The Raymond and Beverly Sackler Faculty of Exact Sciences
Tel Aviv University

The Blavatnik School of Computer Science

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

by

Idan Berkovits

under the supervision of
Prof. Sharon Shoham

Thesis submitted in partial fulfillment of the requirements
for the degree of Master of Science

2020

Abstract

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

Idan Berkovits
Master of Science
School of Computer Science
Tel Aviv University

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.

Acknowledgements

First and foremost, I would like to thank my supervisor, Prof. Sharon Shoham, for her wonderful and close guidance, for teaching me to think creatively and out of the box while insisting on the small details and thorough work. Without her endless patience, this work would not have come to be.

To Oded Padon for the insightful discussions and wonderful ideas. To Marijana Lazić and Giuliano Losa for their good collaboration.

To my parents, Neomi and Moshe, for their enormous support and slight nagging along the way. To my friends and study partners who provided relief, to Omri Lifshitz for nights of shared despair. Together on stage.

Contents

Chapter 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 thesis we develop a methodology for deductive verification of *threshold-based* distributed protocols using decidable logic, where we use *decomposition* into two 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 $\mathbf{n}/2$ processes, where $\mathbf{n}$ is the total number of processes. Such expressions as $\mathbf{n}/2$, 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 $\mathbf{t}$ process crashes, and ensures that every two processes that decide agree on the decided value. The protocol requires $\mathbf{n} > 2\mathbf{t}$ processes, and each process must

obtain confirmation messages from $\mathbf{n} - \mathbf{t}$ processes before making a decision. The protocol is correct due to, among others, the fact that if $\mathbf{n} > 2\mathbf{t}$ then any two sets of $\mathbf{n} - \mathbf{t}$ 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 $\mathbf{n}-\mathbf{t}$ 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 $\mathbf{n}-\mathbf{t}$ 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 $\mathbf{n} - \mathbf{t}$ have a non-empty intersection. This property can be encoded in FOL by modeling sets of cardinality at least $\mathbf{n} - \mathbf{t}$ using an uninterpreted sort along with a corresponding membership relation between this sort and the sort for processes. However, the validity of the intersection property under the assumption that $\mathbf{n} > 2\mathbf{t}$ cannot be verified in FOL.

The key idea of this thesis 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 automated 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 thesis are:

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

5. 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.

Chapter 2

Preliminaries

In this chapter we present the necessary background on the formalization of transition systems using FOL, as well as on the decidable logics used in our work: EPR and BAPA.

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 Σ that consists of sorted constant, function and relation symbols, s.t. s satisfies a finite set of *axioms* Θ in the form of closed formulas over Σ. $\mathcal{D}$ 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 (Σ, Θ, I, TR), where Σ and Θ are as above, I is the *initial condition* given by a closed formula over Σ, and TR is the *transition relation* given by a closed formula over $\Sigma \uplus \Sigma'$ where Σ describes the source state of the transition and $\Sigma' = \{a' \mid a \in \Sigma\}$ describes the target state. We require that TR does not modify any symbol that appears in Θ. The set of initial states and the set of transitions of the system consist of the states, respectively, pairs of states, that satisfy I, respectively, TR. 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 Σ. The system is *safe* if all of its reachable states satisfy P. A closed FO formula Inv over Σ is an *inductive invariant* for a transition system (Σ, Θ, 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. Requirements (i) and (ii) ensure that Inv represents a superset of the reachable states, hence together with (iii) safety follows. 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*, meaning that a satisfiable formula is guaranteed to have a finite model. The size of this model is bounded by the total number of existential quantifiers and constants in the formula. While EPR does not allow any function symbols nor quantifier alternation except $\exists^*\forall^*$, we consider a straightforward extension of

EPR that maintains these properties and is supported by mature 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 φ 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:

- **Function edges:** let f be a function in φ from sorts $s_1, \ldots, s_k$ to sort s. Then there is a $\forall\exists$ edge from s_i to s for every $1 \leq i \leq k$.
- **Quantifier edges:** let $\exists x : s$ be an existential quantifier that resides in the scope of the universal quantifiers $\forall x_1 : s_1, \ldots, \forall x_k : s_k$ in φ. Then there is a $\forall\exists$ edge from s_i to s for every $1 \leq i \leq k$.

The quantifier alternation graph is extended to sets of formulas as expected.

