Formally verified asymptotic consensus in robust networks

Anonymous Author(s)*

Abstract

Distributed architectures are used to improve performance and reliability of various systems. An important capability of a distributed architecture is the ability to reach consensus among all its nodes. To achieve this, several consensus algorithms have been proposed for various scenarii, and many of these algorithms come with proofs of correctness that are not mechanically checked. Unfortunately, those proofs are known to be intricate and prone to errors. In this paper, we formalize and mechanically check a consensus algorithm widely used in the distributed controls community: the *Weighted-Mean Subsequence Reduced (W-MSR) algorithm* proposed by Le Blanc et al. This algorithm provides a way to achieve *asymptotic* consensus in a distributed controls scenario in the presence of adversarial agents (attackers), that may not update their states based on the nominal consensus protocol, and may share inaccurate information with their neighbors. Using the Coq proof assistant, we formalize the necessary and sufficient conditions required to achieve resilient asymptotic consensus under the assumed attacker model. We leverage the existing Coq formalizations of graph theory, finite sets and sequences of the `mathcomp` library for our development. To our knowledge, this is the first mechanical proof of an asymptotic consensus algorithm. During the formalization, we clarify several imprecisions in the paper proof, including an imprecision on quantifiers in the main theorem.

Keywords: W-MSR algorithm, Adversaries, Resilient asymptotic consensus, Directed graphs, 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 [30]. 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 [40]. Consensus has been studied extensively in different communities. In the distributed computer systems communities, some prominent algorithms achieving consensus are Paxos [26], MultiPaxos [42], Raft [32], and Practical Byzantine Fault Tolerance (PBFT) [5]. In the distributed robotics and controls community, the Mean Subsequence Reduced (MSR) algorithm [24] and its recent extension the Weighted Mean Subsequence Reduced (W-MSR) algorithm [27, 45] achieve asymptotic consensus in partially connected groups of nodes.

While the systems community has long invested in producing mechanically checked proofs of its consensus protocols, the controls community 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 [22, 43] to automatic generation of inductive invariants [16, 18, 29, 44]. In the distributed robotics and controls community, researchers prove their consensus protocols using mathematical analysis based on Lyapunov theory and its extensions, usually using paper proofs, without computer-checked formalizations. Formal methods have been used in other areas of controls, using approaches such as model checking [10], linear temporal logic [36], and reachability techniques [7, 15] to verify safety and liveness properties.

In this paper we provide the first formalization of the widely-used W-MSR algorithm, showing that mechanically-checked proofs are possible for the consensus algorithms

Submitted to CPP'23, January 16–17, 2023, Boston, MA

ACM ISBN 978-1-4503-XXXX-X/18/06…$15.00
https://doi.org/XXXXXXX.XXXXXXX

studied in the distributed controls community, and provide a much greater level of assurance than a paper proof. In particular, our formalization reveals imprecision on quantifiers in the main theorem stated in the original paper [27]. This further solidifies the need for formalization of controls-like consensus protocols.

The MSR and W-MSR algorithms are quite different from consensus algorithms such as MultiPaxos, Raft or PBFT. 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, many of the techniques used in model-checking or for generating invariants are not well-suited to prove asymptotic properties, which depend on formalizations of real analysis and limits.

In this paper we provide a machine-checked proof of asymptotic consensus in the Coq proof assistant, and apply it to formally verify the W-MSR algorithm. This algorithm provides *asymptotic* consensus in the presence of malicious agents. We have chosen to formalize this algorithm since it is a widely-used algorithm for resilient consensus [37, 38, 41]. 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. We focus on a time invariant network [27] modeled by a directed graph, and leverage existing formalizations of graph theory [11], finite sets [17] and sequences in Coq.

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 [27]. In Section 3, we state the main theorem statement and the intermediate lemmas required to complete the proof. We split the main proof into two lemmas, one corresponding to the sufficiency condition and the other corresponding to the necessity condition. We discuss informal proofs of the lemmas and elaborate their formalizations in the Coq proof assistant. We also discuss some specific challenges we encountered during the formalization of the lemmas and the theorem. After reviewing 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 Background

In this paper we consider the problem of formalizing consensus in a network, and adopt the problem formulation from [27]. Our Coq formalization is available as attached supplementary material.

Consider a network that is modeled by a *digraph* (directed graph), $\mathcal{D} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V} = \{1, .., n\}$ 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 is 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 $\mathcal{J}_i = \mathcal{V}_i \cup \{i\}$. We next discuss the update model that we use in our formalization.

2.1 Threat Model

The threat model that we use in our formalization is the *F-total malicious model*, i.e., the set of adversary nodes are F-totally bounded, where F is some given constant, and all adversary nodes are malicious. Here F represents the maximum number of faulty or adversary nodes in the network.

Definition 2.1. [27] A node $i \in \mathcal{A}$ is called **Malicious** if it may send whatever value to its neighbors, but no two neighbors receive different values from i.

Definition 2.2. (F-total set [27]) 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 2.3. [27] A set of adversary nodes is **F–totally bounded** if it is an F–total set.

2.2 Robust network topologies

Ideally, we would want a network to be robust so that it is immune to malicious attacks. In our formalization, we refer to a metric for robustness — (r, s)-robustness — proposed by the authors in the original W-MSR paper [27].

Definition 2.4. ((r, s)-robustness [27]):
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

1. $|\mathcal{X}^r_{\mathcal{S}_1}| = |\mathcal{S}_1|$;
2. $|\mathcal{X}^r_{\mathcal{S}_2}| = |\mathcal{S}_2|$; or
3. $|\mathcal{X}^r_{\mathcal{S}_1}| + |\mathcal{X}^r_{\mathcal{S}_2}| \geq s$.

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 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.

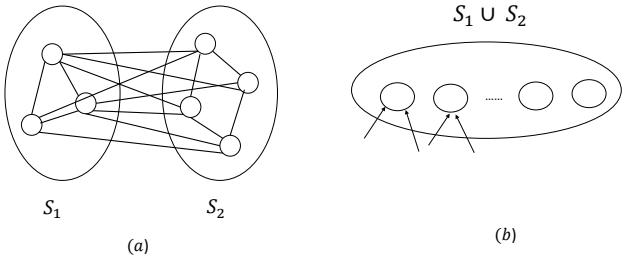

Figure 1. Illustration for $(2, 2)$ robustness. In the illustration (a), every node of the set S_2 has 2 neighboring nodes outside S_2. Similarly every node in the set S_1 has at least 2 neighboring nodes outside S_1. In the illustration (b), there are 2 nodes in the union $S_1 \cup S_2$ that have 2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

2.3 Description of W–MSR

In this paper, we will consider a consensus algorithm, called the W–MSR algorithm [27]. This algorithm provides an update model for the normal nodes in the network. A schematic of the algorithm is illustrated in the 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 $w_{ij}(t)$. 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,

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

where 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 $t \in \mathbb{Z}_{\geq 0}$, assuming that there exists a constant $\alpha \in \mathbb{R}$, such that $0 < \alpha < 1$. The quantity $x_j^i(t)$ is the information that the j^{th} node in the neighboring set of node i sends to the node i. 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 value $x_i(t)$ could represent a measurement like position, velocity, or it could be an optimization variable.

