



# Verification of Threshold-Based Distributed Algorithms by Decomposition to Decidable Logics

Idan Berkovits<sup>1(✉)</sup>, Marijana Lazić<sup>2,3</sup>, Giuliano Losa<sup>4</sup>, Oded Padon<sup>5</sup>, and Sharon Shoham<sup>1</sup>

<sup>1</sup> Tel Aviv University, Tel Aviv-Yafo, Israel

[berkovits@mail.tau.ac.il](mailto:berkovits@mail.tau.ac.il)

<sup>2</sup> TU Wien, Vienna, Austria

<sup>3</sup> TU Munich, Munich, Germany

<sup>4</sup> University of California, Los Angeles, USA

<sup>5</sup> Stanford University, Stanford, USA



**Abstract.** Verification of fault-tolerant distributed protocols is an immensely difficult task. Often, in these protocols, *thresholds* on set cardinalities are used both in the process code and in its correctness proof, e.g., a process can perform an action only if it has received an acknowledgment from at least half of its peers. Verification of threshold-based protocols is extremely challenging as it involves two kinds of reasoning: first-order reasoning about the unbounded state of the protocol, together with reasoning about sets and cardinalities. In this work, we develop a new methodology for decomposing the verification task of such protocols into *two* decidable logics: EPR and BAPA. Our key insight is that such protocols use thresholds in a restricted way as a means to obtain certain properties of “intersection” between sets. We define a language for expressing such properties, and present two translations: to EPR and BAPA. The EPR translation allows verifying the protocol while assuming these properties, and the BAPA translation allows verifying the correctness of the properties. We further develop an algorithm for automatically generating the properties needed for verifying a given protocol, facilitating fully automated deductive verification. Using this technique we have verified several challenging protocols, including Byzantine one-step consensus, hybrid reliable broadcast and fast Byzantine Paxos.

## 1 Introduction

Fault-tolerant distributed protocols play an important role in the avionic and automotive industries, medical devices, cloud systems, blockchains, etc. Their unexpected behavior might put human lives at risk or cause a huge financial loss. Therefore, their correctness is of ultimate importance.

Ensuring correctness of distributed protocols is a notoriously difficult task, due to the unbounded number of processes and messages, as well as the non-deterministic behavior caused by the presence of faults, concurrency, and message delays. In general, the problem of verifying such protocols is undecidable.

This imposes two directions for attacking the problem: (i) developing fully-automatic verification techniques for *restricted* classes of protocols, or (ii) designing deductive techniques for a wide range of systems that *require user assistance*. Within the latter approach, recently emerging techniques [29] leverage decidable logics that are supported by mature automated solvers to significantly reduce user effort, and increase verification productivity. Such logics bring several key benefits: (i) their solvers usually enjoy stable performance, and (ii) whenever annotations provided by the user are incorrect, the automated solvers can provide a counterexample for the user to examine.

Deductive verification based on decidable logic requires a logical formalism that satisfies two conflicting criteria: the formalism should be expressive enough to capture the protocol, its correctness properties, its inductive invariants, and ultimately its verification conditions. At the same time, the formalism should be decidable and have an effective automated tool for checking verification conditions.

In this paper we develop a methodology for deductive verification of *threshold-based* distributed protocols using decidable logic, well-established decidable logics to settle the tension explained above.

In threshold-based protocols, a process may take different actions based on the number of processes from which it received certain messages. This is often used to achieve fault-tolerance. For example, a process may take a certain step once it has received an acknowledgment from a strict majority of its peers, that is, from more than  $n/2$  processes, where  $n$  is the total number of processes. Such expressions as  $n/2$ , are called *thresholds*, and in general they can depend on additional parameters, such as the maximal number of crashed processes, or the maximal number of Byzantine processes.

Verification of such protocols requires two flavors of reasoning, as demonstrated by the following example. Consider the Paxos [20] protocol, in which each process proposes a value and all must agree on a common proposal. The protocol tolerates up to  $t$  process crashes, and ensures that every two processes that decide agree on the decided value. The protocol requires  $n > 2t$  processes, and each process must obtain confirmation messages from  $n - t$  processes before making a decision. The protocol is correct due to, among others, the fact that if  $n > 2t$  then any two sets of  $n - t$  processes have a process in common. To verify this protocol we need to express (i) relationships between an unbounded number of processes and values, which typically requires quantification over uninterpreted domains (“every two processes”), and (ii) properties of sets of certain cardinalities (“any two sets of  $n - t$  processes intersect”). Crucially, these two types of reasoning are intertwined, as the sets of processes for which we need to capture cardinalities may be defined by their relations with other state components (“messages from at least  $n - t$  processes”). While uninterpreted first-order logic (FOL) seems like the natural fit for the first type of reasoning, it is seemingly a poor fit for the second type, since it cannot express set cardinalities and the arithmetic used to define thresholds. Typically, logics that combine both types of reasoning are either undecidable or not flexible enough to capture protocols as intricate as the ones we consider.

The approach we present relies on the observation that threshold-based protocols and their correctness proofs use set cardinality thresholds in a restricted way as a means to obtain certain properties between sets, and that these properties can be expressed in FOL via a suitable encoding. In the example above, the important property is that every two sets of cardinality at least  $n - t$  have a non-empty intersection. This property can be encoded in FOL by modeling sets of cardinality at least  $n - t$  using an uninterpreted sort along with a membership relation between this sort and the sort for processes. However, the validity of the property under the assumption that  $n > 2t$  cannot be verified in FOL.

The key idea of this paper is, hence, to decompose the verification problem of threshold-based protocols into the following problems: (i) Checking protocol correctness assuming certain intersection properties, which can be reduced to verification conditions expressed in the Effectively Propositional (EPR) fragment of FOL [25, 35]. (ii) Checking that sets with cardinalities adhering to the thresholds satisfy the intersection properties (under the protocol assumptions), which can be reduced to validity checks in quantifier-free Boolean Algebra with Presburger Arithmetic (BAPA) [19]. Both BAPA and EPR are decidable logics, and are supported by mature solvers.

A crucial step in employing this decomposition is finding suitable intersection properties that are strong enough to imply the protocol's correctness (i.e., imply the FOL verification conditions), and are also implied by the precise definitions of the thresholds and the protocol's assumptions. Thus, these intersection properties can be viewed as *interpolants* between the FOL verification conditions and the thresholds in the context of the protocol's assumptions. We present fully automated procedures to find such intersection property interpolants, either eagerly or lazily.

The main contributions of this paper are<sup>1</sup>:

1. We define a threshold intersection property (TIP) language for expressing properties of sets whose cardinalities adhere to certain thresholds; TIP is expressive enough to capture the properties required to prove the correctness of challenging threshold-based protocols.
2. We develop two encodings of TIP, one in BAPA, and another in EPR. These encodings facilitate decomposition of protocol verification into decidable EPR and (quantifier-free) BAPA queries.
3. We show that there are only finitely many TIP formulas (up to equivalence) that are valid for any given protocol. Moreover, we present an effective algorithm for computing all TIP formulas valid for a given protocol, as well as an algorithm for lazily finding a set of TIP formulas that suffice to prove a given protocol.
4. Put together, we obtain an effective deductive verification approach for threshold-based protocols: the user models the protocol and its inductive invariants in EPR using a suitable encoding of thresholds, and defines the

---

<sup>1</sup> An extended version of this paper, which includes additional details and proofs, appears in [3].

thresholds and the protocol’s assumptions using arithmetic; verification is carried out automatically via decomposition to well-established decidable logics.

- We implement the approach, leveraging mature existing solvers (Z3 and CVC4), and evaluate it by verifying several challenging threshold-based protocols with sophisticated thresholds and assumptions. Our evaluation shows the effectiveness and flexibility of our approach in modeling and verifying complex protocols, including the feasibility of automatically inferring threshold intersection properties.

## 2 Preliminaries

**Transition Systems in FOL.** We model distributed protocols as transition systems expressed in many-sorted FOL. A state of the system is a first-order (FO) structure  $s = (\mathcal{D}, \mathcal{I})$  over a vocabulary  $\Sigma$  that consists of sorted constant, function and relation symbols, s.t.  $s$  satisfies a finite set of *axioms*  $\Theta$  in the form of closed formulas over  $\Sigma$ .  $\mathcal{D}$  is the *domain* of  $s$  mapping each sort to a set of objects (elements), and  $\mathcal{I}$  is the *interpretation function*. A FO *transition system* is a tuple  $(\Sigma, \Theta, I, TR)$ , where  $\Sigma$  and  $\Theta$  are as above,  $I$  is a closed formula over  $\Sigma$  that defines the *initial states*, and  $TR$  is a closed formula over  $\Sigma \uplus \Sigma'$  that defines the *transition relation* where  $\Sigma$  describes the source state of a transition and  $\Sigma' = \{a' \mid a \in \Sigma\}$  describes the target state. We require that  $TR$  does not modify any symbol that appears in  $\Theta$ . The set of reachable states is defined as usual. In practice, we define FO transition systems using a modeling language with a convenient syntax [29].

**Properties and Inductive Invariants.** A *safety property* is expressed by a closed FO formula  $P$  over  $\Sigma$ . The system is *safe* if all of its reachable states satisfy  $P$ . A closed FO formula  $Inv$  over  $\Sigma$  is an *inductive invariant* for a transition system  $(\Sigma, \Theta, I, TR)$  and property  $P$  if the following formulas, called the *verification conditions*, are valid (equivalently, their negations are unsatisfiable): (i)  $\Theta \rightarrow (I \rightarrow Inv)$ , (ii)  $\Theta \rightarrow (Inv \wedge TR \rightarrow Inv')$  and (iii)  $\Theta \rightarrow (Inv \rightarrow P)$ , where  $Inv'$  results from substituting every symbol in  $Inv$  by its primed version. We also use inductive invariants to verify arbitrary first-order LTL formulas via the reduction of [30, 31].

