consensus. A second major difference is that MSR and W-MSR provide consensus in networks that are *not fully connected*: two normal nodes might not be able to communicate with each other directly, but might have to rely on another (possibly faulty) node to forward their messages to each other. This last property is crucial to model multi-robot systems where complete communication between any two robots may not be feasible at all times. Because of those differences, providing a mechanically-checked proof of W-MSR requires the development and use of different techniques than the ones typically used to mechanically check Multipaxos, Raft or PBFT. In particular, our formalization crucially relies on formalization of limits and real analysis, because many of the techniques used in model-checking or for generating invariants are not well-suited to prove asymptotic properties.

Contributions: The original contribution of this work is the formalization in the Coq theorem prover of the convergence results of the W-MSR algorithm [30]. Specifically, we provide a machine-checked concrete counterexample for the proof of necessity, a clean proof of Lemma 1 and the Coq formalization of the main theorem (Theorem 1). We also fill in several missing details and clarify imprecisions in the proof of sufficiency, which can be viewed as an addition to the existing proof [30]. Additionally, this is, to our knowledge, the first mechanical formalization of a consensus algorithm where the consensus is obtained asymptotically, opening the door to more such proofs.

This paper is organized as follows. In Section 2, we discuss the problem setup and define terminologies related to graph topology and the W–MSR algorithm [30]. In Section 3, we discuss the formalization of the necessary and sufficient conditions in Coq, for achieving resilient asymptotic consensus. We also discuss some specific challenges we encountered during the formalization. After reviewing some related work in Section 4, we conclude in Section 5 by discussing key takeaways from our work and generic challenges we encountered during the formalization. We also lay down a few directions that could be addressed in future work.

2 Preliminaries

In this paper we consider the problem of formalizing consensus in a network, and adopt the problem formulation from [30]. While the original paper discusses consensus in a distributed control graph for both malicious and byzantine threat models for both time-varying and time-invariant graph structures, we limit our formalization to the case of a *time-invariant graph* for a *malicious threat model* and for a particular threat scope: *F-total*, where the total number of malicious nodes in the control graph is bounded. We will next discuss briefly what each of these highlighted terms means in the context of the following problem.

2.1 Problem formulation

Consider a network that is modeled by a *digraph* (directed graph), $\mathcal{D} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V} = \{1, \ldots, n\}$ is the *node set* and $\mathcal{E} \subset \mathcal{V} \times \mathcal{V}$ is the *directed edge set.* The node set is partitioned into a set of *normal nodes* $\mathcal{N}$, and a set of *adversary nodes* $\mathcal{A}$, which are unknown a priori to the normal nodes. Each directed edge $(j, i) \in \mathcal{E}$ models *information flow* and indicates that node i can be influenced by (or receive information from) node j at time-step t. The set of *in-neighbors* of node i is defined as $\mathcal{V}_i = \{j \in \mathcal{V} | (j, i) \in \mathcal{E}\}$. Intuitively, the set of in-neighbors contains all neighboring nodes of i, such that the direction of information flow is from those nodes to i. The cardinality of the set of in-neighbors is called the *in-degree*, $d_i = |\mathcal{V}_i|$. Since each node has access to its own value at time-step t, we also consider a set of *inclusive neighbors* of node i, denoted by $\mathcal{J}_i = \mathcal{V}_i \cup \{i\}$.

2.2 Threat Model

As discussed earlier, we formalize a threat model (*F-total malicious model* [30]) in which every adversary node in the graph is *malicious*, and there exists an upper bound F on the number of malicious agents in the graph, i.e., the set of adversary nodes are F-totally bounded. In the context of the problem in Section 2.1, some relevant formal definitions pertaining to the threat model are stated as:

Definition 1 (Malicious node [30]). *A node $i \in \mathcal{A}$ is called **Malicious** if it sends the same value $x_i(t)$ to all its neighbors at each time step t, but applies a different update function $f_i'(.)$ at some time step.*

Definition 2 (F-total set [30]). *A set $\mathcal{S} \subset \mathcal{V}$ is **F-total** if it contains at most F nodes in the network, i.e., $|S| \leq F$, $F \in \mathbb{Z}_{\geq 0}$.*

Definition 3 (F-totally bounded [30]). *A set of adversary nodes is **F-totally bounded** if it is an F-total set.*

Note that while Definitions 2 and 3 may appear similar, they define different terminologies. Definition 2 defines an F-total set with at most F nodes in a network. Definition 3 specializes this to a set of adversary nodes saying that there are at most F adversarial nodes in a network.

2.3 Robust network topologies

The ability of a set of normal nodes in a control graph to achieve consensus depends on its ability to make local decisions effectively. Le Blanc et al. [30] defined a topological property called *network robustness* for reasoning about the effectiveness of purely local algorithms to succeed, which we formalize in Coq. In particular, they define a property called (r, s)-robustness, which is stated as:

Definition 4 ((r,s)-robustness [30]). *: A digraph $\mathcal{D} = (\mathcal{V}, \mathcal{E})$ on n nodes ($n \geq 2$) is (r,s)-robust, for nonnegative integers $r \in \mathbb{Z}_{\geq 0}$, $1 \leq s \leq n$, if for every pair of nonempty, disjoint subsets $\mathcal{S}_1$ and $\mathcal{S}_2$ of $\mathcal{V}$ at least one of the following holds (i) $|\mathcal{X}^r_{\mathcal{S}_1}| = |\mathcal{S}_1|$; (ii) $|\mathcal{X}^r_{\mathcal{S}_2}| = |\mathcal{S}_2|$; (iii) $|\mathcal{X}^r_{\mathcal{S}_1}| + |\mathcal{X}^r_{\mathcal{S}_2}| \geq s$, where $\mathcal{X}^r_{\mathcal{S}_k} = \{i \in \mathcal{S}_k : |\mathcal{V}_i \backslash \mathcal{S}_k| \geq r\}$ for $k \in \{1,2\}$.*

The condition (iii) states that there are a total of at least s nodes from the union of sets $\mathcal{S}_1$ and $\mathcal{S}_2$, such that each of those nodes have at least r nodes outside of their respective sets in the union $\mathcal{S}_1 \cup \mathcal{S}_2$. The idea is that "enough" nodes in every pair of nonempty, disjoint sets $\mathcal{S}_1, \mathcal{S}_2 \subset \mathcal{V}$ have at least r neighbors outside of their respective sets. This ensures that the network is well connected, and that loss of information from a node due to malicious attack does not affect the whole network. Figure 1 illustrates an example of a network with $(2,2)$ robustness.

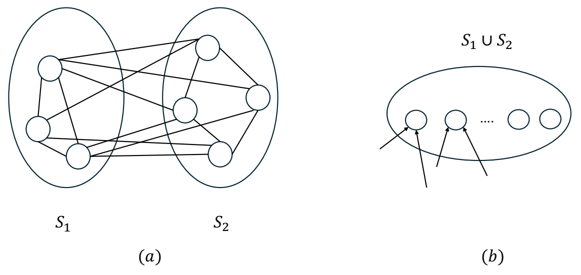

Fig. 1. Illustration for $(2,2)$ robustness. In the illustration (a), every node of the set S_2 has 2 neighboring nodes outside S_2. Similarly every node in the set S_1 has at least 2 neighboring nodes outside S_1. In the illustration (b), there are 2 nodes in the union $S_1 \cup S_2$ that have 2 neighbors outside the set. Note that the sets S_1 and S_2 are disjoint.