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.

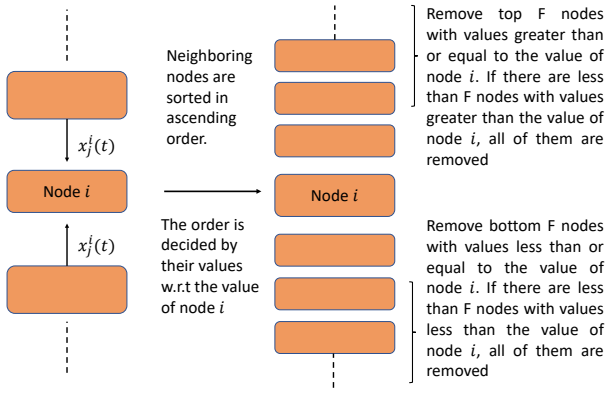

Figure 2. Schematic of the W-MSR update. At time t, the node i obtains values from its neighbors and forms a sorted list. The algorithm then removes the largest and the smallest F nodes in the sorted list, or if there are less than F nodes with values strictly greater than or less than the value of i,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

The theorem that we formalize is stated as follows:

Theorem 3.1. *[27] 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 an intermediate lemma, which provides an invariant condition that must hold at all times in the state update. We will next discuss the proof of the invariant condition, then necessity and sufficient conditions individually.

3.1 Proof of the invariant condition in W-MSR

The invariant property that we use in the proof is given by the following lemma

Lemma 3.2. *[27] 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, $m(t)$ is the minimum value of the normal nodes in the network at time t. Similarly, $M(t)$ is the maximum value of the normal nodes in the network at time t.

Proof. We prove lemma 3.2 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:

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

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.

Proof for the existence of j_1: We define the following sets. Regard $\mathcal{J}_i$ to be the set of neighbors of i, interpreted as a list sorted according to the x-values of it's nodes with ties broken according to a total ordering placed on $\mathcal{V}_i$, 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. If the value $x_k(t)$ is repeated then $idx_l(x_k(t))$ is the index corresponding to where the node k would be relative to the total ordering on $\mathcal{V}_i$. We may use $idx(x_k(t))$ if the list is clear from the context. Let $R_i^<(t) := \{j \in \mathcal{J}_i : x_j(t) < x_i(t) \text{ and } idx_{\mathcal{J}_i}(x_j(t)) < F\}$, and define $R_i^>(t)$ in a similar fashion.

Note that in all cases $|R_i^<(t)| \leq F$, and $j \in R_i^<(t) \implies \forall k \in \mathcal{J}_i \setminus \mathcal{R}_i(t), x_j(t) \leq x_k(t)$. We proceed by case analysis on the size of $R_i^<(t)$.

1. $|R_i^<(t)| = F$, since $|\mathcal{A}| \leq F$, then either $\mathcal{A} = R_i^<(t)$ or there exists a $k \in R_i^<(t)$, such that $k \in \mathcal{N}$. In the first case all nodes in $\mathcal{J}_i \setminus \mathcal{R}_i(t)$ are also normal nodes, so we may take the largest such node as our j_2. In the second case, by the definition of $R_i^<(t)$, $x_k(t) \in \mathcal{N}$, so we may pick k as our j_2.
2. $|R_i^<(t)| < F$. Let j be the node corresponding to the first value in the sorted list $\mathcal{J}_i \setminus \mathcal{R}_i(t)$. Thus, $\forall k \in \mathcal{J}_i \setminus \mathcal{R}_i(t), x_j(t) \leq x_k(t)$. However, we do not know that j is a normal node, but we can prove that $x_j(t) = x_i(t)$. By the above set of inequalities $x_j(t) \leq x_i(t)$. Now we assume WLOG that $j \neq i$. Since we know that $x_j(t) \notin R_i^<(t)$, it follows that $x_i(t) \leq x_j(t)$, or $F \leq idx_{\mathcal{J}_i}(x_j(t))$. However, we know that $R_i^<(t)$ makes up the first $|R_i^<(t)|$ nodes in $\mathcal{J}_i$, so $idx_{\mathcal{J}_i}(x_j(t)) = |R_i^<(t)|$. Since $|R_i^<(t)| < F$, $F \leq idx_{\mathcal{J}_i}(x_j(t))$ is false, we know that $x_i(t) \leq x_j(t)$, and we are done, since $\forall k \in \mathcal{J}_i \setminus \mathcal{R}_i(t), x_i(t) = x_j(t) \leq x_k(t)$, and we know by assumption that $i \in \mathcal{N}$. Thus we may take i as our j_1.

□

Formalization in Coq. We formalize lemma 3.2 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)%Re ∧
        (m mal init A w t ≤ x mal init A w (t+1) i)%Re).
```

where `F_total_malicious` is defined as

```
Definition F_total_malicious (mal:nat → D → R)
(init:D → R) (A:D → bool) (w:nat → D * D → R) :=
F_totally_bounded A ∧
(∀ i:D, i ∈ Adversary A → malicious mal init A w i) ∧
(∀ j:D, j ∈ Normal A → ¬(malicious mal init A w j)).
```

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 before hand, how this function would look like, we assume it as a parameter. `init : D → R` is an initial value associated with a node.

In Coq, we define the sets of *Normal* and *Adversary* nodes as

```
(** Define the vertex set **)
Definition Vertex : {set D}:= setT.

(** Define the set of advversary nodes **)
Definition Adversary (A:D → bool):=
  [set x:D | A x = = true].

(** Defines set of normal nodes **)
```

```
Definition Normal (A:D → bool):= Vertex – (Adversary A).
```

The notation setT defines a full set, i.e., a set of all nodes of type D. D is an instance of a digraph, defined in [11]. 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 $x_i(t+1) \neq \sum_{j\in\mathcal{J}_i\setminus\mathcal{R}_i(t)} w_{ij}(t)x_j^i(t)$.

```
(** Defines 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) !=
    ∑_{j∈J_i\R_i(t)} ((x mal init A w t j) * (w t (i,j)))%Re

(** 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 the section 2. We define wts_well_behaved in Coq as follows

```
Definition wts_well_behaved
(A:D → bool) (mal:nat → D → R) (init:D → R)
(w:nat → D * D → R) :=
∃ a:R, (0<a)%Re ∧ (a <1)%Re ∧
(∀ (t:nat) (i:D),
let incl :=
    (incl_neigh_minus_extremes i (x mal init A w t)) in
(∀ j:D, j ∉ incl → (w t (i,j) = 0)%Re) ∧
(∀ j:D, j ∈ incl → (a ≤ w t (i,j)))%Re ∧
∑_{j∈incl} w t (i,j) = 1%Re.
```

The assignment of weights depend on whether a node j is in the inclusive set of neighbors of a node i minus the removed set, $\mathcal{J}_i\setminus\mathcal{R}_i(t)$, or not, and $\mathcal{J}_i\setminus\mathcal{R}_i(t)$ 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.