**Effectively Propositional Logic (EPR).** The effectively-propositional (EPR) fragment of FOL is restricted to formulas without function symbols and with a quantifier prefix  $\exists^* \forall^*$  in prenex normal form. Satisfiability of EPR formulas is decidable [25]. Moreover, EPR formulas enjoy the *finite model property*, i.e.,  $\varphi$  is satisfiable iff it has a finite model. We consider a straightforward extension of EPR that maintains these properties and is supported by solvers such as Z3 [5]. The extension allows function symbols and quantifier alternations as long as the formula’s *quantifier alternation graph*, denoted  $QA(\varphi)$ , is acyclic. For  $\varphi$  in negation normal form,  $QA(\varphi)$  is a directed graph where the set of vertices is

the set of sorts and the set of edges is defined as follows: every function symbol introduces edges from its arguments' sorts to its image's sort, and every existential quantifier  $\exists x$  that resides in the scope of universal quantifiers introduces edges from the sorts of the universally quantified variables to the sort of  $x$ . The quantifier alternation graph is extended to sets of formulas as expected.

**Boolean Algebra with Presburger Arithmetic (BAPA).** Boolean Algebra with Presburger Arithmetic (BAPA) [19] is a FO theory defined over two sorts: **int** (for integers), and **set** (for subsets of a finite universe). The language is defined as follows:

$$F ::= B_1 = B_2 \mid L_1 = L_2 \mid L_1 < L_2 \mid F_1 \wedge F_2 \mid F_1 \vee F_2 \mid \neg F \mid \exists x.F \mid \forall x.F \mid \exists u.F \mid \forall u.F \\ B ::= x \mid \emptyset \mid \mathbf{a} \mid B_1 \cup B_2 \mid B_1 \cap B_2 \mid B^c \quad L ::= u \mid K \mid \mathbf{n} \mid i \mid L_1 + L_2 \mid K \cdot L \mid |B|$$

where  $L$  defines linear integer terms, where  $u$  denotes an integer variable,  $k \in K$  defines an (interpreted) integer constant symbol  $\dots, -2, -1, 0, 1, 2 \dots$ ,  $\mathbf{n}$  is an integer constant symbol that represents the size of the finite set universe,  $i$  is an uninterpreted integer constant symbol (as opposed to the constant symbols from  $K$ ), and  $|b|$  denotes set cardinality;  $B$  defines set terms, where  $x$  denotes a set variable,  $\emptyset$  is a (interpreted) set constant symbol that represents the empty set, and  $\mathbf{a}$  is an uninterpreted set constant symbol; and  $F$  defines the set of BAPA formulas, where  $\ell_1 = \ell_2$  and  $\ell_1 < \ell_2$  are atomic arithmetic formulas and  $b_1 = b_2$  is an atomic set formula. (Other set constraints such as  $b_1 \subseteq b_2$  can be encoded in the usual way). In the sequel, we also allow arithmetic terms of the form  $\frac{\ell}{k}$  where  $k \in K$  is a positive integer and  $\ell \in L$ , as any formula that contains such terms can be translated to an equivalent BAPA formula by multiplying by  $k$ .

A BAPA structure is  $s_B = (\mathcal{D}, \mathcal{I})$  where the domain  $\mathcal{D}$  maps sort **int** to the set of all integers and maps sort **set** to the set of all subsets of a finite universe  $U$ , called the *universal set*. The semantics of terms and formulas is as expected, where the interpretation of the complement operation is defined with respect to  $U$  (e.g.,  $\mathcal{I}(\emptyset^c) = U$ ), and the integer constant  $\mathbf{n}$  is interpreted to the size of  $U$ , i.e.  $\mathcal{I}(\mathbf{n}) = |U|$ .

Both validity and satisfiability of BAPA formulas (with arbitrary quantification) are decidable [19], and the quantifier-free fragment is supported by CVC4 [2].

### 3 First-Order Modeling of Threshold-Based Protocols

Next we explain our modeling of threshold-based protocols as transition systems in FOL (Note that FOL cannot directly express set cardinality constraints). The idea is to capture each threshold by a designated sort, such that elements of this sort represent sets of nodes that satisfy the threshold. Elements of the threshold sort are then used instead of the actual threshold in the description of

the protocol and in the verification conditions. For verification to succeed, some properties of the sets satisfying the cardinality threshold must be captured in FOL. This is done by introducing additional assumptions (formally, axioms of the transition system) expressed in FOL, as discussed in Sect. 4.

<pre> 1  Input: <math>v_p</math> 2  broadcast <math>v_p</math> to all processes 3  wait until <math>n - t</math> messages have been received 4 5  if there exists <math>v</math> s.t. more than <math>\frac{n+3t}{2}</math> 6    messages contain value <math>v</math> then 7    DECIDE(<math>v</math>) 8  if there exists exactly one <math>v</math> s.t. more than 9    <math>\frac{n-t}{2}</math> messages contain value <math>v</math> then 10   <math>v_p := v</math> 11  call underlying-consensus(<math>v_p</math>) </pre>	<pre> 1  sort node, value, set<math>_{n-t}</math>, set<math>_{\frac{n+3t+1}{2}}</math>, set<math>_{\frac{n-t+1}{2}}</math> 2  ... 3  assume <math>\exists q : \text{set}_{n-t} . \forall m : \text{node} . \text{member}(m, q) \rightarrow</math> 4    <math>\exists u : \text{value} . \text{recv\_msg}(n, m, u)</math> 5  if <math>\exists v : \text{value} . q : \text{set}_{\frac{n+3t+1}{2}} . \forall m : \text{node} .</math> 6    <math>\text{member}(m, q) \rightarrow \text{recv\_msg}(n, m, v)</math> then 7    decision(<math>n, v</math>) := true 8  if <math>\exists v : \text{value} . \exists q : \text{set}_{\frac{n-t+1}{2}} . \forall m : \text{node} .</math> 9    <math>\text{member}(m, q) \rightarrow \text{recv\_msg}(n, m, v)</math> then 10   <math>v_p := v</math> 11  und_cons(<math>n, v_p</math>) := true </pre>
---	--

**Fig. 1.** Bosco: a one-step asynchronous Byzantine consensus algorithm [39], and an excerpt RML (relational modeling language) code of the main transition. Note that we overload the *member* relation for all threshold sorts. The formula  $\exists!x. \varphi(x)$  is a shorthand for exists and unique.

**Running Example.** We illustrate our approach using the example of Bosco—an asynchronous Byzantine fault-tolerant (BFT) consensus algorithm [39]. Its modeling in first-order logic using our technique appears alongside an informal pseudo-code in Fig. 1.

In the BFT consensus problem, each node proposes a value and correct nodes must decide on a unique proposal. BFT consensus algorithms typically require at least two communication rounds to reach a decision. In Bosco, nodes execute a preliminary communication step which, under favorable conditions, reaches an early decision, and then call an underlying BFT consensus algorithm to ensure reaching a decision even if these conditions are not met. Bosco is safe when  $n > 3t$ ; it guarantees that a preliminary decision will be reached if all nodes are non-faulty and propose the same value when  $n > 5t$  (weakly one-step condition), and even if some nodes are faulty, as long as all non-faulty nodes propose the same value, when  $n > 7t$  (strongly one-step condition).

Bosco achieves consensus by ensuring that (a) no two correct nodes decide differently in the preliminary step, and (b) if a correct node decides value  $v$  in the preliminary step then every correct process calls the underlying BFT consensus algorithm with proposal  $v$ . Property (a) is ensured by the fact that a node decides in the preliminary step only if more than  $\frac{n+3t}{2}$  nodes proposed the same value. When  $n > 3t$ , two sets of cardinality greater than  $\frac{n+3t}{2}$  have at least one non-faulty node in common, and therefore no two different values can be proposed by more than  $\frac{n+3t}{2}$  nodes. Similarly, we can derive property (b) from the fact that a set of more than  $\frac{n+3t}{2}$  nodes and a set of  $n - t$  nodes

intersect in  $\frac{n+t}{2}$  nodes, which, after removing  $t$  nodes which may be faulty, still leaves us with more than  $\frac{n-t}{2}$  nodes, satisfying the condition in line 9.

### 3.1 Threshold-Based Protocols

**Parameters and Resilience Conditions.** We consider protocols whose definitions depend on a set of *parameters*,  $Prm$ , divided into *integer parameters*,  $Prm_I$ , and *set parameters*,  $Prm_S$ .  $Prm_I$  always includes  $\mathbf{n}$ , the total number of nodes (assumed to be finite). Protocol correctness is ensured under a set of assumptions  $\Gamma$  called *resilience conditions*, formulated as BAPA formulas over  $Prm$  (this means that all the uninterpreted constants appearing in  $\Gamma$  are from  $Prm$ ). In Bosco,  $Prm_I = \{\mathbf{n}, \mathbf{t}\}$ , where  $\mathbf{t}$  is the maximal number of Byzantine failures tolerated by the algorithm, and  $Prm_S = \{\mathbf{f}\}$ , where  $\mathbf{f}$  is the set of Byzantine nodes;  $\Gamma = \{\mathbf{n} \geq 3\mathbf{t} + 1, |\mathbf{f}| \leq \mathbf{t}\}$ .

**Threshold Conditions.** Both the description of the protocol and the inductive invariant may include conditions that require the size of some set of nodes to be “at least  $t$ ”, “at most  $t$ ”, and so on, where the threshold  $t$  is of the form  $t = \frac{\ell}{k}$ , where  $k$  is a positive integer, and  $\ell$  is a ground BAPA integer term over  $Prm$  (we do not allow comparing sizes of two sets – we observe that it is not needed for threshold-based protocols). We denote the set of thresholds by  $T$ . For example, in Bosco,  $T = \{\mathbf{n} - \mathbf{t}, \frac{\mathbf{n}+3\mathbf{t}+1}{2}, \frac{\mathbf{n}-\mathbf{t}+1}{2}\}$ .