2.4 Update model for the normal nodes

In this paper, we formalize a consensus algorithm, called the W–MSR algorithm [30]. This algorithm provides an update model for the normal nodes in the network. A schematic of the algorithm is illustrated in Figure 2. We denote the value emitted by node i at time t as $x_i(t)$, and the value of the directed weighted edge from node j, to node i at time t as $w_{ij}(t)$. The value $x_i(t)$ could represent a measurement like position, velocity, or it could be an optimization variable. The quantity $x^i_j(t)$ is the information that the j^{th} node in the neighboring set of node i sends to the node i. Each node also has a varying set of neighbors which it ignores that we denote as $\mathcal{R}_i(t)$. The set $\mathcal{R}_i(t)$ changes because the nodes are removed depending on their value with respect to the value of node i at time t. In this algorithm, the updated value of a normal node i at time $t+1$ is the convex sum of the values of its neighboring set including itself. Hence, $x_i(t+1) = \sum_{j \in \mathcal{J}_i \backslash \mathcal{R}_i(t)} w_{ij}(t) x^i_j(t)$, where we assume the existence of a constant $\alpha \in \mathbb{R}$, such that $0 < \alpha < 1$, and the weights $w_{ij}(t)$ satisfy the conditions:

1. $w_{ij}(t) = 0$ whenever $j \notin \mathcal{J}_i$;
2. $w_{ij}(t) \geq \alpha, \forall j \in \mathcal{J}_i$; and
3. $\sum_{j \in \mathcal{J}_i \setminus \mathcal{R}_i(t)} w_{ij}(t) = 1$

for all $i \in \mathcal{N}$, and $t \in \mathbb{Z}_{\geq 0}$. It is important to note that the third condition depends on the set of removed nodes, which may change over time. In order to satisfy this condition the values of the weights may need to change over time.

The choice of neighboring sets in the W–MSR algorithm is defined as follows:

1. At each time-step t, each normal node i obtains the values of its neighbors, and forms a sorted list
2. If there are fewer than F nodes with values strictly greater than the value of i, then the normal node removes all those nodes. Otherwise, it removes precisely the largest F values in the sorted list. Likewise, if there are less than F nodes with values strictly less than the normal node i, the normal node removes all such nodes. Otherwise, it removes precisely the smallest F nodes in the sorted list.

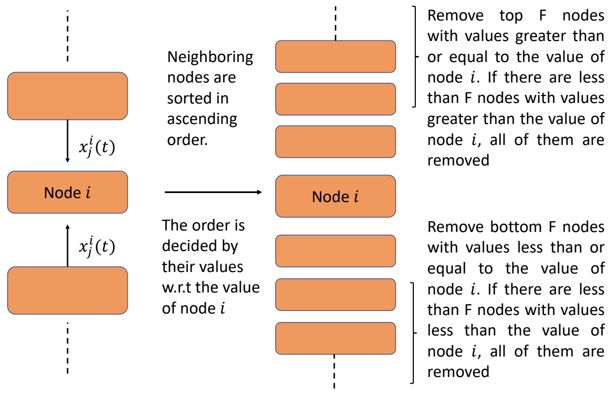
Neighboring nodes are sorted in ascending order.
$x_j^i(t)$
Node i
$x_j^i(t)$
The order is decided by their values w.r.t the value of node i
Node i
Remove top F nodes with values greater than or equal to the value of node i. If there are less than F nodes with values greater than the value of node i, all of them are removed
Remove bottom F nodes with values less than or equal to the value of node i. If there are less than F nodes with values less than the value of node i, all of them are removed

Fig. 2. Schematic of the W-MSR update. At time t, the node i obtains values from its neighbors and forms a sorted list. The algorithm then removes the largest and the smallest F nodes in the sorted list, or if there are less than F nodes with values strictly greater than or less than the value of i, the algorithm removes all those nodes.

An important point to note here is that the above update model holds only for the normal nodes, i.e., $i \in \mathcal{N}$. The update function for adversary nodes, i.e. $i \in \mathcal{A}$, and their influence on the normal nodes depend on the threat model. We will next discuss the formalization of the W–MSR algorithm in Coq.

3 A formal proof of consensus for the W–MSR algorithm

Theorem 1. *[30] Consider a time-invariant network modeled by a digraph $\mathcal{D} = (\mathcal{V}, \mathcal{E})$ where each normal node updates its value according to the W–MSR algorithm with parameter F. Under the F-total malicious model, resilient asymptotic consensus is achieved if and only if the network topology is $(F+1, F+1)$-robust.*

The proof of this theorem requires us to prove both a sufficiency and a necessity condition. The original paper proof relies on a safety condition, which provides an invariant condition that must hold at all times in the state update. We will next discuss the proof of the safety condition (Section 3.1), then sufficiency (Section 3.2) and necessity (Section 3.3) conditions individually.

3.1 Proof of the safety condition in W-MSR

Lemma 1 (Safety condition). *[30] Suppose each node updates its value according to the W-MSR algorithm with parameter F under the F-total malicious model. Then for each node $i \in \mathcal{N}$, $x_i(t+1) \in [m(t), M(t)]$, regardless of the network topology.*

Here, $m(t) = \min_{i\in\mathcal{N}} \{x_i(t)\}$ and $M(t) = \max_{i\in\mathcal{N}} \{x_i(t)\}$. Note that the original paper [30] does not provide a proof of this lemma, and our proof, which we formalize in this paper, is an original contribution. We provide a detailed proof of the lemma by explicitly enumerating the cases from the definition of the W-MSR algorithm. On the other hand, the original paper [30] merely states an outline, making a careful check of the proof difficult.

Proof. We prove Lemma 1 by showing inductively, that at each time t, and for every normal node i, there exists a node $j_1 \in \mathcal{J}_i \cap \mathcal{N}$ such that $\forall k \in \mathcal{J}_i \setminus \mathcal{R}_i(t),\ x_{j_1}(t) \leq x_k(t)$, thus:

$$x_i(t+1) = \sum_{j\in\mathcal{J}_i\setminus\mathcal{R}_i(t)} w_{ij}(t)x_j^i(t) \geq \sum_{j\in\mathcal{J}_i\setminus\mathcal{R}_i(t)} w_{ij}(t)x_{j_1}^i(t) = x_{j_1}^i(t) \geq m(t) \quad (1)$$

Symmetrically there exists a $j_2 \in \mathcal{J}_i \cap \mathcal{N}$ such that $\forall k \in \mathcal{J}_i \setminus \mathcal{R}_i(t), x_{j_2}(t) \geq x_k(t)$. Thus, the symmetric inequality $x_i(t+1) \leq M(t)$, holds for the same reason. Since the proof of the existence of j_1 and j_2 are nearly identical, we only show the proof of the former in Appendix A of the extended version [45].