Presburger Arithmetic (PA) Presburger Arithmetic (PA) [4] is the FO theory of the integers with addition. It is defined over sort int, representing integers. The language of PA is defined by the following grammar:

$$\begin{aligned} F ::=& L_1 = L_2 \mid L_1 < L_2 \mid F_1 \wedge F_2 \mid F_1 \vee F_2 \mid \neg F \mid \exists u.F \mid \forall u.F \\ L ::=& u \mid K \mid i \mid L_1 + L_2 \mid K \cdot L \\ K ::=& \ldots -2 \mid -1 \mid 0 \mid 1 \mid 2 \ldots \end{aligned}$$

where L defines linear integer terms, in which u denotes an integer variable, $k \in K$ defines an (interpreted) integer constant symbol $\ldots, -2, -1, 0, 1, 2 \ldots$, and i is an uninterpreted integer constant symbol (as opposed to the constant symbols from K); and F defines the set of PA formulas, where $\ell_1 = \ell_2$ and $\ell_1 < \ell_2$ are atomic formulas. Note that PA formulas are closed under Boolean operations and allow quantification over integer variables (u). In the sequel, we also allow arithmetic terms of the form $\frac{\ell}{k}$ where $k \in K$ is a positive integer and $\ell \in L$, with the intention that a literal that contains such terms represents the PA literal obtained by multiplying by k (not to be confused with the divisibility predicates and operations that are considered in some extensions of the language).

A (standard) PA structure $s_P = (\mathcal{D}, \mathcal{I})$ consists of a domain $\mathcal{D}$ mapping sort int to the set of all integers and an interpretation function $\mathcal{I}$ of the symbols. The semantics of terms and formulas is as expected. The uninterpreted constant symbols may be interpreted arbitrarily.

Both validity and satisfiability of PA formulas (with arbitrary quantification) are decidable [19]. Decidability of PA formula is supported by mature SMT solvers such as Z3 [5] and CVC4 [2].

Boolean Algebra with Presburger Arithmetic (BAPA) Boolean Algebra with Presburger Arithmetic (BAPA) [19] is a FO theory that combines PA with the theory of Boolean algebra of sets of uninterpreted elements. It is defined over two sorts: int, representing integers, and set, representing subsets of a finite universe. The language is defined by the following grammar:

$$\begin{aligned} F ::=& B_1 = B_2 \mid L_1 = L_2 \mid L_1 < L_2 \mid F_1 \wedge F_2 \mid F_1 \vee F_2 \mid \neg F \mid \exists x.F \mid \forall x.F \mid \exists u.F \mid \forall u.F \\ B ::=& x \mid \emptyset \mid \mathbf{a} \mid B_1 \cup B_2 \mid B_1 \cap B_2 \mid B^c \\ L ::=& u \mid K \mid \mathbf{n} \mid i \mid L_1 + L_2 \mid K \cdot L \mid |B| \\ K ::=& \ldots -2 \mid -1 \mid 0 \mid 1 \mid 2 \ldots \end{aligned}$$

where K and L are defined as in PA, with the addition of $\mathbf{n}$ as an integer constant symbol that represents the size of the finite set universe, 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). BAPA formulas are closed under Boolean operations and allow quantification over integer variables (u), and over set variables (x). Similar to PA, we also allow arithmetic terms of the form $\frac{\ell}{k}$ where $k \in K$ is a positive integer and $\ell \in L$.

A BAPA structure $s_B = (\mathcal{D}, \mathcal{I})$ consists of a domain $\mathcal{D}$ mapping sort int to the set of all integers and mapping the sort set to the set of all subsets of a finite universe U, called the *universal set*, and an interpretation function $\mathcal{I}$ of the symbols for integer and set operations. The semantics of terms and formulas is as expected. The interpretation of the complement operation is defined with respect to U. In particular, this means that $\mathcal{I}(\emptyset^c) = U$. The integer constant $\mathbf{n}$ is interpreted to the size of U, i.e. $\mathcal{I}(\mathbf{n}) = |U|$. The uninterpreted constant symbols may be interpreted arbitrarily.

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

Chapter 3

First-Order Modeling of Threshold-Based Protocols

In this chapter 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 Chapter 4.

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. 3.1.