Wlog we assume that all conditions on set cardinalities are of the form “at least  $t$ ” since every condition can be written this way, possibly by introducing new thresholds:

$$|X| > \frac{\ell}{k} \equiv |X| \geq \frac{\ell+1}{k} \quad |X| \leq \frac{\ell}{k} \equiv |X^c| \geq \frac{k \cdot \mathbf{n} - \ell}{k} \quad |X| < \frac{\ell}{k} \equiv |X| \leq \frac{\ell-1}{k}$$

### 3.2 Modeling in FOL

**FO Vocabulary for Modeling Threshold-Based Protocols.** We describe the protocol’s states (e.g., pending messages, votes, etc.) using a core FO vocabulary  $\Sigma_C$  that includes sort `node` and additional sorts and symbols. Parameters  $Prm$  are *not* part of the FO vocabulary used to model the protocol. Also, we do not model set cardinality directly. Instead, we encode the cardinality thresholds in FOL by defining a FO vocabulary  $\Sigma_T^{Prm}$ :

- For every threshold  $t$  we introduce a *threshold sort*  $\mathbf{set}_t$  with the intended meaning that elements of this sort are sets of nodes whose size is at least  $t$ .
- Each sort  $\mathbf{set}_t$  is equipped with a binary relation symbol  $member_t$  between sorts `node` and  $\mathbf{set}_t$  that captures the membership relation of a node in a set.
- For each set parameter  $\mathbf{a} \in Prm_S$  we introduce a unary relation symbol  $member_{\mathbf{a}}$  over sort `node` that captures membership of a node in the set  $\mathbf{a}$ .

We then model the protocol as a transition system  $(\Sigma, \Theta, I, TR)$  where  $\Sigma = \Sigma_C \uplus \Sigma_T^{Prm}$ .

We are interested only in states (FO structures over  $\Sigma$ ) where the interpretation of the threshold sorts and membership relations is according to their intended meaning in a corresponding BAPA structure. Formally, these are  $T$ -extensions, defined as follows:

**Definition 1.** *We say that a FO structure  $s_C = (\mathcal{D}_C, \mathcal{I}_C)$  over  $\Sigma_C$  and a BAPA structure  $s_B = (\mathcal{D}_B, \mathcal{I}_B)$  over  $Prm$  are compatible if  $\mathcal{D}_B(\text{set}) = \mathcal{P}(\mathcal{D}_C(\text{node}))$ , where  $\mathcal{P}$  is the powerset operator. For such compatible structures and a set of thresholds  $T$  over  $Prm$ , the  $T$ -extension of  $s_C$  by  $s_B$  is the structure  $s = (\mathcal{D}, \mathcal{I})$  over  $\Sigma$  defined as follows:*

$$\begin{aligned} \mathcal{D}(\mathbf{s}) &= \mathcal{D}_C(\mathbf{s}) \text{ for every sort } \mathbf{s} \text{ in } \Sigma_C & \mathcal{I}(a) &= \mathcal{I}_C(a) \text{ for every } a \text{ in } \Sigma_C \\ \mathcal{D}(\text{set}_t) &= \{A \subseteq \mathcal{D}_C(\text{node}) \mid |A| \geq \mathcal{I}_B(t)\} & \mathcal{I}(\text{member}_{\mathbf{a}}) &= \mathcal{I}_B(\mathbf{a}) \\ \mathcal{I}(\text{member}_t) &= \{(e, A) \mid e \in \mathcal{D}_C(\text{node}), A \in \mathcal{D}(\text{set}_t), e \in A\} \end{aligned}$$

Note that for the  $T$ -extension  $s$  to be well defined as a FO structure, we must have that  $\mathcal{D}(\text{set}_t) \neq \emptyset$  for every threshold  $t \in T$ . This means that a  $T$ -extension by  $s_B$  only exists if  $\{A \subseteq \mathcal{D}(\text{node}) \mid |A| \geq \mathcal{I}_B(t)\} \neq \emptyset$ . This is ensured by the following condition:

**Definition 2 (Feasibility).**  *$T$  is  $\Gamma$ -feasible if  $\Gamma \models t \leq \mathbf{n}$  for every  $t \in T$ .*

**Expressing Threshold Constraints.** Cardinality constraints can be expressed in FOL over the vocabulary  $\Sigma = \Sigma_C \uplus \Sigma_T^{Prm}$  using quantification. To express that  $|\{n : \text{node} \mid \varphi(n, \bar{u})\}| \geq t$ , i.e., that there are at least  $t$  nodes that satisfy the FO formula  $\varphi$  over  $\Sigma_C$  (where  $\bar{u}$  are free variables in  $\varphi$ ), we use the following first-order formula over  $\Sigma$ :  $\exists q : \text{set}_t. \forall n : \text{node}. \text{member}_t(n, q) \rightarrow \varphi(n, \bar{u})$ . Similarly, to express the property that a node is a member of a set parameter  $\mathbf{a}$  (e.g., to check if  $n \in \mathbf{f}$ , i.e., a node is faulty) we use the FO formula  $\text{member}_{\mathbf{a}}(n)$ . For example, in Fig. 1, line 5 (right) uses the FO modeling to express the condition in line 5 (left). This modeling is sound in the following sense:

**Lemma 1 (Soundness).** *Let  $s_C = (\mathcal{D}_C, \mathcal{I}_C)$  be a FO structure over  $\Sigma_C$ ,  $s_B = (\mathcal{D}_B, \mathcal{I}_B)$  a compatible BAPA structure over  $Prm$  s.t.  $s_B \models \Gamma$  and  $T$  a  $\Gamma$ -feasible set of thresholds over  $Prm$ . Then there exists a (unique)  $T$ -extension  $s$  of  $s_C$  by  $s_B$ . Further:*

1. For every  $\mathbf{a} \in \text{Prms}$  and FO valuation  $\iota: s, \iota \models \text{member}_{\mathbf{a}}(n)$  iff  $\iota(n) \in \mathcal{I}_B(\mathbf{a})$ ,
2. For every  $t \in T$ , formula  $\varphi$ , and FO valuation  $\iota: s, \iota \models \exists q : \text{set}_t. \forall n : \text{node}. \text{member}_t(n, q) \rightarrow \varphi(n, \bar{u})$  iff  $|\{e \in \mathcal{D}(\text{node}) \mid s_C, \iota[n \mapsto e] \models \varphi(n, \bar{u})\}| \geq \mathcal{I}_B(t)$ .

**Definition 3.** *A first-order structure  $s$  over  $\Sigma$  is threshold-faithful if it is a  $T$ -extension of some  $s_C$  by some  $s_B \models \Gamma$  (as in Lemma 1).*

**Incompleteness.** Lemma 1 ensures that the FO modeling can be soundly used to verify the protocol. It also ensures that the modeling is precise on threshold-faithful structures (Def. 1). Yet, the FO transition system is not restricted to such states, hence it *abstracts* the actual protocol. To have any hope to verify the protocol, we must capture *some* of the intended meaning of the threshold sorts and relations. This is obtained by adding FO axioms to the FO transition system. Soundness is maintained as long as the axioms hold in all threshold-faithful structures. We note that the set of *all* such axioms is not recursively enumerable—this is where the essential incompleteness of our approach lies.

## 4 Decomposition via Threshold Intersection Properties

In this section, we identify a set of properties we call *threshold intersection properties*. When captured via FO axioms, these properties suffice for verifying many threshold-based protocols (all the ones we considered). Importantly, these are properties of sets adhering to the thresholds that do not involve the protocol state. As a result, they can be expressed both in FOL and in BAPA. This allows us to decompose the verification task into: (i) checking that certain threshold properties are valid in all threshold-faithful structures by checking that they are implied by  $\Gamma$  (carried out using quantifier free BAPA), and (ii) checking that the verification conditions of the FO transition-system with the same threshold properties taken as axioms are valid (carried out in first-order logic, and in EPR if quantifier alternations are acyclic).

### 4.1 Threshold Intersection Property Language

Threshold properties are expressed in the *threshold intersection property language* (TIP). TIP is essentially a subset of BAPA, specialized to have the properties listed above.

**Syntax.** We define TIP as follows, with  $t \in T$  a threshold (of the form  $\frac{\ell}{k}$ ) and  $\mathbf{a} \in \text{Prm}_S$ :

$$\begin{aligned} F ::= & B \neq \emptyset \mid B^c = \emptyset \mid g_{\geq t}(B) \mid F_1 \wedge F_2 \mid \forall x : g_{\geq t}.F \\ B ::= & \mathbf{a} \mid \mathbf{a}^c \mid x \mid x^c \mid \emptyset \mid \emptyset^c \mid B_1 \cap B_2 \end{aligned}$$

TIP restricts the use of set cardinality to *threshold guards*  $g_{\geq t}(b)$  with the meaning  $|b| \geq t$ . No other arithmetic atomic formulas are allowed. Comparison atomic formulas are restricted to  $b \neq \emptyset$  and  $b^c = \emptyset$ . Quantifiers must be guarded, and negation, disjunction and existential quantification are excluded. We forbid set union and restrict complementation to atomic set terms. We refer to such formulas as *intersection properties* since they express properties of intersections of (atomic) sets.