Formalization in Coq: We formalize Lemma 1 in Coq as:

```
Lemma lem_1: ∀ (i:D) (t:nat) (mal:nat → D → R) (init:D → R)
(A:D → bool) (w:nat → D ∗ D → R),
F_total_malicious mal init A w →
wts_well_behaved A mal init w →
i ∈ Normal A → ((x mal init A w (t+1) i ≤ M mal init A w t)
 ∧ (m mal init A w t ≤ x mal init A w (t+1) i)).
```

The definition of `F_total_malicious` states that the model is F-total malicious if the set of adversary nodes are F-totally bounded (i.e., there are at most F adversary nodes in the network) and all the adversary nodes are malicious. Here `A: D → bool` is a tagging function. If `A i == true`, then i is classified as an *Adversary* node else it is classified as a *Normal* node. `mal : nat → D → R` is an arbitrary update function for a malicious node. Since we do not know beforehand, how this function would look like, we assume it as a parameter. The function `init : D → R` is an initial value associated with a node. We define a `malicious` node in Coq as that node in the graph for which the normal update model does not hold, i.e., there exists a time t such that $x_i(t+1) \neq \sum_{j\in\mathcal{J}_i\setminus\mathcal{R}_i(t)} w_{ij}(t)x_j^i(t)$.

```
(** Condition for a node to have malicious behavior at a given time **)
Definition malicious_at_i_t (mal:nat → D → R) (init:D → R) (A:D → bool)
(w:nat → D * D → R) (i:D) (t:nat): bool :=
(x mal init A w (t+1) i) != $\sum_{j \in \mathcal{J}_i \setminus \mathcal{R}_i(t)}$ ((x mal init A w t j) * (w t (i,j)))

(** Define maliciousness **)
Definition malicious (mal:nat → D → R) (init:D → R) (A:D → bool)
(w:nat → D * D → R) (i:D) := ∃ t:nat, malicious_at_i_t mal init A w i t.
```

The second hypothesis `wts_well_behaved` states that we respect those three conditions on weights that we discussed in Section 2.4. The assignment of weights depend on whether a node $j \in \mathcal{J}_i \backslash \mathcal{R}_i(t)$ or not. Here, $\mathcal{J}_i$ denotes the inclusive set of neighbors of the node i. $\mathcal{R}_i(t)$ denotes the removed set of nodes according to the W–MSR algorithm, and we define $\mathcal{R}_i(t)$ in Coq as follows

```
Definition remove_extremes (i:D) (l:seq D) (x:D → R) : (seq D) :=
  filter (fun (j:D) ⇒
  (((Rge_dec (x j) (x i)) || (F ≤ (index j l))) && ( Rle_dec (x j) (x i)
    || (index j l ≤ ((size l) − F − 1))))) l.
```

Note that we use the filter function from the `MathComp` sequence library. This is crucial as it gives us lemmas that allow us to assert that any node in $\mathcal{J}_i \setminus \mathcal{R}_i(t)$ satisfies the conditions of the filter. Additionally, the filter function requires that its first argument has a `pred` type, `D → bool` in our case. Therefore, we need our inequality operations to be decidable. Hence, we used the decidable versions of the inequality operations, such as `Rle_dec`, provided by Coq's reals library instead of it's built-in $\leq$ operation. We then define the set $\mathcal{J}_i \setminus \mathcal{R}_i(t)$ in Coq as

```
Definition incl_neigh_minus_extremes
(i:D) (x:D → R) : (seq D) := remove_extremes i (inclusive_neighbor_list i x) x.
```

Since $\mathcal{J}_i \backslash \mathcal{R}_i(t)$ is defined based on the value of node i, $x_i(t)$, which indeed depends on `A`, `mal`, `init`. Hence, `wts_well_behaved` depends on `A`, `mal`, `init`.

The trickiest parts of the proof of Lemma 1 rely on the fact that we desire $\mathcal{J}_i \setminus \mathcal{R}_i(t)$ when treated as a list to be sorted. In order to fulfill this condition we use the formalization for sorting found in the `MathComp` library. To do this we first define a relation on D as:

```
Definition sorted_Dseq_rel (x: D → R) (i j : D) :=
if Rle_dec (x i) (x j) then
   if (x i = = x j) then (index i (enum D) ≤ index j (enum D)) else true
else false.
```

This definition ensures that if $x_i(t) < x_j(t)$, then i is ordered as less than j with respect to this relationship. In the case of nodes with equivalent values we use an arbitrary mechanism to break ties. Doing so ensures that this relation is total, and satisfies transitivity, anti-symmetry, and reflexivity. This relation lets us use the sorting lemmas in `MathComp`'s path library [13], and it ensures the weaker condition that we occasionally use in the proof:

```
Definition sorted_Dseq (x:D → R) (l:seq D) :=
∀ (a b:D), a ∈ l → b ∈ l → (index a l < index b l) → (x a ≤ x b).
```

The biggest difficulty with formalizing this proof arises when dealing with the case that $|R_i^{<}(t)| < F$, where $R_i^{<}(t) := \{j \in \mathcal{J}_i : x_j(t) < x_i(t) \text{ and } idx_{\mathcal{J}_i}(x_j(t)) < F\}$, and define $idx_l(x_k(t))$, to be the index of the value $x_k(t)$ in a given list l of values, or the size of l if $x_k(t)$ is not present.. In particular, showing that $idx_{\mathcal{J}_i \setminus \mathcal{R}_i(t)}(j) = 0 \implies n_j(\mathcal{J}_i) = |R_i^{<}(t)|$. This requires proving an extra lemma on the $\mathcal{J}_i$ list:

```
Lemma partition_incl: ∀ (i:D) (t:nat) (mal:nat → D → R)
(init:D → R) (A:D → bool) (w:nat → D ∗ D → R),
inclusive_neighbor_list i (x mal init A w t) =
(sort ((sorted_Dseq_rel (x mal init A w t)) )
  (enum (R_i_less_than mal init A w i t))) + +
(incl_neigh_minus_extremes i (x mal init A w t)) + +
(sort ((sorted_Dseq_rel (x mal init A w t)) )
  (enum (R_i_greater_than mal init A w i t))).
```

With this lemma, we can reason that the zero-th index of $\mathcal{J}_i \setminus \mathcal{R}_i(t)$, is the $|R_i^{<}(t)|$-th index of $\mathcal{J}_i$. Using this lemma, we can prove the existence of j_1 in the proof of `lem_1`. Symmetrically, we can show the existence of j_2 such that $\forall k \in \mathcal{J}_i \setminus \mathcal{R}_i(t),\ x_{j_2}(t) \geq x_k(t)$. Tying it all together, we complete the proof of the lemma `lem_1` in Coq.

3.2 Proof of Sufficiency

Lemma 2. *[30] Consider a time-invariant network modeled by a digraph $\mathcal{D} = (\mathcal{V}, \mathcal{E})$ where each normal node updates its value according to the W–MSR algorithm with parameter F. Under the F-total malicious model, if a network is (F+1, F+1) robust, resilient asymptotic consensus is achieved.*

This is an important lemma because we would like to design a network such that the normal nodes in the network reach an asymptotic consensus in the presence of malicious nodes in the network. Next we will discuss an informal proof of the Lemma 2 followed by its formalization in the Coq proof assistant.

Proof. The proof of Lemma 2 is done by contradiction. We start by assuming that the limits A_M and A_m of the functions $M(t)$ and $m(t)$ respectively are different, i.e., $A_M \neq A_m$. The limits A_M and A_m of the functions $M(t)$ and $m(t)$, respectively, exist because $M(t)$ and $m(t)$ are both continuous and monotonously decreasing functions of t. Therefore, by definition of limits for $M(t)$ and $m(t)$, we know that $\forall\ t,\ A_M \leq M(t)\ \wedge\ m(t) \leq A_m$, as illustrated in Figure 3. We will show that by carefully constructing the sets S_1 and S_2 in the definition of (r, s)-robustness, and unrolling the definition of (r, s)-robustness at every time-step inductively, we eventually arrive at the desired contradiction: $\exists\ t,\ M(t) < A_M\ \vee\ A_m < m(t)$. We discuss the details of the proof in Appendix B of the extended version [45].