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 $\mathbf{n} > 3\mathbf{t}$; it guarantees that a preliminary decision will be reached if all nodes are non-faulty and propose the same value when $\mathbf{n} > 5\mathbf{t}$ (weakly one-step condition), and even if some nodes are faulty, as long as all non-faulty nodes propose the same value, when $\mathbf{n} > 7\mathbf{t}$ (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 $\frac{\mathbf{n}+3\mathbf{t}}{2}$ nodes proposed the same value. When $\mathbf{n} > 3\mathbf{t}$, two sets of cardinality greater than $\frac{\mathbf{n}+3\mathbf{t}}{2}$ have at least one non-faulty node in common, and therefore no two different values can be proposed by more than $\frac{\mathbf{n}+3\mathbf{t}}{2}$ nodes. Similarly, we can derive property (b) from the fact that a set of more than $\frac{\mathbf{n}+3\mathbf{t}}{2}$ nodes and a set of $\mathbf{n}-\mathbf{t}$ nodes intersect in $\frac{\mathbf{n}+\mathbf{t}}{2}$ nodes, which, after removing t nodes which may be faulty, still leaves us with more than $\frac{\mathbf{n}-\mathbf{t}}{2}$ nodes, satisfying the condition in line 9.

```
1  Input: v_p
2  broadcast v_p to all processes
3  wait until n − t messages have been received
4
5  if there exists v s.t. more than (n+3t)/2
6      messages contain value v then
7    DECIDE(v)
8  if there exists exactly one v s.t. more than
9        (n−t)/2 messages contain value v then
10   v_p := v
11 call underlying−consensus(v_p)
```

```
1  sort node, value, set_{n−t}, set_{(n+3t+1)/2}, set_{(n−t+1)/2}
2  ...
3  assume ∃q : set_{n−t}. ∀m : node. member(m, q) →
4                ∃u : value. rcv_msg(n, m, u)
5  if ∃v : value, q : set_{(n+3t+1)/2}. ∀m : node.
6            member(m, q) → rcv_msg(n, m, v) then
7    decision(n, v) := true
8  if ∃!v : value. ∃q : set_{(n−t+1)/2}. ∀m : node.
9            member(m, q) → rcv_msg(n, m, v) then
10   v_p := v
11 und_cons(n, v_p) := true
```

Figure 3.1: 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 $(\exists x.\,\varphi(x)) \wedge (\forall x, y.\,\varphi(x) \wedge \varphi(y) \rightarrow x = y)$.

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 $\mathbf{n}$, used to represent the total number of nodes (assumed to be finite). Protocol correctness is ensured under a set of assumptions Γ called *resilience conditions*, formulated as BAPA formulas over *Prm* (this means that all the uninterpreted constants appearing in Γ 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 ℓ is a ground BAPA integer term over *Prm* (we restrict k to ensure that the guard can be translated to BAPA). Comparing sizes of two sets is not allowed – we observe that it is not needed for threshold-based protocols. We denote the set of thresholds by T. For example, in Bosco, $T = \{\mathbf{n} - \mathbf{t}, \frac{\mathbf{n}+3\mathbf{t}+1}{2}, \frac{\mathbf{n}-\mathbf{t}+1}{2}\}$.

Without loss of generality, we assume that all conditions on set cardinalities are of the form "at least t". This is because every condition can be written in this form, possibly by introducing new threshold expressions and complementing the set which the condition refers to, according to the following rules:

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

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 Σ_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 Σ_T^{Prm}:

- For every threshold t we introduce a *threshold sort* $\mathsf{set_t}$ with the intended meaning that elements of this sort are sets of nodes whose size is at least t.
- As the threshold sorts represent *sets*, each sort $\mathsf{set_t}$ is equipped with a binary relation symbol

$member_t$ between sort node and sort $\mathsf{set_t}$ that captures the membership relation of a node (first argument) in a set (second argument).

- 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 (Σ, Θ, I, TR) where $\Sigma = \Sigma_C \uplus \Sigma_T^{Prm}$.

We are interested only in states (FO structures over Σ) where the interpretation of the threshold sorts and membership relations is according to their intended meaning in a corresponding BAPA structure. The intended meaning is captured by a BAPA structure (over Prm) that satisfies the resilience condition, and whose universal set coincides with the set of nodes. 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 Σ_C and a BAPA structure $s_B = (\mathcal{D}_B, \mathcal{I}_B)$ over Prm are* compatible *if $\mathcal{D}_B(\mathsf{set}) = \mathcal{P}(\mathcal{D}_C(\mathsf{node}))$, 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 Σ defined as follows:*

$$\mathcal{D}(\mathsf{s}) = \mathcal{D}_C(\mathsf{s}) \text{ for every sort } \mathsf{s} \text{ in } \Sigma_C$$
$$\mathcal{D}(\mathsf{set_t}) = \{A \subseteq \mathcal{D}_C(\mathsf{node}) \mid |A| \geq \mathcal{I}_B(t)\} \text{ for every } t \text{ in } T$$
$$\mathcal{I}(a) = \mathcal{I}_C(a) \text{ for every uninterpreted symbol } a \text{ in } \Sigma_C$$
$$\mathcal{I}(member_{\mathbf{a}}) = \mathcal{I}_B(\mathbf{a}) \text{ for every } \mathbf{a} \text{ in } Prm_S$$
$$\mathcal{I}(member_t) = \{(e, A) \mid e \in \mathcal{D}_C(\mathsf{node}), A \in \mathcal{D}(\mathsf{set_t}), e \in A\} \text{ for every } t \text{ in } T$$