*Example 1.* In Bosco, the following property captures the fact that the intersection of a set of at least  $\mathbf{n} - \mathbf{t}$  nodes and a set of more than  $\frac{\mathbf{n}+3\mathbf{t}}{2}$  nodes consists of at least  $\frac{\mathbf{n}-\mathbf{t}}{2}$  non-faulty nodes. This is needed for establishing correctness of the protocol.

$$\forall x : g_{\geq \mathbf{n}-\mathbf{t}}. \forall y : g_{\geq \frac{\mathbf{n}+3\mathbf{t}+1}{2}}. g_{\geq \frac{\mathbf{n}-\mathbf{t}+1}{2}}(x \cap y \cap \mathbf{f}^c)$$

**Semantics.** As TIP is essentially a subset of BAPA, we define its semantics by translating its formulas to BAPA, where most constructs directly correspond to BAPA constructs, and guards are translated to cardinality constraints:

$$\mathcal{B}(g_{\geq \frac{\ell}{k}}(b)) \stackrel{\text{def}}{=} k \cdot |b| \geq \ell \quad \mathcal{B}(\forall x : g. \varphi) \stackrel{\text{def}}{=} \forall x. \neg \mathcal{B}(g(x)) \vee \mathcal{B}(\varphi)$$

The notions of structures, satisfaction, equivalence, validity, satisfiability, etc. are inherited from BAPA. In particular, given a set of BAPA resilience conditions  $\Gamma$  over the parameters  $Prm$ , we say that a TIP formula  $\varphi$  is  $\Gamma$ -valid, denoted  $\Gamma \models \varphi$ , if  $\Gamma \models \mathcal{B}(\varphi)$ .

If  $\Gamma$  is quantifier-free (which is the typical case),  $\Gamma$ -validity of TIP formulas can be checked via validity checks of quantifier-free BAPA formulas, supported by mature solvers. Note that  $\Gamma$ -validity of a formula of the form  $\forall x : g_{\geq t_1}. |x \cap b| \geq t_2$  is equivalent to  $\Gamma \models \forall u. u \geq t_1 \rightarrow u + |b| - n \geq t_2$ , allowing replacing quantification over sets by quantification over integers, thus improving performance of existing solvers.

## 4.2 Translation to FOL

To verify threshold-based protocols, we translate TIP formulas to FO axioms, using the threshold sorts and relations. To translate  $g_{\geq t}(b)$ , we follow the principle in (Sect. 3.2):

$$\begin{array}{ll} \mathcal{FO}(\neg \varphi) = \neg \mathcal{FO}(\varphi) & \mathcal{FO}(n \in b^c) = \neg \mathcal{FO}(n \in b) \\ \mathcal{FO}(\varphi_1 \wedge \varphi_2) = \mathcal{FO}(\varphi_1) \wedge \mathcal{FO}(\varphi_2) & \mathcal{FO}(n \in \emptyset) = \text{false} \\ \mathcal{FO}(\forall x : g. \varphi) = \forall x : \text{set}_g. \mathcal{FO}(\varphi) & \mathcal{FO}(n \in \mathbf{a}) = \text{member}_{\mathbf{a}}(n) \\ \mathcal{FO}(n \in b_1 \cap b_2) = \mathcal{FO}(n \in b_1) \wedge \mathcal{FO}(n \in b_2) & \mathcal{FO}(n \in x) = \text{member}_t(n, x) \\ \mathcal{FO}(b \neq \emptyset) = \exists n : \text{node}. \mathcal{FO}(n \in b) & \text{where } x \text{ is guarded by } t \\ \mathcal{FO}(b^c = \emptyset) = \forall n : \text{node}. \mathcal{FO}(n \in b) & \\ \mathcal{FO}(g_{\geq t}(b)) = \exists x : \text{set}_t. \forall n : \text{node}. \text{member}_t(n, x) \rightarrow \mathcal{FO}(n \in b) & \end{array}$$

We lift  $\mathcal{FO}$  to sets of formulas:  $\mathcal{FO}(\Delta) = \{\mathcal{FO}(\varphi) \mid \varphi \in \Delta\}$ .

Next, we state the soundness of the translation, which intuitively means that  $\mathcal{FO}(\varphi)$  is “equivalent” to  $\varphi$  over threshold-faithful FO structures (Definition 1). This justifies adding  $\mathcal{FO}(\varphi)$  as a FO axiom whenever  $\varphi$  is  $\Gamma$ -valid.

**Theorem 1 (Translation soundness).** *Let  $s_C = (\mathcal{D}_C, \mathcal{I}_C)$  be a first-order structure over  $\Sigma_C$ ,  $s_B = (\mathcal{D}_B, \mathcal{I}_B)$  a compatible BAPA structure over  $Prm$ , and  $s$  the T-extension of  $s_C$  by  $s_B$ . Then for every closed TIP formula  $\varphi$ , we have  $s_B \models \varphi \Leftrightarrow s \models \mathcal{FO}(\varphi)$ .*

**Corollary 1.** *For every closed TIP formula  $\varphi$  such that  $\Gamma \models \varphi$ , we have that  $\mathcal{FO}(\varphi)$  is satisfied by every threshold-faithful first-order structure.*

## 5 Automatically Inferring Threshold Intersection Properties

To apply the approach described in Sects. 3 and 4, it is crucial to find suitable threshold properties. That is, given the resilience conditions  $\Gamma$  and a FO transition system modeling the protocol, we need to find a set  $\Delta$  of TIP formulas such that (i)  $\Gamma \models \varphi$  for every  $\varphi \in \Delta$ , and (ii) the VCs of the transition system with the axioms  $\mathcal{FO}(\Delta)$  are valid.

In this section, we address the problem of automatically inferring such a set  $\Delta$ . In particular, we prove that for any protocol that satisfies a natural condition, there are finitely many  $\Gamma$ -valid TIP formulas (up to equivalence), enabling a complete automatic inference algorithm. Furthermore, we show that (under certain reasonable conditions formalized in this section), the FO axioms resulting from the inferred TIP properties have an *acyclic* quantifier alternation graph, facilitating protocol verification in EPR.

**Notation.** For the rest of this section, we fix a set  $Prm$  of parameters, a set  $\Gamma$  of resilience conditions over  $Prm$ , and a set  $T$  of thresholds. Note that  $b \neq \emptyset \equiv g_{\geq 1}(b)$  and  $b^c = \emptyset \equiv g_{\geq n}(b)$ . Therefore, for uniformity of the presentation, given a set  $T$  of thresholds, we define  $\hat{T} \stackrel{\text{def}}{=} T \cup \{1, n\}$  and replace atomic formulas of the form  $b \neq \emptyset$  and  $b^c = \emptyset$  by the corresponding guard formulas. As such, the only atomic formulas are of the form  $g_{\geq t}(b)$  where  $t \in \hat{T}$ . Note that guards in quantifiers are still restricted to  $g_{\geq t}$  where  $t \in T$ . Given a set  $Prm_S$ , we also denote  $\hat{Prm}_S = Prm_S \cup \{\mathbf{a}^c \mid \mathbf{a} \in Prm_S\}$ .

### 5.1 Finding Consequences in the Threshold Intersection Property Language

In this section, we present AIP— an algorithm for inferring all  $\Gamma$ -valid TIP formulas. A naïve (non-terminating) algorithm would iteratively check  $\Gamma$ -validity of every TIP formula. Instead, AIP prunes the search space relying on the following condition:

**Definition 4.**  *$T$  is  $\Gamma$ -non-degenerate if for every  $t \in T$  it holds that  $\Gamma \not\models t \leq 0$ .*

If  $\Gamma \models t \leq 0$  then  $t$  is degenerate in the sense that  $g_{\geq t}(b)$  is always  $\Gamma$ -valid, and  $\forall x : g_{\geq t} \cdot g_{\geq t'}(x \cap b)$  is never  $\Gamma$ -valid unless  $t'$  is also degenerate.

We observe that we can (i) push conjunctions outside of formulas (since  $\forall$  distributes over  $\wedge$ ), and assuming non-degeneracy, (ii) ignore terms of the form  $x^c$ :

**Lemma 2.** *If  $T$  is  $\Gamma$ -feasible and  $\Gamma$ -non-degenerate, then for every  $\Gamma$ -valid  $\varphi$  in  $TIP$ , there exist  $\varphi_1, \dots, \varphi_m$  s.t.  $\varphi \equiv \bigwedge_{i=1}^m \varphi_i$  and for every  $1 \leq i \leq m$ ,  $\varphi_i$  is of the form:*

$$\forall x_1 : g_{\geq t_1} \dots \forall x_q : g_{\geq t_q} \cdot g_{\geq t}(x_1 \cap \dots \cap x_q \cap a_1 \dots \cap a_k)$$

where  $q + k > 0$ ,  $t_1, \dots, t_q \in T$ ,  $t \in \hat{T}$ ,  $a_1, \dots, a_k \in \hat{Prm}_S$ , and the  $a_i$ 's are distinct.

We refer to  $\varphi_i$  of the form above as *simple*, and refer to  $g_{\geq t}$  as its *atomic guard*.

By Lemma 2, it suffices to generate all *simple*  $\Gamma$ -valid formulas. Next, we show that this can be done more efficiently by pruning the search space based on a subsumption relation that is checked syntactically avoiding  $\Gamma$ -validity checks.