Formalization in Coq: We introduce the following axiom in Coq to support reasoning by contradiction.

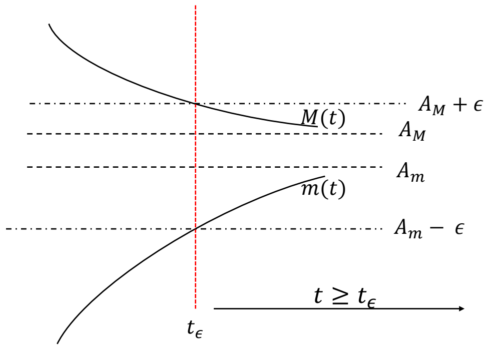

Fig. 3. Illustration of the tube of convergence bounded above by $A_M + \epsilon$ and bounded below by $A_m - \epsilon$. We observe the behavior of functions $M(t)$ and $m(t)$ inside this tube of convergence $\forall t \geq t_\epsilon$. We prove that $M(t)$ and $m(t)$ are monotonous $\forall t \geq t_\epsilon$, and they approach the limits A_M and A_m, respectively. We start by assuming that $A_M \neq A_m$, but later prove that $A_M = A_m$ by contradiction, thereby proving asymptotic consensus.

```
Axiom proposition_degeneracy : ∀ A : Prop, A = True ∨ A = False.
```

This is a propositional completeness lemma that allows us to reason classically and is consistent with the formalization of classical facts in Coq's standard library. We need this lemma because we prove the sufficiency condition using contradiction. We are choosing to use classical reasoning because the original paper [30] does not provide a constructive proof. The reasoning used in the paper is classical. This requires us to state the following lemma in Coq

```
Lemma P_not_not_P: ∀ (P:Prop), P ↔ ¬(¬ P).
```

The proof of P_not_not_P uses the axiom proposition_degeneracy.
We state the sufficiency condition (Lemma 2) for the network to achieve resilient asymptotic consensus as the following in Coq.

```
Lemma strong_sufficiency:
∀ (A:D → bool) (mal:nat → D → R) (init:D → R) (w: nat → D * D → R),
nonempty_nontrivial_graph →
(0 < F+1 ≤ |D|)%N →
wts_well_behaved A mal init w →
r_s_robustness (F + 1) (F + 1) →
Resilient_asymptotic_consensus A mal init w.
```

The sufficiency condition requires that the graph is non-trivial, i.e., there are at least two nodes in the graph, and the number of faulty nodes F in the graph is bounded by the total number of nodes D. We define r_s_robustness in Coq as

```
Definition r_s_robustness (r s:nat):=
nonempty_nontrivial_graph ∧ ((1 ≤ s ≤ |D|) →
∀ (S1 S2: {set D}),
(S1 ⊂ Vertex ∧ (|S1|>0)) →
(S2 ⊂ Vertex ∧ (|S2|>0)) →
[disjoint S1 & S2] →
(( |Xi_S_r S1 r| = = |S1|) ||((| Xi_S_r S2 r| = = |S2|) ||
    (| Xi_S_r S1 r| + |Xi_S_r S2 r| ≥ s)) )).
```

where `Xi_S_r S1 r` is the set of all nodes in the set `S1` such that all of its nodes have at least `r` neighboring nodes outside `S1`. In Coq, we define `Xi_S_r` as

```
Definition Xi_S_r (S: {set D}) (r:nat):=
  [set i:D | i ∈ S & ( | (in_neighbor i) − S| ≥ r)].
```

We define `Resilient_asymptotic_consensus` in Coq as

```
Definition Resilient_asymptotic_consensus
(A:D → bool) (mal:nat → D → R) (init:D → R) (w:nat → D ∗ D → R):=
(F_total_malicious mal init A w) → (∃ L:Rbar, ∀ (i:D),
i ∈ (Normal A) → is_lim_seq (fun t: nat ⇒ x mal init A w t i) L) ∧
  (∀ t:nat, (m mal init A w 0 ≤ m mal init A w t) ∧
    (M mal init A w t ≤ M mal init A w 0)).
```

Here, `is_lim_seq` is a predicate in `Coquelicot` that defines limits of sequences. `Rbar` is the extended set of reals, which includes $+\infty$ and $-\infty$. To prove that the network achieves resilient asymptotic consensus under the $(F+1, F+1)$- robustness condition, we need to prove the following two conditions in the definition of `Resilient_asymptotic_consensus`: (i) $\forall t, m(0) \leq m(t) \wedge M(t) \leq M(0)$, and (ii) $\exists L, \forall i, i \in \mathcal{N} \rightarrow \lim\limits_{t\rightarrow\infty} x_i(t) = L$. We state the first subproof as the lemma statement `interval_bound` in Coq. The proof of lemma `interval_bound` is a consequence of Lemma 1. We prove this lemma by an induction on time t and then apply Lemma 1 to complete the proof.

We prove the second subproof by contradiction in Coq. To start the proof of contradiction, we need to assume that the limits A_M and A_m of the maximum and minimum functions $M(t)$ and $m(t)$ are different. We then instantiate the sets S_1 and S_2 in the definition of (r, s)- robustness with $\mathcal{X}_M(t_\epsilon, \epsilon_o)$ and $\mathcal{X}_m(t_\epsilon, \epsilon_o)$ respectively, where $\mathcal{X}_M(t, \epsilon_l) = \{i \in \mathcal{V} : x_i(t) > A_M - \epsilon_l\}$ and $\mathcal{X}_m(t, \epsilon_l) = \{i \in \mathcal{V} : x_i(t) < A_m + \epsilon_l\}$. In Coq, we define the sets $\mathcal{X}_M$ for any epsilon and t as follows

```
Definition X_m_t_e_i (e_i: R) (A_m :R) (t:nat) (mal : nat → D → R) (init : D → R)
(A: D → bool) (w: nat → D ∗ D → R) :=
[set i:D | Rlt_dec (x mal init A w t i) (A_m + e_i)].
```

where `Rlt_dec` is Coq's standard decidability lemma for less than operation.

We need to prove that the sets $\mathcal{X}_M$ and $\mathcal{X}_m$ are disjoint at all times till we reach a point when either $\mathcal{X}_M$ or $\mathcal{X}_m$ are empty. This requires us to prove the following lemma in Coq

```
Lemma X_M_X_m_disjoint_at_j
(mal : nat → D → R) (init: D → R) (A: D → bool) (w: nat → D ∗ D → R):
  ∀ (t_eps l:nat) (a A_M A_m :R) (eps_0 eps :posreal),
(A_M − (eps_j l eps_0 eps a) > A_m + (eps_j l eps_0 eps a)) →
[disjoint (X_M_t_e_i (eps_j l eps_0 eps a) A_M (t_eps+l) mal init A w) &
  (X_m_t_e_i (eps_j l eps_0 eps a)  A_m (t_eps+l) mal init A w )].
```

Since $\mathcal{X}_m(t_\epsilon+l,\epsilon_l)$ is a set of all nodes with values at least, $A_M-\epsilon_l$ and $\mathcal{X}_m(t_\epsilon+l,\epsilon_l)$ is a set of all nodes with values at most $A_m+\epsilon_l$, these two sets are disjoint if $A_M-\epsilon_l > A_m+\epsilon_l$. For $l=0$, we have defined ϵ_o such that $A_M-\epsilon_o > A_m+\epsilon_o$. To prove that $A_M-\epsilon_l > A_m+\epsilon_l, \forall l, 0 < l$, we need to show that $A_M-\epsilon_l > A_M-\epsilon_o$ and $A_m+\epsilon_o > A_m+\epsilon_l$. This would indeed require us to show that $\epsilon_l < \epsilon_o, \forall l, 0 < l$. This holds since we had defined ϵ_l recursively as $\epsilon_l := \alpha\epsilon_{l-1} + (1-\alpha)\epsilon$.