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

Definition 2 (Feasibility)**.** *T is* Γ-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 the condition that $|\{n : \mathsf{node} \mid \varphi(n, \bar{u})\}| \geq t$, i.e., that there are at least t nodes that satisfy the FO formula φ over Σ_C (where $\bar{u}$ are additional free variables in φ), we use the following first-order formula over Σ: $\exists q : \mathsf{set_t}.\ \forall n : \mathsf{node}.\ member_t(n, q) \rightarrow \varphi(n, \bar{u})$, where $\mathsf{set_t}$ is the threshold sort assigned to t. 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 $member_{\mathbf{a}}(n)$. For example, in Fig. 3.1, 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 Σ_C, $s_B = (\mathcal{D}_B, \mathcal{I}_B)$ a compatible BAPA structure over Prm s.t. $s_B \models \Gamma$ and T a Γ-feasible set of thresholds over Prm. Then there exists a (unique) T-extension s of s_C by s_B. Further:*

1. *For every $\mathbf{a} \in Prm_S$ and FO valuation ι:*

$$s, \iota \models member_{\mathbf{a}}(n) \quad \textit{iff} \quad \iota(n) \in \mathcal{I}_B(\mathbf{a}),$$

2. *For every $t \in T$, formula φ over Σ_C and FO valuation ι:*

$$s, \iota \models \exists q : \mathsf{set_t}.\ \forall n : \mathsf{node}.\ \mathit{member}_t(n, q) \to \varphi(n, \bar{u})$$
$$\textit{iff} \quad |\{e \in \mathcal{D}(\mathsf{node}) \mid s_C, \iota[n \mapsto e] \models \varphi(n, \bar{u})\}| \geq \mathcal{I}_B(t).$$

Proof. As T is Γ-feasible, Def. 1 well defines a unique structure $s = (\mathcal{D}, \mathcal{I})$ that is the T-extension of s_C by s_B. By Def. 1:

1. Let $\mathbf{a} \in \mathit{Prm}_S$ and ι be a FO valuation. Then, $s, \iota \models \mathit{member}_{\mathbf{a}}(n)$ iff $\iota(n) \in \mathcal{I}(\mathit{member}_{\mathbf{a}}) = \mathcal{I}_B(\mathbf{a})$.

2. Assume $t \in T$, φ is a formula and ι is a FO valuation. Then, $s, \iota \models \exists q : \mathsf{set_t}.\ \forall n : \mathsf{node}.\ \mathit{member}_t(n, q) \to \varphi(n, \bar{u})$ iff there exists a set $Q \in \mathcal{D}(\mathsf{set_t})$ such that $s, \iota[q \mapsto Q] \models \forall n : \mathsf{node}.\ \mathit{member}_t(n, q) \to \varphi(n, \bar{u})$, which is equivalent to having all elements $e \in \mathcal{D}(\mathsf{node})$ with $s, \iota[q \mapsto Q][n \mapsto e] \models \mathit{member}_t(n, q)$, satisfying $s, \iota[q \mapsto Q][n \mapsto e] \models \varphi(n, \bar{u})$, or in other words all elements $e \in Q \subseteq \mathcal{D}(\mathsf{node})$, are satisfying $s_C, \iota[q \mapsto Q][n \mapsto e] \models \varphi(n, \bar{u})$. Finally, because the variable q is not in φ we get that $s_C, \iota \models \exists q : \mathsf{set_t}.\ \forall n : \mathsf{node}.\ \mathit{member}_t(n, q) \to \varphi(n, \bar{u})$ iff there exists a set $Q \in \mathcal{D}(\mathsf{set_t})$ such that for every $e \in Q$ $s_C, \iota[n \mapsto e] \models \varphi(n, \bar{u})$. This means that $Q \subseteq \{e \in \mathcal{D}(\mathsf{node}) \mid s_C, \iota[n \mapsto e] \models \varphi(n, \bar{u})\}$, and because $|Q| \geq \mathcal{I}_B(t)$ we get the wanted result.

□

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

Incompleteness. Lem. 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, making it an *abstraction* of 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* FO formulas over Σ that hold in all threshold-faithful structures is not recursively enumerable[1], which is where the essential incompleteness of our approach lies.