**Definition 5 (Subsumption).** *For every  $h_1, h_2 \in \hat{T} \cup \hat{Prm}_S$ , we denote  $h_1 \sqsubseteq_{\Gamma} h_2$  if one of the following holds: (1)  $h_1 = h_2$ , or (2)  $h_1, h_2 \in \hat{T}$  and  $\Gamma \models h_1 \geq h_2$ , or (3)  $h_1 \in \hat{Prm}_S$ ,  $h_2 \in \hat{T}$  and  $\Gamma \models |h_1| \geq h_2$ .*

For  $h_1, h_2 \in \hat{T}$  and  $h_3 \in \hat{Prm}_S$ ,  $h_1 \sqsubseteq_{\Gamma} h_2$  means that  $\Gamma \models \forall x : g_{\geq h_1} \cdot g_{\geq h_2}(x)$ , and  $h_3 \sqsubseteq_{\Gamma} h_2$  means that  $\Gamma \models g_{\geq h_2}(h_3)$ . We lift the relation  $\sqsubseteq_{\Gamma}$  to act on simple formulas:

**Definition 6.** *Given simple formulas*

$$\begin{aligned} \alpha &= \forall x_1 : g_{\geq h_1} \dots \forall x_q : g_{\geq h_q} \cdot g_{\geq t}(x_1 \cap \dots \cap x_q \cap h_{q+1} \dots \cap h_k) \\ \beta &= \forall x_1 : g_{\geq h'_1} \dots \forall x_{q'} : g_{\geq h'_{q'}} \cdot g_{\geq t'}(x_1 \cap \dots \cap x_{q'} \cap h'_{q'+1} \dots \cap h'_{k'}) \end{aligned}$$

we say that  $\alpha \sqsubseteq_{\Gamma} \beta$  if (i)  $t \sqsubseteq_{\Gamma} t'$ , and (ii) there exists an injective function  $f : \{1, \dots, k'\} \rightarrow \{1, \dots, k\}$  s.t. for any  $1 \leq i \leq k'$  it holds that  $h'_i \sqsubseteq_{\Gamma} h_{f(i)}$ .

**Lemma 3.** *Let  $\alpha, \beta$  be simple formulas such that  $\alpha \sqsubseteq_{\Gamma} \beta$ . If  $\Gamma \models \alpha$  then  $\Gamma \models \beta$ .*

**Corollary 2.** *If no simple formula with  $q$  quantifiers is  $\Gamma$ -valid then no simple formula with more than  $q$  quantifiers is  $\Gamma$ -valid.*

Algorithm 1 depicts AIP that generates all  $\Gamma$ -valid simple formulas, relying on Lemma 3. AIP uses a naïve search strategy; different strategies can be used (e.g. [26]). Based on Corollary 2, AIP terminates if for some number of quantifiers no  $\Gamma$ -valid formula is discovered.

---

**Algorithm 1.** Algorithm for Inferring Intersection Properties (AIP)

---

**Input:**  $Prm_S, T, \Gamma$

- 1 set `checked_true` = `checked_false` = `[]` ;
- 2 **foreach**  $q = 0, 1, \dots$  **do**
- 3     **foreach** simple formula  $\varphi$  over  $T$  and  $Prm_S$  with  $q$  quantifiers **do**
- 4         **if** exists  $\psi \in \text{checked\_true}$  s.t.  $\psi \sqsubseteq_{\Gamma} \varphi$  **then** yield  $\varphi$  ;
- 5         **else if** exists  $\psi \in \text{checked\_false}$  s.t.  $\varphi \sqsubseteq_{\Gamma} \psi$  **then** continue ;
- 6         **else if**  $\Gamma \models \varphi$  **then** yield  $\varphi$  ; add  $\varphi$  to `checked_true` ;
- 7         **else** add  $\varphi$  to `checked_false` ;
- 8     **if** no formulas were added to `checked_true` **then** terminate ;

---

**Lemma 4 (Soundness).** *Every formula  $\varphi$  that is returned by the algorithm is  $\Gamma$ -valid.*

**Lemma 5 (Completeness).** *If  $T$  is  $\Gamma$ -feasible and  $\Gamma$ -non-degenerate, then for every  $\Gamma$ -valid TIP formula  $\varphi$  there exist  $\varphi_1 \dots \varphi_m$  s.t.  $\varphi \equiv \bigwedge_{i=1}^m \varphi_i$  and AIP yields every  $\varphi_i$ .*

Next, we characterize the cases in which there are finitely many  $\Gamma$ -valid TIP formulas, up to equivalence, and thus, AIP is guaranteed to terminate.

**Definition 7.**  *$T$  is  $\Gamma$ -sane if for every  $t_1, t_2 \in T$ ,  $\Gamma \not\models t_1 \leq 0 \vee t_2 > \mathbf{n} - 1$ .  $(T, \text{Prm}_S)$  is  $\Gamma$ -sane if, in addition, for every  $t_1 \in T$ ,  $a \in \text{Prm}_S$ ,  $\Gamma \not\models t_1 \leq 0 \vee |a| = \mathbf{n}$ .*

**Theorem 2.** *Assume that  $T$  is  $\Gamma$ -feasible. Then the following conditions are equivalent: (1) There are finitely many  $\Gamma$ -valid simple formulas. (2) There are finitely many  $\Gamma$ -valid TIP formulas, up to equivalence. (3)  $T$  is  $\Gamma$ -sane.*

**Corollary 3 (Termination).** *If  $T$  is  $\Gamma$ -feasible and  $\Gamma$ -sane, AIP terminates.*

## 5.2 From TIP to Axioms in EPR

The set of simple formulas generated by AIP,  $\Delta$ , is translated to FOL axioms as described in Sect. 4.2. Next, we show how to ensure that the quantifier alternation graph (Sect. 2) of  $\mathcal{FO}(\Delta)$  is acyclic. A simple formula induces quantifier alternation edges in  $QA(\mathcal{FO}(\varphi))$  from the sorts of its universal quantifiers to the sort of its atomic guard  $g_{\geq t}$  (or if  $t = 1$  to the node sort). Therefore, from Lemma 3, for a  $\Gamma$ -valid  $\varphi$ , cycles in  $QA(\mathcal{FO}(\varphi))$  may only occur if they occur in the graph obtained by  $\sqsubseteq_\Gamma$ . Furthermore, if  $QA(\mathcal{FO}(\varphi))$  is not acyclic, then the atomic guard must be equal to one of the quantifier guards. We refer to such a formula as a *cyclic formula*. We show that, under the following assumption, we can eliminate all cyclic formulas from  $\Delta$ .

**Definition 8.**  *$T$  is  $\Gamma$ -acyclic if for every  $t_1, t_2 \in T$ , if  $\Gamma \models t_1 = t_2$  then  $t_1 = t_2$ .*

Intuitively, if  $T$  is not  $\Gamma$ -acyclic, then it has (at least) two “equivalent” thresholds, making one of them redundant. If that is the case, we can alter the protocol and its proof so that one of these guards is eliminated and the other one is used instead.

**Theorem 3.** *Let  $T$  be  $\Gamma$ -feasible and  $\Gamma$ -acyclic and  $(T, \text{Prm}_S)$  be  $\Gamma$ -sane. Let  $\Delta$  be the set returned by AIP, and  $\Delta' = \{\varphi \in \Delta \mid \varphi \text{ is acyclic}\}$ . Then the VCs of the FO transition system with axioms  $\mathcal{FO}(\Delta)$  are valid iff they are valid with axioms  $\mathcal{FO}(\Delta')$ . Further,  $QA(\mathcal{FO}(\Delta'))$  is acyclic.*

## 5.3 Finding Minimal Properties Required for a Protocol

If  $\Delta$  consists of *all* acyclic  $\Gamma$ -valid TIP formulas returned by AIP, using  $\mathcal{FO}(\Delta)$  as FO axioms leads to divergence of the verifier. To overcome this, we propose two variants.

**Minimal Equivalent.**  $\Delta_{min}$ . Some of the formulas in  $\mathcal{FO}(\Delta)$  are implied by others, making them redundant. We remove such formulas using a greedy procedure that for every  $\varphi_i \in \Delta$ , checks whether  $\mathcal{FO}(\Delta \setminus \{\varphi_i\}) \models \mathcal{FO}(\varphi_i)$ , and if so, removes  $\varphi_i$  from  $\Delta$ . Note that if  $QA(\mathcal{FO}(\Delta))$  is acyclic, the check translates to (un)satisfiability in EPR.

This procedure results in  $\Delta_{min} \subseteq \Delta$  s.t.  $\mathcal{FO}(\Delta_{min}) \models \mathcal{FO}(\Delta)$  and no strict subset of  $\Delta_{min}$  satisfies this condition. That is,  $\Delta_{min}$  is a local minimum for that property.

**Interpolant.**  $\Delta_{int}$ . There may exist  $\Delta_{int} \subseteq \Delta$  s.t.  $\mathcal{FO}(\Delta_{int}) \not\models \mathcal{FO}(\Delta)$  but  $\mathcal{FO}(\Delta_{int})$  suffices to prove the first-order VCs, and enables to discharge the VCs more efficiently. We compute such a set  $\Delta_{int}$  iteratively. Initially,  $\Delta_{int} = \emptyset$ . In each iteration, we check the VCs. If a counterexample to induction (CTI) is found, we add to  $\Delta_{int}$  a formula from  $\Delta$  not satisfied by the CTI. In this approach,  $\Delta$  is not pre-computed. Instead, AIP is invoked lazily to generate candidate formulas in reaction to CTIs.

## 6 Evaluation

We evaluate the approach by verifying several challenging threshold-based distributed protocols that use sophisticated thresholds: we verify the safety of Bosco [39] (presented in Sect. 3) under its 3 different resilience conditions, the safety and liveness (using the liveness to safety reduction presented in [30]) of Hybrid Reliable Broadcast [40], and the safety of Byzantine Fast Paxos [23]. Hybrid Reliable Broadcast tolerates four different types of faults, while Fast Byzantine Paxos is a fast-learning [21, 22] Byzantine fault-tolerant consensus protocol; fast-learning protocols are notorious because two such algorithms, Zyzzyva [17] and FaB [28], were recently revealed incorrect [1] despite having been published at major systems conferences.