A crucial aspect of the sufficiency proof is proving that the $(F+1,F+1)$-robustness implies that there exists a node in the union of the set $\mathcal{X}_M \cap \mathcal{N}$ and $\mathcal{X}_m \cap \mathcal{N}$ such that it has at least $F+1$ nodes outside the set. This was particularly challenging because in the original paper [30], the authors do not use all three conditions in the definition of $(F+1,F+1)$ robustness condition to informally prove the implication. They use only the third condition $(F+1 \leq |\mathcal{X}_{\mathcal{X}_M}^{F+1}| + |\mathcal{X}_{\mathcal{X}_m}^{F+1}|)$ to state the implication, while leaving it up on the readers to connect the missing dots with the first two conditions. For the implication to hold, all three conditions in the definition of $(F+1,F+1)$- robustness should imply the existence of such a node since there is an *or* in the definition of $(F+1,F+1)$-robustness connecting the three conditions. To prove the implication from the first two conditions, we need to first prove the existence of a normal node in the sets $\mathcal{X}_M$ and $\mathcal{X}_m$ for all $l \leq N$. This holds since the node i with value $M(t_\epsilon+l)$ will always be above the threshold $A_M-\epsilon_l$ because $M(t) \geq A_M, \forall t$ due to the existence of the limit A_M. Hence, $0 < |\mathcal{X}_M(t_\epsilon+l,\epsilon_l)|, \forall l \leq N$. Since the first condition of $(F+1,F+1)$- robustness states that $|\mathcal{X}_{\mathcal{X}_M(t_\epsilon+l,\epsilon_l)}^{F+1}| = |\mathcal{X}_M(t_\epsilon+l,\epsilon_l)|$, $0 < |\mathcal{X}_{\mathcal{X}_M(t_\epsilon+l,\epsilon_l)}^{F+1}|$. Hence by definition of $\mathcal{X}_{\mathcal{X}_M(t_\epsilon+l,\epsilon_l)}^{F+1}$, there exists a normal node in the set $X_M(t_\epsilon+l,\epsilon_l)$ such that it has at least $F+1$ nodes outside $X_M(t_\epsilon+l,\epsilon_l)$. We prove this formally in Coq using the following lemma statement

```
Lemma X_m_normal_exists_at_j (t_eps l N: nat) (a A_m: R)(eps_0 eps:posreal)
(mal : nat → D → R) (init : D → R) (A: D → bool) (w: nat → D * D → R):
F_total_malicious mal init A w →
wts_well_behaved A mal init w →
(0 < F + 1 ≤ |D|) →
is_lim_seq [eta m mal init A w] A_m →
(0 < N) → (l ≤ N) → (0 < a < 1) → (eps < a^N / (1 − a^N) * eps_0) →
∃ i:D, i ∈ (X_m_t_e_i (eps_j l eps_0 eps a) A_m (t_eps + l) mal init A w) ∧
        i ∈ Normal A.
```

By symmetry, we prove that $0 < |\mathcal{X}_{\mathcal{X}_m(t_\epsilon+l,\epsilon_l)}^{F+1}|$. The other part that was not explicit from the paper proof in the original paper [30] was that the largest value that the node i uses at time step $t_\epsilon+l$ is $M(t_\epsilon+l)$, which is provided without proof. This was a challenge during our formalization. To formally prove this we had to split the neighbor set of i into two parts depending on their relative position with respect to i. While it is easy to bound the values of the nodes positioned in the left side of i with $M(t_\epsilon + l)$ since the neighboring list is assumed to be sorted at the time of update and we have established this upper bound for any normal node from lemma 1, bounding the values for the nodes positioned in the right of the normal node i was not trivial. We proved this using a case analysis on the cardinality of the set $R_i^{>}(t)$. In Coq, we formally prove this using the lemma statement `x_right_ineq_1` in Coq. We do not expand on this lemma here for brevity.

Another challenge during the formalization was using the bound of the neighboring node of i, $A_M - \epsilon_l$ in the update of the value of i at the next time step. We know that the neighbors outside the set $\mathcal{J}_i(t_\epsilon + l)\backslash\mathcal{X}_M(t_\epsilon + l, \epsilon_l)$ have value at most $A_M - \epsilon_l$. But to use these nodes in the update function, we need to show that these neighboring nodes are in the inclusive set of the normal node i minus the extremes, i.e, there exists a node in the intersection of the sets $\mathcal{J}_i(t_\epsilon + l)$ and the set s which contains nodes outside the set $\mathcal{J}_i(t_\epsilon + l)\backslash\mathcal{X}_M(t_\epsilon + l, \epsilon_l)$.We prove the existence of such a node using the following lemma statement in Coq

```
Lemma exists_in_intersection: ∀ (A B: {set D}) (s: seq D) (F:nat),
|s| = (F+1)%N → ( |B| ≤ F)%N →
{subset s <= A − B} → ∃ x:D, x ∈ [set x | x ∈ s] ∩ A.
```

We instantiate the set `A` with $\mathcal{J}_i\backslash\mathcal{R}_i(t)$ and the set `B` with $\mathcal{R}_i^{<}(t)$. We know that by definition of the W–MSR algorithm, $|\mathcal{R}_i^{<}(t)| \leq F$. To use the lemma `exists_in_intersection`, we first had to prove that $s \subset (\mathcal{J}_i\backslash\mathcal{R}_i(t)) \cup \mathcal{R}_i^{<}(t)$. Applying the lemma `exists_in_intersection` then gives us a node k as a witness which lies in the intersection of the set s and $\mathcal{J}_i\backslash\mathcal{R}_i(t)$. We use this node to apply the bound $A_M - \epsilon_l$ in the proof of inequality 1 for $l \leq N$. All other nodes in the neighboring list of the normal node i minus extremes are shown to be bounded by $M(t)$.

To show that the inequality $\exists t, M(t) < A_M \vee A_m < m(t)$ holds, we need to prove that for every l such that $l \leq N$, the cardinality of the set $\mathcal{X}_M$ decreases or the cardinality of the set $\mathcal{X}_m$ decreases or both under the $(F+1, F+1)$-robustness condition. This requires us proving the following lemma in Coq

```
Lemma sj_ind_var (s1 s2: nat → nat) (N:nat): (0< N) → (s1 1 + s2 1 < N) →
(∀ l:nat, (0 < l) → (l ≤ N ) → (0< s1 l) → (0 < s2 l) →
 (s1 l ≤ s1 l.−1) ∧ (s2 l ≤ s2 l.−1) ∧ (( s1 l < s1 l.−1 ) ∨ (s2 l < s2 l.−1))) →
∃ T:nat, (T ≤ N) ∧ (s1 T = 0 ∨ s2 T = 0)
```

We instantiate `s1` and `s2` with $\mathcal{X}_M(t_\epsilon + l, \epsilon_l)$ and $\mathcal{X}_m(t_\epsilon + l, \epsilon_l)$ respectively. We use the lemma `sj_ind_var` to arrive at a contradiction and complete the proof of the sufficiency.