In the statement of lemma 3.2, $m(t)$ and $M(t)$ are the minimum and maximum value of the normal nodes at some time t, respectively. We define the minimum function m and the maximum function M in Coq as

```
Definition m (mal:nat → D → R) (init:D → R) (A:D → bool)
(w:nat → D * D → R) (t:nat): R :=
  –(bigmaxr 0 (( map (fun i:D ⇒ –(x mal init A w t i)))
    (enum (Normal A)))).
Definition M (mal:nat → D → R) (init:D → R) (A:D → bool)
(w:nat → D * D → R) (t:nat): R :=
  bigmaxr 0 (map (x mal init A w t) (enum (Normal A))).
```

Here, we use the mathcomp's bigmaxr operator to iteratively look for the maximum value of $x_i(t)$ in a list of normal nodes i. The map function just maps each normal node i to its corresponding value $x_i(t)$ at time t. To define the minimum function m, we take maximum of the negative values, $x_i(t)$.

We formalize $\mathcal{J}_i \setminus \mathcal{R}_i(t)$ using the following two definitions:

```
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))%N) &&
     ( Rle_dec (x j) (x i) ||
    (index j l ≤ ((size l) – F – 1))%N))) l.

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

remove_extremes removes the extreme set of nodes from the inclusive neighbors list of the node i based on the conditions defined by the W–MSR algorithm. 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 ≤ operation. %Re, is used to scope a value to be a real number based on the implementation of Reals in the standard library. Likewise, %N is used to scope a value to be a natural number.

The trickiest parts of the proof of lemma 3.2 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 like so:

```
Definition sorted_Dseq_rel (x: D → R) (i j:D):=
if Rle_dec (x i) (x j) then if (x i = = x j)%Re then
(index i (enum D) ≤ index j (enum D))%N 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, 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)%N → (x a ≤ x b)%Re.
```

The biggest difficulty with formalizing this proof arises when dealing with the case that $|R_i^<(t)| < F$. 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 zeroth 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 that there exists j_1 satisfying the desired properties, which we formalize as:

```
Lemma exists_j1:
∀ (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 →
(∃ (j1:D),
 j1 ∈ (inclusive_neighbor_list i (x mal init A w t)) ∧
 j1 ∈ Normal A ∧
 ∀ (k:D),
 k ∈ (incl_neigh_minus_extremes i (x mal init A w t)) →
 (( x mal init A w t j1) ≤ (x mal init A w t k))%Re).
```

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

The lemma that we prove here is the following:

Lemma 3.3. *[27] 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 the 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 3.3 followed by its formalization in the Coq proof assistant.

Proof. The proof of lemma 3.3 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 $M(t)$ is a continuous and monotonously decreasing, and $m(t)$ is a continuous and monotonously increasing function of t. It then follows that $A_m < A_M$.

High level overview: The idea of the proof is to first construct the sets S_1 and S_2 in the definition of $(r, s)-$ robustness. We then unroll the definition of $(r, s)-$ robustness at every time-step to inductively prove an intermediate lemma which helps us arrive at the desired contradiction. We know that $\forall t, A_M \leq M(t) \ \wedge \ m(t) \leq A_m$ by the definition of limits for $M(t)$ and $m(t)$. Therefore, the contradiction that we eventually arrive to through the proof construction is

$$\exists t, M(t) < A_M \vee A_m < m(t)$$

We will next discuss the proof construction in detail.

Construction of the sets S_1 and S_2 in the definition of $(r, s)-$ robustness: To use the definition of $(r, s)-$ robustness in the hypothesis of the lemma, we need to instantiate the sets S_1 and S_2 in its definition. The construction of these sets are as follows.

Let us construct a set, $\mathcal{X}_M(t, \epsilon_l) = \{i \in \mathcal{V} : x_i(t) > A_M - \epsilon_l\}$ which includes all normal and malicious nodes that have values larger than $A_M - \epsilon_l$. We can similarly construct a set, $\mathcal{X}_m(t, \epsilon_l) = \{i \in \mathcal{V} : x_i(t) < A_m + \epsilon_l\}$ which includes all normal and malicious nodes that have values smaller than $A_m + \epsilon_l$. By the definition of convergence, there exists a time t_ϵ such that $M(t) < A_M + \epsilon$ and $m(t) > A_m - \epsilon$, $\forall t \geq t_\epsilon$. The figure 3 illustrates the behavior of $M(t)$ and $m(t)$ inside the tube of convergence bounded above by $A_M+\epsilon$ and bounded below by $A_m - \epsilon$. At time instance t_ϵ, consider

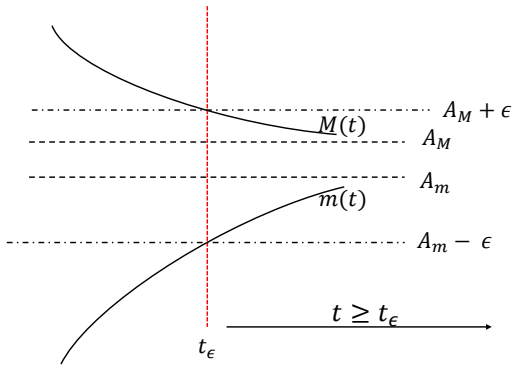

Figure 3. Illustration of the tube of convergence bounded above by $A_M + \epsilon$ and bounded below by $A_m - \epsilon$. We observe the behavior of functions $M(t)$ and $m(t)$ inside this tube of convergence $\forall t \geq t_\epsilon$. We prove that $M(t)$ and $m(t)$ are monotonous $\forall t \geq t_\epsilon$, 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.

the nonempty sets $\mathcal{X}_M(t_\epsilon, \epsilon_o)$ and $\mathcal{X}_m(t_\epsilon, \epsilon_o)$. By density of reals, there exists a constant $\epsilon_o > 0$ such $A_M - \epsilon_o > A_m + \epsilon_o$. Therefore, $\mathcal{X}_M(t_\epsilon, \epsilon_o)$ and $\mathcal{X}_m(t_\epsilon, \epsilon_o)$ are disjoint. Here, α is

a lower bound on the weights $w_{ij}(t)$ which comes from the conditions on weights we discussed in section 2. N is the cardinality of the normal set of nodes $\mathcal{N}$. We obtain the constant ϵ by fixing it such that $\epsilon < \frac{\alpha^N}{1-\alpha^N}\epsilon_o$ which satisfies $\epsilon_o > \epsilon > 0$. At time t_ϵ, we instantiate S_1 and S_2 with $\mathcal{X}_M(t_\epsilon, \epsilon_o)$ and $\mathcal{X}_m(t_\epsilon, \epsilon_o)$, respectively. For all t, $t \geq t_\epsilon$, we instantiate the set S_1 and S_2 with $\mathcal{X}_M(t, \epsilon_l)$ and $\mathcal{X}_m(t, \epsilon_l)$, respectively, as long as there is a normal node in these sets.

Unrolling the definition of $(r,s)-$ robustness for one time step: Since $\mathcal{X}_M(t_\epsilon, \epsilon_o)$ and $\mathcal{X}_m(t_\epsilon, \epsilon_o)$ are nonempty and disjoint, $(F+1, F+1)$-robustness implies that there exists a normal node in the union of $\mathcal{X}_M(t_\epsilon, \epsilon_o)$ and $\mathcal{X}_m(t_\epsilon, \epsilon_o)$ such that it has at least $F+1$ neighbors outside its set. This follows from the definition of (r,s)- robustness. According to the condition (iii), at least $F+1$ nodes must have at least $F+1$ neighbors outside the set. Since the network is allowed to have a maximum of F faulty nodes, there is at least one normal node in the union that has at least $F+1$ neighbors outside the union. By definition, these neighbors have values at most equal to $A_M - \epsilon_o$ or at least $A_m + \epsilon_o$.

Since there exists a normal node in the union of the sets $\mathcal{X}_M(t_\epsilon, \epsilon_o)$ and $\mathcal{X}_m(t_\epsilon, \epsilon_o)$, let us assume for the purpose of illustration that such a node lies in the set $\mathcal{X}_M(t_\epsilon, \epsilon_o)$, i.e., $i \in \mathcal{X}_M(t_\epsilon, \epsilon_o) \cap \mathcal{N}$ with at least $F+1$ neighbors outside of $\mathcal{X}_M(t_\epsilon, \epsilon_o)$. The set of arguments we lay for $i \in \mathcal{X}_M(t_\epsilon, \epsilon_o) \cap \mathcal{N}$ can be similarly constructed for $i \in \mathcal{X}_m(t_\epsilon, \epsilon_o) \cap \mathcal{N}$ by symmetry.

Let us now consider an update of the value of the node i at the next time step, i.e., $x_i(t_\epsilon + 1)$. According to the W–MSR update,

$$x_i(t_\epsilon + 1) = \sum_{j \in \mathcal{J}_i \backslash \mathcal{R}_i(t_\epsilon)} w_{ij}(t_\epsilon) x_j^i(t_\epsilon)$$

The problem is now to bound $x_i(t_\epsilon + 1)$. This bound can be obtained by the following set of inequalities

$$\begin{aligned} x_i(t_\epsilon + 1) &\leq (1-\alpha)M(t_\epsilon) + \alpha(A_M - \epsilon_o) \\ &\leq (1-\alpha)(A_M + \epsilon) + \alpha(A_M - \epsilon) \\ &[since, M(t_\epsilon) < A_M + \epsilon, \forall t \geq t_\epsilon] \\ &\leq A_M - \alpha\epsilon_o + (1-\alpha)\epsilon \end{aligned} \tag{2}$$

To prove this inequality, we need to show that the upper bound on the value of nodes in the set $\mathcal{J}_i \backslash \mathcal{R}_i(t_\epsilon)$ is $M(t_\epsilon + 1)$, and that at least one node in the set $\mathcal{J}_i \backslash \mathcal{R}_i(t_\epsilon)$ has an upper bound of $A_M - \epsilon_o$ on its value. We next present an informal proof of 2.

Proof. Consider the sets $R_i^>(t_\epsilon)$ as the set of all nodes with values strictly greater than $x_i(t_\epsilon)$. Similarly, $R_i^<(t_\epsilon)$ is the set of all nodes with values strictly less than $x_i(t_\epsilon)$. The nodes in $R_i^<(t_\epsilon)$ and $R_i^>(t_\epsilon)$ will be removed when we update the value of node i at next time step, $(t_\epsilon + 1)$. By the W–MSR algorithm, $|R_i^<(t_\epsilon)| \leq F$ and $|R_i^>(t_\epsilon)| \leq F$. The remaining set of nodes in the inclusive neighbors of i form the set $\mathcal{J}_i \backslash \mathcal{R}_i(t_\epsilon)$. The sets $R_i^<(t_\epsilon)$, $R_i^>(t_\epsilon)$ and $\mathcal{J}_i \backslash \mathcal{R}_i(t_\epsilon)$ are mutually disjoint and their union form the set of inclusive neighbors of i. Since the node i takes a sorted list of neighboring nodes for its update according to the W–MSR algorithm, we assume that the inclusive and the inclusive neighbors minus extremes, i.e., $\mathcal{J}_i \backslash \mathcal{R}_i(t_\epsilon)$ are sorted.