**Implementation.** We implemented both algorithms described in Sect. 5.3.  $AIP_{EAGER}$  eagerly constructs  $\Delta$  by running AIP, and then uses EPR reasoning to remove redundant formulas (whose FO representation is implied by the FO representation of others). To reduce the number of EPR validity checks used during this minimization step, we implemented an optimization that allows us to prove redundancy of TIP formulas internally based on an extension of the notion of subsumption from Sect. 5.  $AIP_{LAZY}$  computes a subset of  $\Delta$  while using AIP in a lazy fashion, guided by CTIs obtained from attempting to verify the FO transition system. Our implementations use CVC4 to discharge BAPA queries, and Z3 to discharge EPR queries. Verification of first-order transition systems is performed using Ivy, which internally uses Z3 as well. All experiments reported were performed on a laptop running 64-bit Windows 10, with a Core-i5 2.2 GHz CPU, using Z3 version 4.8.4, CVC4 version 1.7, and the latest version of Ivy.

Figure 2 lists the protocols we verified and the details of the evaluation. Each experiment was repeated 10 times, and we report the mean time ( $\mu$ ) and standard

Protocol	$T$	$\Gamma$	$\mathbf{V}$	$\mathbf{I}$	$\mathbf{Q}$	$\mathbf{t}_C$	$\Delta_{EAGER}^{\text{Protocol}}$	AIP <sub>LAZY</sub>			
								$\mathbf{V}$	$\mathbf{I}$	$\mathbf{CTI}^{\mathbf{Q}}$	
Bosco	$t_1 = \mathbf{n} - t$ $t_2 = \mathbf{n} + 3t + 1$ $t_3 = \mathbf{n} - 2t + 1$	$\mathbf{n} > 3t$ $ f  \leq t$	$\frac{23}{39}$ $\frac{21}{1216}$	$6$ $3s$	$\mu(12s)$ $\sigma(4s)$	$g_1(f^c)$ $\forall x: g_1 \cdot y: g_1 \cdot z: g_2 \cdot g_3 \cdot (x \cap y \cap z)$ $\forall x: g_1 \cdot y: g_2 \cdot z: g_3 \cdot x: (y \cap z \neq \emptyset)$	$\Delta_{LAZY}^{\text{Protocol}}$	$24$ $\forall x: g_1 \cdot y: g_2 \cdot g_3 \cdot (x \cap y \cap z \cap f^c)$	$6$ $\sigma(1m)$	$18$ $\mu(3m)$	
Bosco	Weakly One-step	$\mathbf{n} > 5t$ $ f  \leq t$	$\frac{16}{51}$ $\frac{24}{1204}$	$6$ $3s$	$\mu(13m)$ $\sigma(14m)$	$\Delta_{EAGER}^{\text{Bosco}}$ $\forall x: g_1 \cdot g_2 \cdot (x)$	$\Delta_{LAZY}^{\text{Bosco}}$ $\forall x: g_1 \cdot g_2 \cdot (x)$	$32$ $\forall x: g_1 \cdot g_2 \cdot (x \cap f^c)$	$7$ $\sigma(9m)$	$19$ $\mu(13m)$	
Bosco	Strongly One-step	$\mathbf{n} > 7t$ $ f  \leq t$	$\frac{26}{63}$ $\frac{24}{2407}$	$8$ $8s$	T.O.	$\Delta_{EAGER}^{\text{Bosco}}$ $\forall x: g_1 \cdot y: g_1 \cdot g_2 \cdot (x \cap y)$	$\Delta_{LAZY}^{\text{Bosco}}$ $\forall x: g_1 \cdot g_2 \cdot (x \cap f^c)$	$34$ $\forall x: g_1 \cdot g_2 \cdot (x \cap f^c)$	$9$ $\sigma(8m)$	$20$ $\mu(23m)$	
Hybrid Reliable Broadcast	$t_1 = t_a + t_s + 1$ $t_2 = \mathbf{n} - t_c - t_a$ $-t_s - t_i$	$\mathbf{n} > t_c \cdot 3t_a + 2t_s + 2t_i$ $ f_x  \leq t_x$ $f_x \cap f_i = \emptyset$ $\text{for } x \neq y$ $x, y \in \{a, c, i, s\}$	$\frac{25}{63}$ $\frac{34}{1877}$	$2$ $37s$	$\mu(35s)$ $\sigma(0.3s)$	$g_2(f^c \cap f_a^c \cap f_s^c \cap f_i^c)$ $\forall x: g_1 \cdot x: f_a \cap f_s \cap f_i \neq \emptyset$ $\forall x: g_2 \cdot g_1 \cdot (x \cap f_a \cap f_i^c)$	$\Delta_{EAGER}^{\text{Reliable Broadcast}}$	$63$ $\forall x: g_2 \cdot g_1 \cdot (x \cap f_a \cap f_i^c)$	$15$ $\sigma(1.5m)$	$45$ $\mu(15m)$	
Byzantine Fast Paxos	$t_1 = \mathbf{n} - t$ $t_2 = \mathbf{n} - q$ $t_3 = \mathbf{n} - 2t - q$ $t_4 = t + 1$	$\mathbf{n} > 2q + 3t$ $t \geq q$ $q \geq 0$ $ b  \leq t$	$\frac{22}{79}$ $\frac{44}{3695}$	$6$ $6s$	T.O.	$g_1(b^c)$ $\forall x: g_2 \cdot g_1 \cdot (x)$ $\forall x: g_1 \cdot y: g_1 \cdot x: f^c \neq \emptyset$ $\forall x: g_1 \cdot y: g_3 \cdot y: g_4 \cdot (x \cap y)$ $\forall x: g_1 \cdot y: g_1 \cdot z: g_2 \cdot g_3 \cdot (x \cap y \cap z)$	$\Delta_{LAZY}^{\text{Protocol}}$ $\forall x: g_3 \cdot x \neq \emptyset$ $\forall x: g_4 \cdot x \neq \emptyset$ $\forall x: g_1 \cdot g_3 \cdot x \cap b^c$ $\forall x: g_1 \cdot y: g_4 \cdot (x \cap y)$	$44$ $\forall x: g_1 \cdot g_3 \cdot x \cap b^c$	$11$ $\sigma(21m)$	$19$ $\mu(36m)$	$2$ $\sigma(28m)$

**Fig. 2.** Protocols verified using our technique. For each protocol,  $T$  is the set of thresholds and  $\Gamma$  is the resilience condition. AIP<sub>AGER</sub> lists metrics for the procedure of finding all  $\Gamma$ -valid TIP formulas (taking time  $\mathbf{t}_C$ ), and verifying the transition system using the resulting properties (taking time  $\mathbf{t}_V$ ). Obtaining a minimal subset that FO-implies the rest takes negligible time, so we did not include it in the table. The properties are given in  $\Delta_{EAGER}^{\text{Protocol}}$ , where  $g_i$  denotes  $g \geq t_i$ . In addition to the run times,  $\mathbf{V}$  shows  $\frac{c}{v}$ , where  $c$  is the number of  $\Gamma$ -valid simple formulas that were checked using the BAPA solver (CVC4), and  $v$  is the total number of  $\Gamma$ -valid simple formulas. Namely,  $v - c$  simple formulas were inferred to be valid via subsumption.  $\mathbf{I}$  reports the analogous metric for  $\Gamma$ -invalid simple formulas. Finally,  $\mathbf{Q}$  reports the maximal number of quantifiers considered (for which all formulas were  $\Gamma$ -invalid). AIP<sub>LAZY</sub> lists metrics for the procedure of finding a set of  $\Gamma$ -valid TIP formulas on counterexamples. The resulting set is listed in  $\Delta_{LAZY}^{\text{Protocol}}$ , and  $\mathbf{t}_I$  lists the total Ivy runtime, with the standard deviation specified below.  $\mathbf{V}$  (resp.  $\mathbf{I}$ ) lists the number of  $\Gamma$ -valid (resp.  $\Gamma$ -invalid) simple formulas considered before the final set was reached.  $\mathbf{CTI}$  lists the number of counterexample iterations required, and  $\mathbf{Q}$  lists the maximal number of quantifiers of any TIP formula considered. Finally,  $\mathbf{t}_V$  lists the time required to verify the first-order transition system assuming the obtained set of properties. T.O. indicates that a time out of 1h was reached.

deviation ( $\sigma$ ). The figure's caption explains the presented information, and we discuss the results below.

**Aip<sub>EAGER</sub>** For all protocols, running AIP took less than 1 min (column  $\mathbf{t_C}$ ), and generated all  $\Gamma$ -valid simple TIP formulas. We observe that for most formulas, (in)validity is deduced from other formulas by subsumption, and less than 2%–5% of the formulas are actually checked using a BAPA query. With the optimization of the redundancy check, minimization of the set is performed in negligible time. The resulting set,  $\Delta_{\text{EAGER}}$ , contains 3–5 formulas, compared to 39–79 before minimization.

Due to the optimization described in Sect. 4 for the BAPA validity queries, the number of quantifiers in the TIP formulas that are checked by AIP does not affect the time needed to compute the full  $\Delta$ . For example, Bosco under the Strongly One-step resilience condition contains  $\Gamma$ -valid simple TIP formulas with up to 7 quantifiers (as  $\mathbf{n} > 7\mathbf{t}$  and  $t_1 = \mathbf{n} - \mathbf{t}$ ), but AIP does not take significantly longer to find  $\Delta$ . Interestingly, in this example the  $\Gamma$ -valid TIP formulas with more than 3 quantifiers are implied (in FOL) by formulas with at most 3 quantifiers, as indicated by the fact that these are the only formulas that remain in  $\Delta_{\text{EAGER}}^{\text{Bosco Strongly One-step}}$ .