3.3 Proof of necessity

Lemma 3. *[30] Consider a time-invariant network modeled by a digraph $\mathcal{D} = (\mathcal{V}, \mathcal{E})$ where each normal node updates its value according to the W–MSR algorithm with parameter F. Under the F-total malicious model, if resilient asymptotic consensus is achieved then the network is (F+1, F+1)-robust.*

Necessity is a secondary, but still significant lemma. It tells us that there is no weaker condition than $(F+1, F+1)$-robustness such that the normal nodes within the network reach asymptotic consensus. We now discuss an informal proof of Lemma 3. Note that the original paper [30] does not provide a clean proof of this lemma. For example, the original paper provides a sketch of the proof of Lemma 3 by contrapositivity, but does not provide a concrete counterexample to discharge the proof by contrapositive. The paper proof in [30] does not talk about construction of weights or the proof that these weights are not well-behaved under non-(r,s)-robustness. These issues were non-trivial and posed challenges in Coq, as will be explained in this section. We also highlight challenges in the construction of this counterexample and the proof of necessity in Coq, including an issue of mutual recursion in Coq. The issues with missing details in the original paper proof, which we had to develop explicitly, make the proof in this paper an original contribution.

Proof. We proceed by proving the contrapositive of necessity, that is: if the network is not $(F+1, F+1)$ robust then it does not achieve resilient asymptotic consensus. Assuming that the network is not $(F+1, F+1)$-robust we know that there are non-empty sets $S_1, S_2 \subset \mathcal{V}$, such that $S_1 \cap S_2 = \emptyset$, $|\chi_{S_1}^{F+1}| \neq |S_1|$, $|\chi_{S_2}^{F+1}| \neq |S_2|$, and $|\chi_{S_1}^{F+1}| + |\chi_{S_2}^{F+1}| < F+1$. It follows that $|\chi_{S_1}^{F+1}| < F+1$, and $|\chi_{S_2}^{F+1}| < F+1$. Also recall that $\chi_{S_1}^{F+1} \subseteq S_1$, and $\chi_{S_2}^{F+1} \subseteq S_2$. One way of interpreting this condition is that the number of nodes within S_1 and S_2 that can receive a lot of information from outside of their respective sets is less than $F+1$ in total, and less than the number of nodes in each set respectively. We seek to construct a set of adversaries, initial values, malicious functions, and weights such that resilient asymptotic consensus is not achieved. In particular we seek to prove that there exists two normal nodes i, j such that $\lim_{t\to\infty} x_i(t) \neq \lim_{t\to\infty} x_j(t)$. We discuss the details of the proof in the Appendix D of the extended version [45].

Formalization in Coq: We formalize the lemma 3 in Coq as

```
Lemma necessity_proof:
nonempty_nontrivial_graph →
(¬ r_s_robustness (F + 1) (F + 1) →
 ¬ (∀ (A:D → bool) (mal:nat → D → R) (init:D → R) (w:nat → D ∗ D → R),
     wts_well_behaved A mal init w →
     Resilient_asymptotic_consensus A mal init w)).
```

Formalization of `necessity_proof` exposed some inconsistencies in definitions in the original paper [30]. In particular, the paper defines those three conditions on weights, that we discussed in the Section 2.4, only for normal nodes. During our formalization, we found this to be restrictive. Those conditions on weights should hold for any node. The need for applying the conditions in the paper to the weights of adversary nodes, is that in order to ensure that a node $i \in \mathcal{A}$ is malicious, as defined in the paper, there must exist a time t such that the quantity $x_i(t+1) \neq \sum_{j\in\mathcal{J}_i\setminus\mathcal{R}_i(t)} w_{ij}(t)x_j^i(t)$. In other words at some time the value emitted by a given node must not equal the value it would emit if it was normal, but the sum is clearly undefined if the weights of an adversary node are undefined. Therefore, we relax the condition that the set of weights described in the paper only exists for normal nodes. Fortunately this does not create a problem as adversary nodes can update their values according to any function they wish, meaning that they do not have to use the described set of weights, or any weights at all, leaving their values unconstrained by this condition.

Another thing that was not explicit in the original paper [30] was the right placement of quantifiers. Formalizing the proof of necessity helped us identify the right placement of quantifiers and provide an accurate formal specification for the W–MSR algorithm. At the start of our formalization it was not evidently clear to us whether the paper meant to imply that:

```
(∀ (A:D → bool) (mal:nat → D → R) (init:D → R), wts_well_behaved A mal init →
(Resilient_asymptotic_consensus A mal init ↔ r_s_robustness (F + 1) (F + 1))).
```

or:

```
(∀ (A:D → bool) (mal:nat → D → R) (init:D → R),
    wts_well_behaved A mal init →
 Resilient_asymptotic_consensus A mal init) ↔ r_s_robustness (F + 1) (F + 1).
```

In the first formula, the quantified values `A`, `mal`, `init` are not bound to the definition of resilient asymptotic consensus. Therefore, in the necessity proof, we cannot construct a counterexample by appropriate instantiation of `A`, `mal` and `init`, to discharge the proof by contradiction. In the second formula, the quantified values are bound to the definition of resilient asymptotic consensus, which allows us to construct the counterexample by propagating the negation through the quantified values. Essentially, the difference is between the formulae $(\forall X, P(X) \to Q(X))$ and $((\forall X.\ P(X)) \to (\forall X.\ Q(X)))$, where X represents the tuple $(\texttt{A}, \texttt{mal}, \texttt{init})$, and the first statement is stronger. Therefore, the former, stronger condition is not necessarily true in the necessity direction, while the weaker later condition is.

Another difficulty we encountered was defining the weights in such a way that $w_{ij}(t) = \frac{1}{|\mathcal{J}_i \setminus \mathcal{R}_i|}$. This is a result of Coq's sensitivity to ill-defined recursion. The issue arises because defining w_{ij} at time t requires knowing the value of x_i at time t, however, as we had defined x_i, it takes the set of weights it uses as a parameter, even though mathematically there is no issue since $x_i(t)$ only relies on the values of $x_j(t-1)$, and $w_{ij}(t-1)$. In order to solve this issue we defined a function which returns a pair of functions (x_i, w_{ij}). In order to ensure that Coq could guess the parameter being recursed on we also had to add another parameter two_t which is initialized as $2 \cdot t$, and ensure that the pair $(x_i(t), w_{ij}(t))$ is returned when $two_t = 2 \cdot t$, and $(x_{t+1}, w_{ij}(t))$ is returned when $two_t = (2 \cdot t) + 1$.

3.4 Formal proof of the main theorem

We state the main theorem statement 1 in Coq as:

```
Theorem F_total_consensus:
nonempty_nontrivial_graph →
```

```
(0 < F+1 ≤ |D|)%N →
(∀ (A:D → bool) (mal:nat → D → R) (init:D → R) (w:nat → D * D → R),
wts_well_behaved A mal init w →
Resilient_asymptotic_consensus A mal init w) ↔ r_s_robustness (F + 1) (F + 1).
```

We close the proof of `F_total_consensus` by splitting the theorem into sufficiency and necessity sub-proofs and applying the lemmas `sufficiency_proof` and `necessity_proof`. The only detail worth noting is that `necessity_proof` relies on the decidable of `r_s_robustness`, which we need the axiom of the excluded middle to conclude.