We can divide the set $\mathcal{J}_i \backslash \mathcal{R}_i(t_\epsilon)$ into two sets:

- $(\mathcal{J}_i \backslash \mathcal{R}_i(t_\epsilon))_{high}$
- $(\mathcal{J}_i \backslash \mathcal{R}_i(t_\epsilon))_{low}$

depending on their relative position to the node i. The values of the nodes in the set $(\mathcal{J}_i \backslash \mathcal{R}_i(t_\epsilon))_{high}$ at time t_ϵ are bounded above by $M(t_\epsilon)$. This holds because if $|R_i^>(t_\epsilon)| < F$, then all nodes with values strictly greater than the node i are removed and all nodes in the set $(\mathcal{J}_i \backslash \mathcal{R}_i(t_\epsilon))_{high}$ have the same value as the node i. Since the node i is normal, its value is bounded above by $M(t_\epsilon)$ at time step t_ϵ since $M(t_\epsilon) = max_i\{x(t_\epsilon)\}, i \in \mathcal{N}$. Hence, all the nodes in the set $(\mathcal{J}_i \backslash \mathcal{R}_i(t_\epsilon))_{high}$ have value at most $M(t_\epsilon)$. If $|R_i^>(t_\epsilon)| = F$, we consider two cases:

1. All nodes in the removed set are adversary. Since by definition of F-total malicious model, there can be at most F malicious nodes in the network, all nodes in the set $(\mathcal{J}_i \backslash \mathcal{R}_i(t_\epsilon))_{high}$ are normal and are bounded above by $M(t_\epsilon)$ at time t_ϵ.
2. At least one node in the removed set is normal. Therefore, the values of all the nodes in the set $(\mathcal{J}_i \backslash \mathcal{R}_i(t_\epsilon))_{high}$ will be bounded above by the value of the removed normal node which in itself is bounded above by $M(t_\epsilon)$.

Therefore, $\quad x_k(t_\epsilon) \leq M(t_\epsilon), \forall k \in (\mathcal{J}_i \backslash \mathcal{R}_i(t_\epsilon))_{high}$

Since there are at least $F+1$ neighbors outside the set $\mathcal{X}_M(t_\epsilon, \epsilon_o)$, there exists a set s with $F+1$ nodes such that $s \subset \mathcal{J}_i \backslash \mathcal{X}_M(t_\epsilon, \epsilon_o)$ and its values are at most $A_M - \epsilon_o$. Since $|R_i^<(t_\epsilon)| \leq F$, there exists a node in the intersection of sets s and $\mathcal{J}_i$. This node will have a value at most $A_M - \epsilon_o$. We can prove that except for this node, other nodes in the set $\mathcal{J}_i$ is bounded above by $M(t_\epsilon)$. This holds because for the nodes in the set $(\mathcal{J}_i \backslash \mathcal{R}_i(t_\epsilon))_{high}$, the values are bounded above by $M(t_\epsilon)$ as discussed earlier. For the nodes in the set $(\mathcal{J}_i \backslash \mathcal{R}_i(t_\epsilon))_{low}$, the values are also bounded above by $M(t_\epsilon)$ since the set $\mathcal{J}_i$ is sorted and the nodes in the set $(\mathcal{J}_i \backslash \mathcal{R}_i(t_\epsilon))_{low}$ lie to the left of the node i. Therefore,

$$x_k(t_\epsilon) \leq \begin{cases} A_M - \epsilon_o, \ \forall k \in s \cap (\mathcal{J}_i \backslash \mathcal{R}_i(t_\epsilon)) \\ M(t_\epsilon), \ \forall k \in (\mathcal{J}_i \backslash \mathcal{R}_i(t_\epsilon)) \backslash (s \cap (\mathcal{J}_i \backslash \mathcal{R}_i(t_\epsilon))) \end{cases}$$

Hence,

$$\begin{aligned} x_i(t_\epsilon + 1) &\leq (1-\alpha)M(t_\epsilon) + \alpha(A_M - \epsilon_o) \\ &\leq (1-\alpha)(A_M + \epsilon) + \alpha(A_M - \epsilon) \\ &[since, M(t_\epsilon) < A_M + \epsilon, \forall t \geq t_\epsilon] \\ &\leq A_M - \alpha\epsilon_o + (1-\alpha)\epsilon \end{aligned} \tag{3}$$

□

Here, we consider the fact that α is a lower bound on the weights and the sum of all weights is 1. By following a similar line of arguments starting from the set $\mathcal{X}_m(t_\epsilon, \epsilon_o) \cap \mathcal{N}$, we can prove that

$$x_i(t_\epsilon + 1, \epsilon_1) \geq A_m + \alpha\epsilon_o - (1-\alpha)\epsilon$$

Let us define $\epsilon_1 \triangleq \alpha\epsilon_o - (1-\alpha)\epsilon$, which satisfies $0 < \epsilon < \epsilon_1 < \epsilon_0$. Consider the set $\mathcal{X}_M(t_\epsilon, \epsilon_1)$. Since at least one of the normal nodes of $\mathcal{X}_M(t_\epsilon, \epsilon_o)$ decreases at least to $A_M - \epsilon_1$ (or below) or increases to at least $A_m + \epsilon_1$, it must be that $|\mathcal{X}_M(t_\epsilon{+}1, \epsilon_1) \cap \mathcal{N}| < |\mathcal{X}_M(t_\epsilon, \epsilon_o) \cap \mathcal{N}|$ or $|\mathcal{X}_m(t_\epsilon{+}1, \epsilon_1) \cap \mathcal{N}| < |\mathcal{X}_m(t_\epsilon, \epsilon_o) \cap \mathcal{N}|$, or both, i.e., that node is kicked out of the set $\mathcal{X}_M(t_\epsilon + 1, \epsilon_1) \cap \mathcal{N}$ or $\mathcal{X}_m(t_\epsilon + 1, \epsilon_1) \cap \mathcal{N}$, or both.

Proving consensus inductively: So far, we have shown that due to $(r, s)-$ robustness, a normal node is kicked out of the sets $\mathcal{X}_M(t_\epsilon + 1, \epsilon_1) \cap \mathcal{N}$ and/or $\mathcal{X}_m(t_\epsilon + 1, \epsilon_1) \cap \mathcal{N}$.

We can carry this forward inductively as long as there are still normal nodes in $\mathcal{X}_M(t_\epsilon + l, \epsilon_l)$ and $\mathcal{X}_m(t_\epsilon + l, \epsilon_l)$ for time step $t_\epsilon + l$, and $|\mathcal{X}_M(t_\epsilon + l, \epsilon_l) \cap \mathcal{N}| \leq |\mathcal{X}_M(t_\epsilon + (l-1), \epsilon_{l-1}) \cap \mathcal{N}|$, and $|\mathcal{X}_m(t_\epsilon + l, \epsilon_l) \cap \mathcal{N}| \leq |\mathcal{X}_m(t_\epsilon + (l-1), \epsilon_{l-1}) \cap \mathcal{N}|$. For $l \geq 1$, define ϵ_l recursively as $\epsilon_l = \alpha\epsilon_{l-1} - (1-\alpha)\epsilon$. We can prove that $\epsilon_l < \epsilon_{l-1}$. At time step $t_\epsilon + l$, we have that $|\mathcal{X}_M(t_\epsilon + l, \epsilon_l) \cap \mathcal{N}| < |\mathcal{X}_M(t_\epsilon + (l-1), \epsilon_{l-1}) \cap \mathcal{N}|$ or $|\mathcal{X}_m(t_\epsilon + l, \epsilon_l) \cap \mathcal{N}| < |\mathcal{X}_m(t_\epsilon + (l-1), \epsilon_{l-1}) \cap \mathcal{N}|$. Since $|\mathcal{X}_M(t_\epsilon, \epsilon_o) \cap \mathcal{N}| + |\mathcal{X}_m(t_\epsilon, \epsilon_o) \cap \mathcal{N}| \leq N$, there must exist some time-step $t_\epsilon{+}T$ $(T \leq N)$ such that $\mathcal{X}_M(t_\epsilon{+}T, \epsilon_T) \cap \mathcal{N}$ or $\mathcal{X}_m(t_\epsilon{+}T, \epsilon_T) \cap \mathcal{N}$ is empty. This means that all normal nodes have value at most $A_M - \epsilon_T$ or at least $A_m + \epsilon_T$. Therefore,

$$\begin{aligned}\exists T, (T \leq N) \wedge ((\forall i, i \in \mathcal{N}, x_i(T) < A_M) \\ \vee\ (\forall i, i \in \mathcal{N}, x_i(T) > A_m)\end{aligned}$$

or equivalently,

$$\exists t, M(t) < A_M \vee A_m < m(t) \tag{4}$$

But we know that

$$\forall t, A_M \leq M(t) \wedge m(t) \leq A_m \tag{5}$$

We can observe that the inequalities 4 and 5 are contradictory. Hence, it must be the case that $A_M = A_m$, i.e., the limits of $M(t)$ and $m(t)$ converge as t approaches infinity. Thus, resilient asymptotic consensus is achieved. This ends the proof of the sufficiency condition. □

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