**Aip<sub>LAZY</sub>** With the lazy approach based on CTIs, the time for finding the set of TIP formulas,  $\Delta_{\text{LAZY}}$ , is generally longer. This is because the run time is dominated by calls to Ivy with FO axioms that are too weak for verifying the protocol. However, the resulting  $\Delta_{\text{LAZY}}$  has a significant benefit: it lets Ivy prove the protocol much faster compared to using  $\Delta_{\text{EAGER}}$ . Comparing  $\mathbf{t_V}$  in Aip<sub>EAGER</sub> vs. Aip<sub>LAZY</sub> shows that when the former takes a minute, the latter takes a few seconds, and when the former times out after 1 h, the latter terminates, usually in under 1 min. Comparing the formulas of  $\Delta_{\text{EAGER}}$  and  $\Delta_{\text{LAZY}}$  reveals the reason. While the FO translation of both yields EPR formulas, the formulas resulting from  $\Delta_{\text{EAGER}}$  contain more quantifiers and generate much more ground terms, which degrades the performance of Z3.

Another advantage of the lazy approach is that during the search, it avoids considering formulas with many quantifiers unless those are actually needed. Comparing the 3 versions of Bosco we see that Aip<sub>LAZY</sub> is not sensitive to the largest number of quantifiers that may appear in a  $\Gamma$ -valid simple TIP formula. The downside is that Aip<sub>LAZY</sub> performs many Ivy checks in order to compute the final  $\Delta_{\text{LAZY}}$ . The total duration of finding CTIs varies significantly (as demonstrated under the column  $\mathbf{t_I}$ ), in part because it is very sensitive to the CTIs returned by Ivy, which are in turn affected by the random seed used in the heuristics of the underlying solver.

Finally,  $\Delta_{\text{LAZY}}$  provides more insight into the protocol design, since it presents minimal assumptions that are required for protocol correctness. Thus, it may be useful in designing and understanding protocols.

## 7 Related Work

**Fully Automatic Verification of Threshold-Based Protocols.** Algorithms modeled as Threshold automata (TA) [14] have been studied in [13, 16], and verified using an automated tool ByMC [15]. The tool also automatically synthesizes thresholds as arithmetic expressions [24]. Reachability properties of TAs for more general thresholds are studied in [18]. There have been recent advances in verification of synchronous threshold-based algorithms using TAs [41], and of asynchronous randomized algorithms where TAs support coin tosses and an unbounded number of rounds [4]. Still, this modeling is very restrictive and not as faithful to the pseudo-code as our modeling.

Another approach for full automation is to use sound and incomplete procedures for deduction and invariant search for logics that combine quantifiers and set cardinalities [8, 10]. However, distributed systems of the level of complexity we consider here (e.g., Byzantine Fast Paxos) are beyond the reach of these techniques.

**Verification of Distributed Protocols Using Decidable Logics.** Padon et al. [33] introduced an interactive approach for the safety verification of distributed protocols based on EPR using the Ivy [29] verification tool. Later works extended the approach to more complex protocols [32], their implementations [42], and liveness properties [30, 31]. Those works verified some threshold protocols using ad-hoc first-order modeling and axiomatization of threshold-intersection properties, whereas we develop a systematic methodology. Moreover, the axioms were not mechanically verified, except in [42], where a simple intersection property—intersection of two sets with more than  $\frac{n}{2}$  nodes—requires a proof by induction over  $n$ . The proof relies on a user provided induction hypothesis that is automatically checked using the FAU decidable fragment [9]. This approach requires user ingenuity even for a simple intersection property, and we expect that it would not scale to the more complex properties required for e.g. Bosco or Fast Byzantine Paxos. In contrast, our approach completely automates both verification and inference of threshold-intersection properties required to verify protocol correctness.

Dragoi et al. [6] propose a decidable logic supporting cardinalities, uninterpreted functions, and universal quantifiers for verifying consensus algorithms expressed in the partially synchronous Heard-Of Model. As in this paper, the user is expected to provide an inductive invariant. The PSync framework [7] extends the approach to protocol implementations. Compared to our approach, the approach of Dragoi et al. is less flexible due to the specialized logic used and the restrictions of the Heard-Of Model.

Our approach decomposes verification into EPR and BAPA. Piskac [34] presents a decidable logic that combines BAPA and EPR, with some restrictions. The verification conditions of the protocols we consider are outside the scope of this fragment since they include cardinality constraints in the scope of quantifiers. Furthermore, this logic is not supported by mature solvers. Instead

of looking for a specialized logic per protocol, we rely on a decomposition which allows more flexibility.

Recently, [11] presented an approach for verifying asynchronous algorithms by reduction to synchronous verification. This technique is largely orthogonal and complementary to our approach, which is focused on the challenge of cardinality thresholds.

*Verification using interactive theorem provers.* We are not aware of works based on interactive theorem provers that verified protocols with complex thresholds as we do in this work (although doing so is of course possible). However, many works used interactive theorem provers to verify related protocols, e.g., [12, 27, 36–38, 43] (the most related protocols use either  $\frac{n}{2}$  or  $\frac{2n}{3}$  as the only thresholds, other protocols do not involve any thresholds). The downside of verification using interactive theorem provers is that it requires tremendous human efforts and skills. For example, the Verdi proof of Raft included 50,000 lines of proof in Coq for 500 lines of code [44].

## 8 Conclusion

This paper proposes a new deductive verification approach for threshold-based distributed protocols by decomposing the verification problem into two well-established decidable logics, BAPA and EPR, thus allowing greater flexibility compared to monolithic approaches based on domain-specific, specialized logics. The user models their protocol in EPR, defines the thresholds and resilience conditions using arithmetic in BAPA, and provides an inductive invariant. An automatic procedure infers threshold intersection properties expressed in TIP that are both (1) sound w.r.t. the resilience conditions (checked in quantifier-free BAPA) and (2) sufficient to discharge the VCs (checked in EPR). Both logics are supported by mature solvers, and allow providing the user with an understandable counterexample in case verification fails.

Our evaluation, which includes notoriously tricky fast-learning consensus protocols, shows that threshold intersection properties are inferred in a matter of minutes. While this may be too slow for interactive use, we expect improvements such as memoization and parallelism to provide response times of a few seconds in an iterative, interactive setting. Another potential future direction is combining our inference algorithm with automated invariant inference algorithms.

**Acknowledgements.** We thank the anonymous referees for insightful comments which improved this paper. This publication is part of a project that has received funding from the European Research Council (ERC) under the European Union’s Horizon 2020 research and innovation programme (grant agreement No [759102-SViS] and [787367-PaVeS]). The research was partially supported by Len Blavatnik and the Blavatnik Family foundation, the Blavatnik Interdisciplinary Cyber Research Center, Tel Aviv University, the Israel Science Foundation (ISF) under grant No. 1810/18, the United States-Israel Binational Science Foundation (BSF) grant No. 2016260 and the Austrian Science Fund (FWF) through Doctoral College LogiCS (W1255-N23).

## References

1. Abraham, I., Gueta, G., Malkhi, D., Alvisi, L., Kotla, R., Martin, J.P.: Revisiting Fast Practical Byzantine Fault Tolerance (2017)
2. Bansal, K., Reynolds, A., Barrett, C., Tinelli, C.: A new decision procedure for finite sets and cardinality constraints in SMT. In: Olivetti, N., Tiwari, A. (eds.) IJCAR 2016. LNCS (LNAI), vol. 9706, pp. 82–98. Springer, Cham (2016). [https://doi.org/10.1007/978-3-319-40229-1\\_7](https://doi.org/10.1007/978-3-319-40229-1_7)
3. Berkovits, I., Lazić, M., Losa, G., Padon, O., Shoham, S.: Verification of threshold-based distributed algorithms by decomposition to decidable logics. CoRR abs/1905.07805 (2019). <http://arxiv.org/abs/1905.07805>
4. Bertrand, N., Konnov, I., Lazic, M., Widder, J.: Verification of Randomized Distributed Algorithms under Round-Rigid Adversaries. HAL hal-01925533, November 2018. <https://hal.inria.fr/hal-01925533>
5. de Moura, L., Bjørner, N.: Z3: an efficient SMT solver. In: Ramakrishnan, C.R., Rehof, J. (eds.) TACAS 2008. LNCS, vol. 4963, pp. 337–340. Springer, Heidelberg (2008). [https://doi.org/10.1007/978-3-540-78800-3\\_24](https://doi.org/10.1007/978-3-540-78800-3_24)
6. Drăgoi, C., Henzinger, T.A., Veith, H., Widder, J., Zufferey, D.: A logic-based framework for verifying consensus algorithms. In: McMillan, K.L., Rival, X. (eds.) VMCAI 2014. LNCS, vol. 8318, pp. 161–181. Springer, Heidelberg (2014). [https://doi.org/10.1007/978-3-642-54013-4\\_10](https://doi.org/10.1007/978-3-642-54013-4_10)
7. Drăgoi, C., Henzinger, T.A., Zufferey, D.: PSync: A partially synchronous language for fault-tolerant distributed algorithms. In: Proceedings of the 43rd Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL 2016, St. Petersburg, FL, USA, January 20–22, 2016, vol. 51, no. 1, pp. 400–415 (2016). <https://dblp.uni-trier.de/rec/bibtex/conf/popl/DragoiHZ16?q=speculative%20AQ4%20Byzantine%20fault%20tolerance>
8. Dutertre, B., Jovanović, D., Navas, J.A.: Verification of fault-tolerant protocols with sally. In: Dutle, A., Muñoz, C., Narkawicz, A. (eds.) NFM 2018. LNCS, vol. 10811, pp. 113–120. Springer, Cham (2018). [https://doi.org/10.1007/978-3-319-77935-5\\_8](https://doi.org/10.1007/978-3-319-77935-5_8)
9. Ge, Y., de Moura, L.: Complete instantiation for quantified formulas in satisfiability modulo theories. In: Bouajjani, A., Maler, O. (eds.) CAV 2009. LNCS, vol. 5643, pp. 306–320. Springer, Heidelberg (2009). [https://doi.org/10.1007/978-3-642-02658-4\\_25](https://doi.org/10.1007/978-3-642-02658-4_25)
10. v. Gleissenthall, K., Bjørner, N., Rybalchenko, A.: Cardinalities and universal quantifiers for verifying parameterized systems. In: Proceedings of the 37th ACM SIGPLAN Conference on Programming Language Design and Implementation, PLDI 2016, pp. 599–613. ACM (2016)
11. von Gleissenthall, K., Kici, R.G., Bakst, A., Stefan, D., Jhala, R.: Pretend synchrony: synchronous verification of asynchronous distributed programs. PACMPL 3(POPL), 59:1–59:30 (2019). <https://dl.acm.org/citation.cfm?id=3290372>
12. Hawblitzel, C., Howell, J., Kapritsos, M., Lorch, J.R., Parno, B., Roberts, M.L., Setty, S.T.V., Zill, B.: Ironfleet: proving practical distributed systems correct. In: Proceedings of the 25th Symposium on Operating Systems Principles, SOSP 2015, Monterey, CA, USA, 4–7 October 2015, pp. 1–17 (2015). <https://doi.org/10.1145/2815400.2815428>