4 Related Work

Recently there has been a growing interest in the formalization of distributed systems and control theory, using both automated and interactive verification approaches.

Some notable works in the area of automated verification use model checking, temporal logic, and reachability techniques. For instance, Cimatti et al. [11] have used model checking techniques to formally verify the implementation of a part of safety logic for railway interlocking system. Schrer et al. [43] extended the JavaPathFinder [24] model checker to support modeling of a real-time scheduler and physical system that are defined by differential equations. They verify the safety and liveness properties of a control system, and also verify the programming errors. Besides model checking, temporal logic based techniques have been applied to control synthesis [40], robust model predictive control [14] and automatic verification of sequential control systems [35]. Other approaches for verifying safety use reachability methods like flow pipe approximations [10], zonotope approximation algorithms [19, 28, 2], and ellipsoidal calculus [4].

There has also been significant work in the formalization of control theory using interactive theorem provers [39, 1, 38]. In the area of formalization of stability analysis for control theory, Cyril Cohen and Damien Rouhling formalized the LaSalle's principle in Coq [12]. Stability is important for the control of dynamical systems since it guarantees that trajectories of dynamical systems like cars and airplanes, are bounded. Chan et al. [7] formalize safety properties like Lyapunov stability and exponential stability of cyber-physical systems, in Coq. In [39], Damien Rouhling formalized the soundness of a control function [32] for an inverted pendulum. Some works have also emerged in the area of signal processing for controls. Gallois-Wang et al. [17] formalized some error analysis theorems about digital filters in Coq. Araiza-Illan et al. [3] formally verified high level properties of control systems such stability, feedback gain, or robustness using the Why3 tool [15]. Rashid et al. [38] formalized the transform methods in HOL-Light [22]. Transform methods are used in signal processing and controls to switch between the time domain and the frequency domains for design and analysis of control systems. A few works have emerged in the area of formalization of the feedback control theory to guarantee robustness of control systems. Jasim and Veres et al [26] proved one of the most fundamental and general result of nonlinear feedback system - the *Small-gain theorem (SGT)*, formally using Isabelle/HOL [37]. Hasan et al [23] formalized the theoretical foundations of feedback controls in HOL Light. Another notable work in the formalization of control systems is the formalization of safety properties of robot manipulators by Affeldt et al. [1].

Most of the above works deal with the problem of formalizing the theoretic foundations of control theory – stability analysis, transform methods, filtering algorithms for signal processing, feedback control design. But, to our knowledge, none of these works tackles the problem of consensus in a formal setting. Given that consensus is a quantity of interest in distributed control applications, our work on the formalization of the W–MSR algorithm, is a first step towards formally verified distributed control systems.

5 Conclusion

In this work, we formalize a consensus algorithm [30] for distributed controls in Coq. We formally prove the necessary and sufficient conditions for a set of normal nodes in the network to achieve asymptotic consensus in the presence of a fix bound of malicious nodes in the network. During the process of formalization we discover several areas where the proof in the original paper is imprecise, especially when defining the lemma statements of sufficiency and necessity. In particular, the order of quantifiers on some variables was unclear, and we had to spend time clarifying their order. We also prove a stronger version of the sufficiency condition than the original theorem requires. This is done to ensure that the conditions in both directions of the double implication holds. The definitions and lemmas we formalize in this paper can be used for verifying consensus for other threat models described in the original paper [30]. Overall our work is a first of its kind to provide formal specifications of a consensus algorithm in distributed controls. The total length of Coq proofs is about 11 thousand lines of code. It took us 6 person months for the entire formalization.

A possible future direction of work is to verify the implementation of the algorithm. The proof of this algorithm in the original paper [30], and our formalization assume that all computations are in the real field. However, an actual implementation would need to use finite precision arithmetic. It would therefore be interesting to study the effect of finite precision on the robustness of this algorithm. It would also be interesting to formalize the algorithm for time-variant networks in which the edge relation between the nodes can change with time. Possible use cases for such network model are drone swarms for military and rescue operations, in which each drone in the network could be expected to dynamically change the flow of information from its neighbors.

Acknowledgments: This research was funded in part by NSF grant CCF-2219997.

References

1. Affeldt, R., Cohen, C.: Formal foundations of 3d geometry to model robot manipulators. In: Proceedings of the 6th ACM SIGPLAN Conference on Certified Programs and Proofs. pp. 30–42 (2017)
2. Althoff, M., Krogh, B.H.: Zonotope bundles for the efficient computation of reachable sets. In: 2011 50th IEEE conference on decision and control and European control conference. pp. 6814–6821. IEEE (2011)
3. Araiza-Illan, D., Eder, K., Richards, A.: Formal verification of control systems' properties with theorem proving. In: 2014 UKACC International Conference on Control (CONTROL). pp. 244–249. IEEE (2014)
4. Botchkarev, O., Tripakis, S.: Verification of hybrid systems with linear differential inclusions using ellipsoidal approximations. In: International Workshop on Hybrid Systems: Computation and Control. pp. 73–88. Springer (2000)
5. Carr, H., Jenkins, C., Moir, M., Miraldo, V.C., Silva, L.: Towards formal verification of hotstuff-based byzantine fault tolerant consensus in agda. In: NASA Formal Methods Symposium. pp. 616–635. Springer (2022)
6. Castro, M., Liskov, B., et al.: Practical byzantine fault tolerance. In: OSDI. vol. 99, pp. 173–186 (1999)
7. Chan, M., Ricketts, D., Lerner, S., Malecha, G.: Formal verification of stability properties of cyber-physical systems. Proc. CoqPL (2016)
8. Charron-Bost, B., Merz, S.: Formal verification of a consensus algorithm in the heard-of model. Int. J. Softw. Informatics **3**(2-3), 273–303 (2009)
9. Charron-Bost, B., Merz, S.: Formal Verification of a Consensus Algorithm in the Heard-Of Model. International Journal of Software and Informatics (IJSI) **3**(2-3), 273–303 (2009), https://inria.hal.science/inria-00426388
10. Chutinan, A., Krogh, B.H.: Verification of polyhedral-invariant hybrid automata using polygonal flow pipe approximations. In: International workshop on hybrid systems: computation and control. pp. 76–90. Springer (1999)
11. Cimatti, A., Giunchiglia, F., Mongardi, G., Romano, D., Torielli, F., Traverso, P.: Formal verification of a railway interlocking system using model checking. Formal aspects of computing **10**(4), 361–380 (1998)
12. Cohen, C., Rouhling, D.: A formal proof in coq of lasalle's invariance principle. In: International Conference on Interactive Theorem Proving. pp. 148–163. Springer (2017)
13. Doczkal, C., Pous, D.: Graph theory in coq: Minors, treewidth, and isomorphisms. Journal of Automated Reasoning **64**(5), 795–825 (2020)
14. Farahani, S.S., Raman, V., Murray, R.M.: Robust model predictive control for signal temporal logic synthesis. IFAC-PapersOnLine **48**(27), 323–328 (2015)
15. Filliâtre, J.C., Paskevich, A.: Why3—where programs meet provers. In: European symposium on programming. pp. 125–128. Springer (2013)
16. Függer, M., Nowak, T., Schwarz, M.: Tight bounds for asymptotic and approximate consensus. Journal of the ACM (JACM) **68**(6), 1–35 (2021)
17. Gallois-Wong, D., Boldo, S., Hilaire, T.: A coq formalization of digital filters. In: International Conference on Intelligent Computer Mathematics. pp. 87–103. Springer (2018)
18. Gao, S., Zhan, B., Liu, D., Sun, X., Zhi, Y., Jansen, D.N., Zhang, L.: Formal verification of consensus in the taurus distributed database. In: Formal Methods: 24th International Symposium, FM 2021, Virtual Event, November 20–26, 2021, Proceedings 24. pp. 741–751. Springer (2021)