[1] In a threshold-faithful structure the set of nodes must be finite, so validity of a general formula in all threshold-faithful structures can easily encode validity of FOL over finite structures, which has no complete proof system.

Chapter 4

Decomposition via Threshold Intersection Properties

In this chapter, 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 Γ (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, where $t \in T$ is a threshold (of the form $\frac{\ell}{k}$) and $\mathbf{a} \in Prm_S$:

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

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 ($\ell_1 = \ell_2$ or $\ell_1 < \ell_2$) are allowed. TIP only allows guarded quantifiers over set variables, that is, quantification is restricted to sets of a certain cardinality (we say that the variable x is *guarded* by t). We exclude negation, disjunction and existential quantification in formulas. We restrict comparison atomic formulas to $b \neq \emptyset$ and $b^c = \emptyset$, which correspond to asserting that the cardinality of the set represented by b is *at least* 1, respectively $\mathbf{n}$. Furthermore, 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*

$\mathbf{n-t}$ *nodes and a set of more than* $\frac{\mathbf{n+3t}}{2}$ *nodes consists of at least* $\frac{\mathbf{n-t}}{2}$ *non-faulty nodes. This is needed for establishing correctness of the protocol.*

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

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: Set terms (derived from B) are also set terms of BAPA, and most set formula constructs map to constructs of BAPA directly. The only constructs that are not in BAPA are those involving guards, which correspond to a special case of cardinality constraints. We therefore define the following translation $\mathcal{B}$:

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

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

If Γ is quantifier-free (which is the typical case), Γ-validity of TIP formulas can be checked via validity checks of quantifier-free BAPA formulas, supported by mature solvers. Further improvements can be made, as shown in Sec. 5.3.

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 (Sec. 3.2):

$$\begin{array}{rcl@{\qquad}rcl}
\mathcal{FO}(\neg\varphi) & = & \neg\mathcal{FO}(\varphi) & \mathcal{FO}(n \in b^c) & = & \neg\mathcal{FO}(n \in b) \\
\mathcal{FO}(\varphi_1 \wedge \varphi_2) & = & \mathcal{FO}(\varphi_1) \wedge \mathcal{FO}(\varphi_2) & \mathcal{FO}(n \in \emptyset) & = & \mathit{false} \\
\mathcal{FO}(\forall\, x : g.\, \varphi) & = & \forall\, x : \mathsf{set_g}\,.\mathcal{FO}(\varphi) & \mathcal{FO}(n \in \mathbf{a}) & = & \mathit{member}_{\mathbf{a}}(n) \\
\mathcal{FO}(n \in b_1 \cap b_2) & = & \mathcal{FO}(n \in b_1) \wedge \mathcal{FO}(n \in b_2) & \mathcal{FO}(n \in x) & = & \mathit{member}_t(n, x) \\
\mathcal{FO}(b \neq \emptyset) & = & \exists n : \mathsf{node}.\, \mathcal{FO}(n \in b) & \multicolumn{3}{l}{\text{where } x \text{ is guarded by } t} \\
\mathcal{FO}(b^c = \emptyset) & = & \forall n : \mathsf{node}.\, \mathcal{FO}(n \in b) & & & \\
\mathcal{FO}(g_{\geq t}(b)) & = & \exists x : \mathsf{set_t}.\, \forall n : \mathsf{node}.\, \mathit{member}_t(n, x) \rightarrow \mathcal{FO}(n \in b) & & &
\end{array}$$

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

Soundness of the translation to FOL. Next, we state the soundness of the translation, which intuitively means that $\mathcal{FO}(\varphi)$ is "equivalent" to φ over threshold-faithful FO structures (Def. 1). This justifies adding $\mathcal{FO}(\varphi)$ as a FO axiom whenever φ is Γ-valid.

Theorem 1 (Translation soundness)**.** *Let* $s_C = (\mathcal{D}_C, \mathcal{I}_C)$ *be a first-order structure over* Σ_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* φ*, we have* $s_B \models \varphi \Leftrightarrow s \models \mathcal{FO}(\varphi)$*.*

Proof. We prove the theorem by induction over the structure of TIP formulas. Specifically, we prove that for every TIP formula φ (not necessarily closed), for every valuation ι, we have that $s_B, \iota \models \varphi$ iff $s, \iota \models \mathcal{FO}(\varphi)$. Note that the same valuation is well defined in both structures since s is a T-extension of s_C. Base cases of the induction:

- $\varphi = n \in \emptyset$: $s_B, \iota \models n \in \emptyset$ iff $\iota(n) \in \mathcal{I}_B(\emptyset)$ iff $\iota(n) \in \emptyset$ iff $s, \iota \models \mathit{false}$ iff $s, \iota \models \mathcal{FO}(n \in \emptyset)$.