```
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 [27] 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 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.
```

where `wts_well_behaved` is defined in Coq as stated in the section 3.1.

The second hypothesis means that the maximum number of adversary nodes in the graph is less than the total number of vertices, D, and there is at least one normal node in the graph. We define `r_s_robustness` in Coq as

```
Definition r_s_robustness (r s:nat):=
nonempty_nontrivial_graph ∧ ((1 ≤ s ≤ |D|)%N →
∀ (S1 S2: {set D}),
(S1 ⊂ Vertex ∧ (|S1|>0)%N) →
(S2 ⊂ Vertex ∧ (|S2|>0)%N) →
[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))%N )).
```

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)%N].
```

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)%Re ∧
        (M mal init A w t ≤ M mal init A w 0)%Re).
```

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 `Reslient_asymptotic_consensus`: $\forall t, m(0) \leq m(t) \wedge M(t) \leq M(0)$, and $\exists L, \forall i, i \in \mathcal{N} \rightarrow \lim_{t\to\infty} x_i(t) = L$. We state the first sub proof as the following lemma statement in Coq

```
Lemma interval_bound (A: D → bool) (mal : nat → D → R)
(init : D → R) (w: nat → D * D → R):
F_total_malicious mal init A w →
wts_well_behaved A mal init w →
(0 < F + 1 ≤ |D|)%N →
∀ t : nat,
    (m mal init A w 0 ≤ m mal init A w t)%Re ∧
    (M mal init A w t ≤ M mal init A w 0)%Re.
```

The proof of lemma `interval_bound` is a consequence of lemma 3.2. We prove this lemma by an induction on time t and then apply lemma 3.2 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. 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)%Re].
```

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))%Re →
  [disjoint (X_M_t_e_i (eps_j l eps_0 eps a)
        A_M (t_eps+l)%N mal init A w) &
    (X_m_t_e_i (eps_j l eps_0 eps a)
        A_m (t_eps+l)%N 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 [27], 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 $(F + 1 \leq |\mathcal{X}^{F+1}_{\mathcal{X}_M}| + |\mathcal{X}^{F+1}_{\mathcal{X}_m}|)$ 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 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 $|\mathcal{X}^{F+1}_{\mathcal{X}_M(t_\epsilon+l,\epsilon_l)}| = |\mathcal{X}_M(t_\epsilon + l, \epsilon_l)|$, $0 < |\mathcal{X}^{F+1}_{\mathcal{X}_M(t_\epsilon+l,\epsilon_l)}|$. Hence by definition of $\mathcal{X}^{F+1}_{\mathcal{X}_M(t_\epsilon+l,\epsilon_l)}$, 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|)%N →
    is_lim_seq [eta m mal init A w] A_m →
    (0 < N)%N → (1 ≤ N)%N → (0 < a < 1)%Re →
    (eps < a^N / (1 - a^N) * eps_0)%Re →
    ∃ i:D,
    i ∈ (X_m_t_e_i (eps_j l eps_0 eps a) A_m
        (t_eps + l)%N mal init A w) ∧
    i ∈ Normal A.
```

By symmetry, we prove that $0 < |\mathcal{X}^{F+1}_{\mathcal{X}_m(t_\epsilon+l,\epsilon_l)}|$.

The other part that was not very clear from the paper proof in the original paper [27] was that the largest value that the node i uses at time step $t_\epsilon + l$ is $M(t_\epsilon + l)$. They merely state this statement instead of providing a proof for it. 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 3.2, bounding the values for the nodes positioned in the right of the normal node i was a bit non 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 following lemma statement

```
Lemma x_right_ineq_1 (i:D) (a A_M:R) (t_eps l:nat)
(eps eps_0: posreal) (mal : nat → D → R) (init: D → R)
(A: D → bool) (w: nat → D * D → R):
  F_total_malicious mal init A w →
  i ∈ Normal A →
  (0 < a)%Re → (a < 1)%Re →
  let incl :=  incl_neigh_minus_extremes i
       (x mal init A w (t_eps + l)%N) in
  (∀ k:D, k ∈ incl → (a ≤ w (t_eps + l)%N (i, k))%Re) →
    ∑i0<size incl ∧ i0>index i incl
    (x mal init A w (t_eps + l)%N (nth i incl i0) *
        w (t_eps + l)%N (i, nth i incl i0))%Re ≤
    ∑i0<size incl ∧ i0>index i incl
    w (t_eps + l)%N (i, nth i incl i0)) *
       M mal init A w (t_eps + l)%N)%Re.
```

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 4 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)%N → (s1 1%N + s2 1%N < N)%N →
  (∀ l:nat, (0 < l)%N → (l ≤ N)%N → (0< s1 l)%N →
    (0 < s2 l)%N →
      (s1 l ≤ s1 l.-1)%N ∧ (s2 l ≤ s2 l.-1)%N ∧
      ((  s1 l < s1 l.-1 )%N ∨ (s2 l < s2 l.-1)%N)) →
  ∃ T:nat, (T ≤ N)%N ∧ (s1 T = 0%N ∨ s2 T = 0%N)
```

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 with 5 and complete the proof of the sufficiency.

3.3 Proof of necessity

The lemma that we formalize is stated as follows:

Lemma 3.4. *[27] 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 3.4.

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 $\lim_{t\to\infty} x_i(t) \neq \lim_{t\to\infty} x_j(t)$.

Define the adversary set to be $\chi_{S_1}^{F+1} \cup \chi_{S_2}^{F+1}$. Then we know there exists a normal node in $S_1 \setminus \chi_{S_1}^{F+1}$, and in $S_2 \setminus \chi_{S_2}^{F+1}$. This follows because $|\chi_{S_1}^{F+1}| \neq |S_1|$ which implies $|\chi_{S_1}^{F+1}| < |S_1|$, so since $|\chi_{S_1}^{F+1}| \subset |S_1|$, there exists a node in $S_1 \setminus \chi_{S_1}^{F+1}$ which by definition must be normal, and likewise for S_2. We initialize all nodes in S_1 to have value 0, all nodes in S_2 to have value 1, and all nodes not in S_1, or S_2 to have value $\frac{1}{2}$. Furthermore, We fix all values of nodes in $\chi_{S_1}^{F+1}$ to be 0, and all nodes in $\chi_{S_2}^{F+1}$ to be 1 for all time. We inductively prove that $\forall k_1 \in S_1,\ x_{k_1}(t) = 0$, and $\forall k_2 \in S_2,\ x_{k_2}(t) = 1$ forall t. If k_1 or k_2 are adversary nodes we are done, so assume they are normal. Note that the base case where $t = 0$ is clear from the definitions.