19. Girard, A., Guernic, C.L.: Zonotope/hyperplane intersection for hybrid systems reachability analysis. In: International Workshop on Hybrid Systems: Computation and Control. pp. 215–228. Springer (2008)
20. Goel, A., Sakallah, K.: On symmetry and quantification: A new approach to verify distributed protocols. In: NASA Formal Methods Symposium. pp. 131–150. Springer (2021)
21. Hance, T., Heule, M., Martins, R., Parno, B.: Finding invariants of distributed systems: It's a small (enough) world after all. In: 18th USENIX Symposium on Networked Systems Design and Implementation (NSDI 21). pp. 115–131 (2021)
22. Harrison, J.: Hol light: A tutorial introduction. In: International Conference on Formal Methods in Computer-Aided Design. pp. 265–269. Springer (1996)
23. Hasan, O., Ahmad, M.: Formal analysis of steady state errors in feedback control systems using hol-light. In: 2013 Design, Automation & Test in Europe Conference & Exhibition (DATE). pp. 1423–1426. IEEE (2013)
24. Havelund, K., Pressburger, T.: Model checking java programs using java pathfinder. International Journal on Software Tools for Technology Transfer **2**(4), 366–381 (2000)
25. Hawblitzel, C., Howell, J., Kapritsos, M., Lorch, J.R., Parno, B., Roberts, M.L., Setty, S., Zill, B.: Ironfleet: proving practical distributed systems correct. In: Proceedings of the 25th Symposium on Operating Systems Principles. pp. 1–17 (2015)
26. Jasim, O.A., Veres, S.M.: Towards formal proofs of feedback control theory. In: 2017 21st International Conference on System Theory, Control and Computing (ICSTCC). pp. 43–48. IEEE (2017)
27. Kieckhafer, R.M., Azadmanesh, M.H.: Reaching approximate agreement with mixed-mode faults. IEEE Transactions on Parallel and Distributed Systems **5**(1), 53–63 (1994)
28. Kochdumper, N., Althoff, M.: Sparse polynomial zonotopes: A novel set representation for reachability analysis. IEEE Transactions on Automatic Control **66**(9), 4043–4058 (2020)
29. Lamport, L., et al.: Paxos made simple. ACM Sigact News **32**(4), 18–25 (2001)
30. LeBlanc, H.J., Zhang, H., Koutsoukos, X., Sundaram, S.: Resilient asymptotic consensus in robust networks. IEEE Journal on Selected Areas in Communications **31**(4), 766–781 (2013)
31. Losa, G., Dodds, M.: On the formal verification of the stellar consensus protocol. In: 2nd Workshop on Formal Methods for Blockchains (FMBC 2020). Schloss Dagstuhl-Leibniz-Zentrum für Informatik (2020)
32. Lozano, R., Fantoni, I., Block, D.J.: Stabilization of the inverted pendulum around its homoclinic orbit. Systems & control letters **40**(3), 197–204 (2000)
33. Ma, H., Goel, A., Jeannin, J.B., Kapritsos, M., Kasikci, B., Sakallah, K.A.: I4: incremental inference of inductive invariants for verification of distributed protocols. In: Proceedings of the 27th ACM Symposium on Operating Systems Principles. pp. 370–384 (2019)
34. Mesbahi, M., Egerstedt, M.: Graph theoretic methods in multiagent networks. Princeton University Press (2010)
35. Moon, I., Powers, G.J., Burch, J.R., Clarke, E.M.: Automatic verification of sequential control systems using temporal logic. AIChE Journal **38**(1), 67–75 (1992)
36. Ongaro, D., Ousterhout, J.: In search of an understandable consensus algorithm. In: 2014 {USENIX} Annual Technical Conference ({USENIX}{ATC} 14). pp. 305–319 (2014)
37. Paulson, L.C.: Isabelle: A generic theorem prover. Springer (1994)

38. Rashid, A., Hasan, O.: Formalization of transform methods using hol light. In: International Conference on Intelligent Computer Mathematics. pp. 319–332. Springer (2017)
39. Rouhling, D.: A formal proof in coq of a control function for the inverted pendulum. In: Proceedings of the 7th ACM SIGPLAN International Conference on Certified Programs and Proofs. pp. 28–41 (2018)
40. Sadraddini, S., Belta, C.: Robust temporal logic model predictive control. In: 2015 53rd Annual Allerton Conference on Communication, Control, and Computing (Allerton). pp. 772–779. IEEE (2015)
41. Saldana, D., Prorok, A., Sundaram, S., Campos, M.F., Kumar, V.: Resilient consensus for time-varying networks of dynamic agents. In: 2017 American control conference (ACC). pp. 252–258. IEEE (2017)
42. Saulnier, K., Saldana, D., Prorok, A., Pappas, G.J., Kumar, V.: Resilient flocking for mobile robot teams. IEEE Robotics and Automation letters **2**(2), 1039–1046 (2017)
43. Scherer, S., Lerda, F., Clarke, E.M.: Model checking of robotic control systems (2005)
44. Schneider, F.B.: Implementing fault-tolerant services using the state machine approach: A tutorial. ACM Comput. Surv. **22**(4), 299–319 (dec 1990). https://doi.org/10.1145/98163.98167, https://doi.org/10.1145/98163.98167
45. Tekriwal, M., Tachna-Fram, A., Jeannin, J.B., Kapritsos, M., Panagou, D.: Formally verified asymptotic consensus in robust networks (extended version). arXiv preprint arXiv:2202.13833 (2022)
46. Usevitch, J., Garg, K., Panagou, D.: Finite-time resilient formation control with bounded inputs. In: 2018 IEEE Conference on Decision and Control (CDC). pp. 2567–2574. IEEE (2018). https://doi.org/10.1109/CDC.2018.8619697
47. Van Renesse, R., Altinbuken, D.: Paxos made moderately complex. ACM Computing Surveys (CSUR) **47**(3), 1–36 (2015)
48. Wilcox, J.R., Woos, D., Panchekha, P., Tatlock, Z., Wang, X., Ernst, M.D., Anderson, T.: Verdi: a framework for implementing and formally verifying distributed systems. In: Proceedings of the 36th ACM SIGPLAN Conference on Programming Language Design and Implementation. pp. 357–368 (2015)
49. Yao, J., Tao, R., Gu, R., Nieh, J., Jana, S., Ryan, G.: DistAI: Data-driven automated invariant learning for distributed protocols. In: 15th USENIX Symposium on Operating Systems Design and Implementation (OSDI 21). pp. 405–421 (2021)
50. Zhang, H., Sundaram, S.: Robustness of information diffusion algorithms to locally bounded adversaries. In: 2012 American Control Conference (ACC). pp. 5855–5861. IEEE (2012)