13. Konnov, I., Lazic, M., Veith, H., Widder, J.: Para<sup>2</sup>: Parameterized path reduction, acceleration, and SMT for reachability in threshold-guarded distributed algorithms. *Form. Methods Syst. Des.* **51**(2), 270–307 (2017). <https://link.springer.com/article/10.1007/s10703-017-0297-4>
14. Konnov, I., Veith, H., Widder, J.: On the completeness of bounded model checking for threshold-based distributed algorithms: reachability. *Inf. Comput.* **252**, 95–109 (2017)
15. Konnov, I., Widder, J.: ByMC: byzantine model checker. In: Margaria, T., Steffen, B. (eds.) ISoLA 2018. LNCS, vol. 11246, pp. 327–342. Springer, Cham (2018). [https://doi.org/10.1007/978-3-030-03424-5\\_22](https://doi.org/10.1007/978-3-030-03424-5_22)
16. Konnov, I.V., Lazic, M., Veith, H., Widder, J.: A short counterexample property for safety and liveness verification of fault-tolerant distributed algorithms. In: Proceedings of the 44th ACM SIGPLAN Symposium on Principles of Programming Languages, POPL 2017, Paris, France, 18–20 January 2017, pp. 719–734 (2017)
17. Kotla, R., Alvisi, L., Dahlin, M., Clement, A., Wong, E.: Zyzzyva: speculative Byzantine fault tolerance. *SIGOPS Oper. Syst. Rev.* **41**(6), 45–58 (2007)
18. Kukovec, J., Konnov, I., Widder, J.: Reachability in parameterized systems: all flavors of threshold automata. In: CONCUR. LIPIcs, vol. 118, pp. 19:1–19:17. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik (2018)
19. Kuncak, V., Nguyen, H.H., Rinard, M.: An algorithm for deciding BAPA: boolean algebra with presburger arithmetic. In: Nieuwenhuis, R. (ed.) CADE 2005. LNCS (LNAI), vol. 3632, pp. 260–277. Springer, Heidelberg (2005). [https://doi.org/10.1007/11532231\\_20](https://doi.org/10.1007/11532231_20)
20. Lamport, L.: The Part-time Parliament 16(2), 133–169 (1998–2005). <https://doi.org/10.1145/279227.279229>
21. Lamport, L.: Lower bounds for asynchronous consensus. In: Schiper, A., Shvartsman, A.A., Weatherspoon, H., Zhao, B.Y. (eds.) Future Directions in Distributed Computing. LNCS, vol. 2584, pp. 22–23. Springer, Heidelberg (2003). [https://doi.org/10.1007/3-540-37795-6\\_4](https://doi.org/10.1007/3-540-37795-6_4)
22. Lamport, L.: Lower bounds for asynchronous consensus. *Distrib. Comput.* **19**(2), 104–125 (2006)
23. Lamport, L.: Fast byzantine paxos, 17 November 2009. uS Patent 7,620,680
24. Lazic, M., Konnov, I., Widder, J., Bloem, R.: Synthesis of distributed algorithms with parameterized threshold guards. In: OPODIS (2017, to appear). <http://forsyte.at/wp-content/uploads/opodis17.pdf>
25. Lewis, H.R.: Complexity results for classes of quantificational formulas. *Comput. Syst. Sci.* **21**(3), 317–353 (1980)
26. Lifflton, M.H., Previti, A., Malik, A., Marques-Silva, J.: Fast, flexible nus enumeration. *Constraints* **21**(2), 223–250 (2016)
27. Liu, Y.A., Stoller, S.D., Lin, B.: From clarity to efficiency for distributed algorithms. *ACM Trans. Program. Lang. Syst.* **39**(3), 121–1241 (2017). <https://doi.org/10.1145/2994595>
28. Martin, J.P., Alvisi, L.: Fast Byzantine consensus. *IEEE Trans. Dependable Secure Comput.* **3**(3), 202–215 (2006)
29. McMillan, K.L., Padon, O.: Deductive verification in decidable fragments with ivy. In: Podelski, A. (ed.) SAS 2018. LNCS, vol. 11002, pp. 43–55. Springer, Cham (2018). [https://doi.org/10.1007/978-3-319-99725-4\\_4](https://doi.org/10.1007/978-3-319-99725-4_4)
30. Padon, O., Hoenicke, J., Losa, G., Podelski, A., Sagiv, M., Shoham, S.: Reducing liveness to safety in first-order logic. *PACMPL* **2**(POPL), 26:1–26:33 (2018)

31. Padon, O., Hoenicke, J., McMillan, K.L., Podelski, A., Sagiv, M., Shoham, S.: Temporal prophecy for proving temporal properties of infinite-state systems. In: FMCAD, pp. 1–11. IEEE (2018)
32. Padon, O., Losa, G., Sagiv, M., Shoham, S.: Paxos made EPR: decidable reasoning about distributed protocols. PACMPL **1**(OOPSLA), 1081–10831 (2017)
33. Padon, O., McMillan, K.L., Panda, A., Sagiv, M., Shoham, S.: Ivy: safety verification by interactive generalization. In: Krantz, C., Berger, E. (eds.) Proceedings of the 37th ACM SIGPLAN Conference on Programming Language Design and Implementation, PLDI 2016, Santa Barbara, CA, USA, 13–17 June 2016, pp. 614–630. ACM (2016)
34. Piskac, R.: Decision procedures for program synthesis and verification (2011). <http://infoscience.epfl.ch/record/168994>
35. Piskac, R., de Moura, L., Bjørner, N.: Deciding effectively propositional logic using DPLL and substitution sets. J. Autom. Reason. **44**(4), 401–424 (2010)
36. Rahli, V., Guaspari, D., Bickford, M., Constable, R.L.: Formal specification, verification, and implementation of fault-tolerant systems using eventml. ECEASST **72** (2015). <https://doi.org/10.14279/tuj.eceasst.72.1013>
37. Rahli, V., Vukotic, I., Völp, M., Esteves-Verissimo, P.: Velisarios: Byzantine fault-tolerant protocols powered by Coq. In: Ahmed, A. (ed.) ESOP 2018. LNCS, vol. 10801, pp. 619–650. Springer, Cham (2018). [https://doi.org/10.1007/978-3-319-89884-1\\_22](https://doi.org/10.1007/978-3-319-89884-1_22)
38. Sergey, I., Wilcox, J.R., Tatlock, Z.: Programming and proving with distributed protocols. PACMPL **2**(POPL), 28:1–28:30 (2018)
39. Song, Y.J., van Renesse, R.: Bosco: one-step Byzantine asynchronous consensus. In: Taubenfeld, G. (ed.) DISC 2008. LNCS, vol. 5218, pp. 438–450. Springer, Heidelberg (2008). [https://doi.org/10.1007/978-3-540-87779-0\\_30](https://doi.org/10.1007/978-3-540-87779-0_30)
40. Srikanth, T., Toueg, S.: Simulating authenticated broadcasts to derive simple fault-tolerant algorithms. Dist. Comp. **2**, 80–94 (1987)
41. Stoilkovska, I., Konnov, I., Widder, J., Zuleger, F.: Verifying safety of synchronous fault-tolerant algorithms by bounded model checking. In: Vojnar, T., Zhang, L. (eds.) TACAS 2019. LNCS, vol. 11428, pp. 357–374. Springer, Cham (2019). [https://doi.org/10.1007/978-3-030-17465-1\\_20](https://doi.org/10.1007/978-3-030-17465-1_20)
42. Taube, M., et al.: Modularity for decidability of deductive verification with applications to distributed systems. In: PLDI, pp. 662–677. ACM (2018)
43. Wilcox, J.R., et al.: Verdi: a framework for implementing and formally verifying distributed systems. In: Proceedings of the 36th ACM SIGPLAN Conference on Programming Language Design and Implementation, Portland, OR, USA, 15–17 June 2015, pp. 357–368 (2015). <https://doi.org/10.1145/2737924.2737958>
44. Woos, D., Wilcox, J.R., Anton, S., Tatlock, Z., Ernst, M.D., Anderson, T.E.: Planning for change in a formal verification of the raft consensus protocol. In: Proceedings of the 5th ACM SIGPLAN Conference on Certified Programs and Proofs, Saint Petersburg, FL, USA, 20–22 January 2016, pp. 154–165 (2016). <https://doi.org/10.1145/2854065.2854081>

**Open Access** This chapter is licensed under the terms of the Creative Commons Attribution 4.0 International License (<http://creativecommons.org/licenses/by/4.0/>), which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons license and indicate if changes were made.

The images or other third party material in this chapter are included in the chapter's Creative Commons license, unless indicated otherwise in a credit line to the material. If material is not included in the chapter's Creative Commons license and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder.