- $\varphi = n \in b^c$: $s_B, \iota \models n \in b^c$ iff $\iota(n) \in \mathcal{I}_B(b^c)$ iff $\iota(n) \notin \mathcal{I}_B(b)$ iff $s_B, \iota \not\models n \in b$ iff $s, \iota \not\models \mathcal{FO}(n \in b)$ iff $s, \iota \models \neg\mathcal{FO}(n \in b)$ iff $s, \iota \models \mathcal{FO}(n \in b^c)$.

- $\varphi = n \in \mathbf{a}$: $s_B, \iota \models n \in \mathbf{a}$ iff $\iota(n) \in \mathcal{I}_B(\mathbf{a})$ iff $\iota(n) \in \mathcal{I}(member_{\mathbf{a}})$ iff $s, \iota \models member_{\mathbf{a}}(n)$ iff $s, \iota \models \mathcal{FO}(n \in \mathbf{a})$.

- $\varphi = n \in x$: Similar to the previous case.

- $\varphi = b \neq \emptyset$: $s_B, \iota \models b \neq \emptyset$ iff $\iota(b) \neq \mathcal{I}_B(\emptyset)$ iff $\iota(b) \neq \emptyset$ iff there exists an element $e \in \mathcal{D}(\mathsf{node})$ such that $e \in \iota(b)$ iff $\exists e \in \mathcal{D}(\mathsf{node})$ such that $s_B, \iota[n \mapsto e] \models n \in b$ iff $\exists e \in \mathcal{D}(\mathsf{node})$ such that $s, \iota[n \mapsto e] \models \mathcal{FO}(n \in b)$ iff $s, \iota \models \exists n : \mathsf{node}.\ \mathcal{FO}(n \in b)$ iff $s, \iota \models \mathcal{FO}(b \neq \emptyset)$.

- $\varphi = b^c = \emptyset$: $s_B, \iota \models b^c = \emptyset$ iff $\iota(b^c) = \mathcal{I}_B(\emptyset)$ iff $\iota(b^c) = \emptyset$ iff $\iota(b) = \emptyset^c$ iff every element $e \in \mathcal{D}(\mathsf{node})$ holds that $e \in \iota(b)$ iff $\forall e \in \mathcal{D}(\mathsf{node})$ it holds that $s_B, \iota[n \mapsto e] \models n \in b$ iff $\forall e \in \mathcal{D}(\mathsf{node})$ it holds that $s, \iota[n \mapsto e] \models \mathcal{FO}(n \in b)$ iff $s, \iota \models \forall n : \mathsf{node}.\ \mathcal{FO}(n \in b)$ iff $s, \iota \models \mathcal{FO}(b^c = \emptyset)$.

- $\varphi = g_{\geq t}(b)$: According to Lem. 1, $s_B, \iota \models g_{\geq t}(b)$ iff $s_B, \iota \models |b| \geq t$ iff $|\iota(b)| \geq \mathcal{I}_B(t)$ iff
$$|\{e \in \mathcal{D}(snode) \mid e \in \iota(b)\}| \geq \mathcal{I}_B(t)$$
iff
$$|\{e \in \mathcal{D}(snode) \mid s, \iota[n \mapsto e] \models n \in b\}| \geq \mathcal{I}_B(t)$$
iff
$$|\{e \in \mathcal{D}(snode) \mid s_C, \iota[n \mapsto e] \models \mathcal{FO}(n \in b)\}| \geq \mathcal{I}_B(t)$$
iff
$$s, \iota \models \exists x : \mathsf{set_t}.\ \forall n : \mathsf{node}.\ member_t(n, x) \rightarrow \mathcal{FO}(n \in b)$$
iff $s, \iota \models \mathcal{FO}(g_{\geq t}(b))$ as required.

To complete the proof we need to prove the induction steps. For formulas formed by negation and conjunction, the proof is trivial. We are left with the case of $\varphi = \forall x : g.\, \varphi_1$. Assuming $s_B, \iota' \models \varphi_1 \Leftrightarrow s, \iota' \models \mathcal{FO}(\varphi_1)$ for every valuation ι', it can be easily shown that the statement holds for φ, using set reasoning and the correspondence between the elements (sets) that satisfy $\mathcal{B}(g(x))$ in s_B and the elements in $\mathcal{D}(\mathsf{set_g})$. □

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

This justifies using the translation to FOL in order to generate first-order axioms from TIP formulas φ that are entailed by the resilience conditions. Namely, if $\Gamma \models \varphi$, then $\mathcal{FO}(\varphi)$ may be safely added as a first-order axiom.