To prove the inductive case note that with these sets of choices, (by definition) all nodes in $S_1 \setminus \chi_{S_1}^{F+1}$ receive at most F values from nodes outside S_1. Since all other inputs a node in $S_1 \setminus \chi_{S_1}^{F+1}$ receives are from S_1, which are all 0 by induction, the W-MSR procedure removes all nodes that are not zero from the set of neighbors it considers for it's update procedure. Hence at time $t+1$, the node in consideration still has value 0. For a similar reason for any node $i \in S_2$, and for all of its neighbors $j \in \mathcal{J}_i \setminus \mathcal{R}_i(t)$, $x_j(t) = 1$. The only difference to prove the result for S_2, is that we must have a set of weights that are well behaved, so that when a given node in S_2 performs the update step of the W-MSR procedure, the weights, and hence the weighted average, sum to 1. One such set of weights is $w_{ij}(t) := \frac{1}{|\mathcal{J}_i \setminus \mathcal{R}_i(t)|}$ if $j \in \mathcal{J}_i \setminus \mathcal{R}_i(t)$, and $w_{ij}(t) := 0$ otherwise. Therefore, $\exists k_1, \in S_1 \cap \mathcal{N}, k_2 \in S_2 \cap \mathcal{N}$ such that $\forall t \in \mathbb{N}$, $x_{k_1}(t) = 0$, and $x_{k_2}(t) = 1$ which implies that $\lim_{t\to\infty} x_{k_1}(t) = 0 \neq 1 = \lim_{t\to\infty} x_{k_2}(t)$, hence resilient asymptotic consensus is not achieved.