Chapter 5

From Infinitely Many TIP Formulas to Finitely Many EPR Formulas

To apply the approach described in Chapters 3 and 4 for verifying threshold-based protocols, it is crucial to find suitable threshold properties for a given protocol. That is, given the resilience conditions Γ and a FO transition system modeling the protocol, we need to find a set Δ 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 chapter we lay the foundation for automatically inferring such a set Δ. We address the problem of doing so *efficiently* in Chapter 6. In particular, we prove that for any protocol that satisfies a natural condition, there are finitely many Γ-valid TIP formulas (up to equivalence), and that it is easy to check their validity using BAPA entailment, enabling a complete automatic inference algorithm. Furthermore, we show that (under certain reasonable conditions formalized in this chapter) the FO axioms resulting from the inferred TIP properties have an *acyclic* quantifier alternation graph, facilitating protocol verification in EPR. Finally, we suggest two approaches for choosing a *subset* of the Γ-valid TIP formulas in order to simplify the verification conditions and avoid divergence of the EPR solver as it attempts to discharge them.

Notation. For the rest of this chapter, we fix a set Prm of parameters, a set Γ of resilience conditions over Prm, and a set T of thresholds. Note that $b \neq \emptyset \equiv g_{\geq 1}(b)$ and $b^c = \emptyset \equiv g_{\geq \mathbf{n}}(b)$. Therefore, for uniformity of the presentation, given a set T of thresholds, we define $\hat{T} \stackrel{\text{def}}{=} T \cup \{1, \mathbf{n}\}$ 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 Simple Formulas in the Threshold Intersection Property Language

In this section, we discuss the first property of Γ-valid TIP formulas that is key to the development of an algorithm for inferring all of them automatically. A naïve (non-terminating) algorithm would iteratively check Γ-validity of every TIP formula. Instead, relying on the following condition, which essentially

states that no guard is "equivalent" to $g_{\geq 0}$ with respect to Γ, we show that there is no need to check every TIP formula.

Definition 4. *T is* Γ-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 Γ-valid, and $\forall x : g_{\geq t}.\ g_{\geq t'}(x \cap b)$ is never Γ-valid unless t' is also degenerate. (Note that checking if a threshold is degenerate amounts to a (non) entailment check in BAPA, which is decidable.)

When T is Γ-non-degenerate, we observe that we can (i) push conjunctions outside of formulas (since $\forall$ distributes over $\wedge$), and (ii) ignore terms of the form x^c as, under the assumption that T is Γ-non-degenerate, they will not appear in Γ-valid formulas. These observations, formalized in Lem. 2, justify focusing on *simple* formulas:

Definition 5. *A TIP formula φ is called* simple *if there exist $t \in \hat{T}$, $t_1 \dots t_q \in T$ and $h_1 \dots h_k \in \hat{Prm}_S$ such that $q + k > 0$, the h_i's are distinct and φ is of the form:*

$$\forall x_1 : g_{\geq t_1} \dots \forall x_q : g_{\geq t_q}.\ g_{\geq t}(x_1 \cap \dots \cap x_q \cap h_1 \dots \cap h_k)$$

We refer to $g_{\geq t}$ as φ's atomic guard*, and denote said t with $g_\varphi \in \hat{T}$*

The following lemma shows that enumerating simple formulas is sufficient to achieve the goal of finding all Γ-valid TIP formulas.

Lemma 2. *If T is Γ-feasible and Γ-non-degenerate, then for every Γ-valid φ in TIP, there exist $\varphi_1, \dots, \varphi_m$ TIP formulas s.t. $\varphi \equiv \bigwedge_{i=1}^m \varphi_i$ and every φ_i is simple.*

Proof. Let φ be a formula in TIP such that $\Gamma \models \varphi$. Since universal quantification distributes over conjunction, there exist $\varphi_1, \dots, \varphi_m$ such that $\varphi \equiv \bigwedge_{i=1}^m \varphi_i$ and for every $1 \leq i \leq m$, the formula φ_i is conjunction-free, i.e., φ_i is a quantified atomic formula.

Since φ is Γ-valid, each of the φ_i's is Γ-valid as well. As for any interpretation $\mathcal{I}$ and any set term b, it holds that (1) $\mathcal{I}(b \cap \emptyset^c) = \mathcal{I}(b)$, and (2) $\mathcal{I}(b \cap b) = \mathcal{I}(b)$, the term $\emptyset^c$ as well as duplicated instances of terms in φ_i can be omitted, resulting in an equivalent formula. Furthermore, if the term $\emptyset$ is in some φ_i, the fact that T is Γ-non-degenerate would mean that that φ_i is not Γ-valid, in contradiction to the assumption. That means that the term $\emptyset$ is not in any φ_i. Therefore, we can assume that all constant terms in φ_i are distinct and in $\hat{Prm}_S$. We are left with showing that x^c will not appear in Γ-valid formulas.