□

Formalization in Coq. We formalize the lemma 3.4 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 the proof `necessity_proof` exposed some inconsistencies in definitions in the original paper [27]. In particular, the paper defines those three conditions on weights, that we discussed in the section 2, 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 $x_i(t+1) \neq \sum_{j \in \mathcal{J}_i \setminus \mathcal{R}_i(t)} w_{ij}(t) x_j^i(t)$. 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 very clear in the original paper [27] 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) →
    ((∀ (A:D → bool) (mal:nat → D → R) (init:D → R),
    Resilient_asymptotic_consensus A mal init) ↔
    r_s_robustness (F + 1) (F + 1)).
```

However, based on the proof of necessity we discovered that 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 the 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 $(x_i(t+1), w_{ij}(t))$ is returned when $two_t = (2 \cdot t) + 1$.

3.4 Formal proof of the main theorem

We state the main theorem statement 3.1 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. [8] have used model checking techniques to formally verify the implementation of a part of safety logic for railway interlocking system. Schrer et al. [39] extended the JavaPathFinder [21] 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 [36], robust model predictive control [12] and automatic verification of sequential control systems [31]. Other approaches for verifying safety use reachability methods like flow pipe approximations [7], zonotope approximation algorithms [2, 15, 25], and ellipsoidal calculus [4].

There has also been significant work in the formalization of control theory using interactive theorem provers [1, 34, 35]. In the area of formalization of stability analysis for control theory, Cyril Cohen and Damien Rouhling formalized the LaSalle's principle in Coq [9]. 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. [6] formalize safety properties like Lyapunov stability and Exponential stability, of cyber-physical systems, in Coq. In [35], Damien Rouhling formalized the soundness of a control function [28] for an inverted pendulum. Some works have also emerged in the area of signal processing for controls. Gallois-Wang et al. [14] 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 [13]. Rashid et al. [34] formalized the transform methods in HOL-Light [19]. 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. Jasmin et al [23] proved one of the most fundamental and general result of nonlinear feedback system - the *Small-gain theorem (SGT)*, formally using Isabelle/HOL [33]. Hasan et al [20] 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 work 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 [27] 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. We leverage the existing formalization of sets, sequences, graph theory in the `mathcomp` library and the formalization of real analysis in `Coquelicot` library. 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. Overall our work is a first of its kind to provide formal specifications of a consensus algorithm in distributed controls.

5.1 Future directions

A possible future direction of work is to verify the implementation of the algorithm. The proof of this algorithm in the original paper [27], 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.

5.2 Effort

The total length of Coq proofs is about 11k lines of code. It took us 6 person months for the entire formalization.

References

[1] Reynald Affeldt and Cyril Cohen. 2017. Formal foundations of 3D geometry to model robot manipulators. In *Proceedings of the 6th ACM SIGPLAN Conference on Certified Programs and Proofs*. 30–42.

[2] Matthias Althoff and Bruce H Krogh. 2011. Zonotope bundles for the efficient computation of reachable sets. In *2011 50th IEEE conference on decision and control and European control conference*. IEEE, 6814–6821.

[3] Dejanira Araiza-Illan, Kerstin Eder, and Arthur Richards. 2014. Formal verification of control systems' properties with theorem proving. In *2014 UKACC International Conference on Control (CONTROL)*. IEEE, 244–249.

[4] Oleg Botchkarev and Stavros Tripakis. 2000. Verification of hybrid systems with linear differential inclusions using ellipsoidal approximations. In *International Workshop on Hybrid Systems: Computation and Control*. Springer, 73–88.

[5] Miguel Castro, Barbara Liskov, et al. 1999. Practical byzantine fault tolerance. In *OSDI*, Vol. 99. 173–186.

[6] Matthew Chan, Daniel Ricketts, Sorin Lerner, and Gregory Malecha. 2016. Formal verification of stability properties of cyber-physical systems. *Proc. CoqPL* (2016).

[7] Alongkrit Chutinan and Bruce H Krogh. 1999. Verification of polyhedral-invariant hybrid automata using polygonal flow pipe approximations. In *International workshop on hybrid systems: computation and control*. Springer, 76–90.

[8] Alessandro Cimatti, Fausto Giunchiglia, Giorgio Mongardi, Dario Romano, Fernando Torielli, and Paolo Traverso. 1998. Formal verification of a railway interlocking system using model checking. *Formal aspects of computing* 10, 4 (1998), 361–380.

[9] Cyril Cohen and Damien Rouhling. 2017. A formal proof in Coq of LaSalle's invariance principle. In *International Conference on Interactive Theorem Proving*. Springer, 148–163.

[10] Ankush Desai, Tommaso Dreossi, and Sanjit A Seshia. 2017. Combining model checking and runtime verification for safe robotics. In *International Conference on Runtime Verification*. Springer, 172–189.

[11] Christian Doczkal and Damien Pous. 2020. Graph theory in Coq: Minors, treewidth, and isomorphisms. *Journal of Automated Reasoning* 64, 5 (2020), 795–825.

[12] Samira S Farahani, Vasumathi Raman, and Richard M Murray. 2015. Robust model predictive control for signal temporal logic synthesis. *IFAC-PapersOnLine* 48, 27 (2015), 323–328.

[13] Jean-Christophe Filliâtre and Andrei Paskevich. 2013. Why3—where programs meet provers. In *European symposium on programming*. Springer, 125–128.

[14] Diane Gallois-Wong, Sylvie Boldo, and Thibault Hilaire. 2018. A Coq formalization of digital filters. In *International Conference on Intelligent Computer Mathematics*. Springer, 87–103.

[15] Antoine Girard and Colas Le Guernic. 2008. Zonotope/hyperplane intersection for hybrid systems reachability analysis. In *International Workshop on Hybrid Systems: Computation and Control*. Springer, 215–228.

[16] Aman Goel and Karem Sakallah. 2021. On symmetry and quantification: A new approach to verify distributed protocols. In *NASA Formal Methods Symposium*. Springer, 131–150.

[17] Georges Gonthier and Assia Mahboubi. 2010. An introduction to small scale reflection in Coq. *Journal of formalized reasoning* 3, 2 (2010), 95–152.

[18] Travis Hance, Marijn Heule, Ruben Martins, and Bryan Parno. 2021. 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)*. 115–131.

[19] John Harrison. 1996. HOL Light: A tutorial introduction. In *International Conference on Formal Methods in Computer-Aided Design*. Springer, 265–269.

[20] Osman Hasan and Muhammad Ahmad. 2013. Formal analysis of steady state errors in feedback control systems using HOL-light. In *2013 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 1423–1426.

[21] Klaus Havelund and Thomas Pressburger. 2000. Model checking java programs using java pathfinder. *International Journal on Software Tools for Technology Transfer* 2, 4 (2000), 366–381.

[22] Chris Hawblitzel, Jon Howell, Manos Kapritsos, Jacob R Lorch, Bryan Parno, Michael L Roberts, Srinath Setty, and Brian Zill. 2015. IronFleet: proving practical distributed systems correct. In *Proceedings of the 25th Symposium on Operating Systems Principles*. 1–17.

[23] Omar A Jasim and Sandor M Veres. 2017. Towards formal proofs of feedback control theory. In *2017 21st International Conference on System Theory, Control and Computing (ICSTCC)*. IEEE, 43–48.

[24] Roger M. Kieckhafer and Mohammad H. Azadmanesh. 1994. Reaching approximate agreement with mixed-mode faults. *IEEE Transactions on Parallel and Distributed Systems* 5, 1 (1994), 53–63.

[25] Niklas Kochdumper and Matthias Althoff. 2020. Sparse polynomial zonotopes: A novel set representation for reachability analysis. *IEEE Trans. Automat. Control* 66, 9 (2020), 4043–4058.

[26] Leslie Lamport et al. 2001. Paxos made simple. *ACM Sigact News* 32, 4 (2001), 18–25.

[27] Heath J LeBlanc, Haotian Zhang, Xenofon Koutsoukos, and Shreyas Sundaram. 2013. Resilient asymptotic consensus in robust networks. *IEEE Journal on Selected Areas in Communications* 31, 4 (2013), 766–781.

[28] Rogelio Lozano, Isabelle Fantoni, and Dan J Block. 2000. Stabilization of the inverted pendulum around its homoclinic orbit. *Systems & control letters* 40, 3 (2000), 197–204.

[29] Haojun Ma, Aman Goel, Jean-Baptiste Jeannin, Manos Kapritsos, Baris Kasikci, and Karem A Sakallah. 2019. I4: incremental inference of inductive invariants for verification of distributed protocols. In *Proceedings of the 27th ACM Symposium on Operating Systems Principles*. 370–384.

[30] Mehran Mesbahi and Magnus Egerstedt. 2010. *Graph theoretic methods in multiagent networks*. Princeton University Press.

[31] Il Moon, Gary J Powers, Jerry R Burch, and Edmund M Clarke. 1992. Automatic verification of sequential control systems using temporal logic. *AIChE Journal* 38, 1 (1992), 67–75.

[32] Diego Ongaro and John Ousterhout. 2014. In search of an understandable consensus algorithm. In *2014 {USENIX} Annual Technical Conference ({USENIX}{ATC} 14)*. 305–319.

[33] Lawrence C Paulson. 1994. *Isabelle: A generic theorem prover*. Springer.

[34] Adnan Rashid and Osman Hasan. 2017. Formalization of transform methods using HOL light. In *International Conference on Intelligent Computer Mathematics*. Springer, 319–332.

[35] Damien Rouhling. 2018. 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*. 28–41.

[36] Sadra Sadraddini and Calin Belta. 2015. Robust temporal logic model predictive control. In *2015 53rd Annual Allerton Conference on Communication, Control, and Computing (Allerton)*. IEEE, 772–779.

[37] David Saldana, Amanda Prorok, Shreyas Sundaram, Mario FM Campos, and Vijay Kumar. 2017. Resilient consensus for time-varying networks of dynamic agents. In *2017 American control conference (ACC)*. IEEE, 252–258.

[38] Kelsey Saulnier, David Saldana, Amanda Prorok, George J Pappas, and Vijay Kumar. 2017. Resilient flocking for mobile robot teams. *IEEE Robotics and Automation letters* 2, 2 (2017), 1039–1046.

[39] Sebastian Scherer, Flavio Lerda, and Edmund M Clarke. 2005. Model checking of robotic control systems. (2005).

[40] Fred B. Schneider. 1990. Implementing Fault-Tolerant Services Using the State Machine Approach: A Tutorial. *ACM Comput. Surv.* 22, 4 (dec 1990), 299–319. https://doi.org/10.1145/98163.98167

[41] James Usevitch, Kunal Garg, and Dimitra Panagou. 2018. Finite-Time Resilient Formation Control with Bounded Inputs. In *2018 IEEE Conference on Decision and Control (CDC)*. IEEE, 2567–2574. https://doi.org/10.1109/CDC.2018.8619697

[42] Robbert Van Renesse and Deniz Altinbuken. 2015. Paxos made moderately complex. *ACM Computing Surveys (CSUR)* 47, 3 (2015), 1–36.

[43] James R Wilcox, Doug Woos, Pavel Panchekha, Zachary Tatlock, Xi Wang, Michael D Ernst, and Thomas Anderson. 2015. Verdi: a framework for implementing and formally verifying distributed systems. In *Proceedings of the 36th ACM SIGPLAN Conference on Programming Language Design and Implementation*. 357–368.

[44] Jianan Yao, Runzhou Tao, Ronghui Gu, Jason Nieh, Suman Jana, and Gabriel Ryan. 2021. DistAI: Data-driven Automated Invariant Learning for Distributed Protocols. In *15th USENIX Symposium on Operating Systems Design and Implementation (OSDI 21)*. 405–421.

[45] Haotian Zhang and Shreyas Sundaram. 2012. Robustness of information diffusion algorithms to locally bounded adversaries. In *2012 American Control Conference (ACC)*. IEEE, 5855–5861.