Consider such φ_i with an atomic guard t_b and a universally quantified variable x with guard t_a. Recall that T is Γ-non-degenerate and let $s_B = (\mathcal{D}, \mathcal{I})$ be such that $s_B \models \Gamma$ but $s_B \not\models t_b \leq 0$, i.e., $\mathcal{I}(t_b) > 0$. Since T is also Γ-feasible, we have that in particular $\mathcal{I}(t_a) \leq \mathcal{I}(\mathbf{n})$. Therefore, taking a valuation where $\iota(x) = \mathcal{I}(\emptyset^c)$ shows that $s_B \models \exists x.\ |x| \geq t_a \wedge |x^c| < t_b$. Let b be a set term. As it must hold that $\mathcal{I}(x^c \cap b) \subseteq \mathcal{I}(x^c)$, we get that $s_B \models \exists x.\ |x| \geq t_a \wedge |x^c \cap b| < t_b$, hence $\forall x : g_{\geq t_a} \dots . g_{\geq t_b}(x^c \cap b)$ is not Γ-valid. □

Lem. 2 shows that it is sufficient to search the space of *simple* TIP formulas in order to effectively find all Γ-valid TIP formulas. Therefore, for the rest of this thesis we mainly consider simple TIP formulas.

5.2 Finiteness Property

In this section, we characterize the cases in which there are finitely many Γ-valid TIP formulas, up to equivalence.

Definition 6. *T is* Γ-sane *if for every $t_1, t_2 \in T$, $\Gamma \not\models t_1 \leq 0 \vee t_2 > \mathbf{n} - 1$.*

Saneness implies that no threshold in T is equivalent to 0 or to $\mathbf{n}$ under Γ (in particular, this implies non-degeneracy). In fact, it captures a stronger requirement that for every pair of thresholds, there is a model of Γ in which one of them is not interpreted as 0 and the other is not interpreted as $\mathbf{n}$.

Theorem 2. *Assume that T is Γ-feasible. Then the following conditions are equivalent:*

1. *There are finitely many Γ-valid simple formulas (up to variable naming).*
2. *There are finitely many Γ-valid TIP formulas, up to equivalence.*
3. *T is* Γ-sane.

The interesting part in the theorem is the implication (3) $\Rightarrow$ (1). The essence of the proof of this implication is showing that there is a bound on the number of quantifiers that Γ-valid TIP formulas can have (depending on Γ). Lem. 4 proves that such a bound exists for each threshold pair (consisting of a quantifier guard and an atomic guard), and Lem. 3 shows that omitting guarded variables or constant terms from a Γ-valid simple TIP formula yields a new Γ-valid simple TIP formula. Combining these properties, we prove that (3) implies (1).

Lemma 3. *For any $t_1 \dots t_q \in T$, $t \in \hat{T}$, $h_1 \dots h_k \in \hat{Prm_S}$, it holds that*

$$\forall x_1 : g_{\geq t_1} \dots \forall x_q : g_{\geq t_q}.\ g_{\geq t}\left(\bigcap_{i=1}^{q} x_i \cap \bigcap_{i=1}^{k} h_i\right) \models \forall x_1 : g_{\geq t_1} \dots \forall x_{q-1} : g_{\geq t_{q-1}}.\ g_{\geq t}\left(\bigcap_{i=1}^{q-1} x_i \cap \bigcap_{i=1}^{k} h_i\right)$$

$$\forall x_1 : g_{\geq t_1} \dots \forall x_q : g_{\geq t_q}.\ g_{\geq t}\left(\bigcap_{i=1}^{q} x_i \cap \bigcap_{i=1}^{k} h_i\right) \models \forall x_1 : g_{\geq t_1} \dots \forall x_q : g_{\geq t_q}.\ g_{\geq t}\left(\bigcap_{i=1}^{q} x_i \cap \bigcap_{i=1}^{k-1} h_i\right)$$

Proof. For any valuation ι it holds that

$$\iota\left(\bigcap_{i=1}^{q} x_i \cap \bigcap_{i=1}^{k} h_i\right) \subseteq \iota\left(\bigcap_{i=1}^{q-1} x_i \cap \bigcap_{i=1}^{k} h_i\right)$$

$$\iota\left(\bigcap_{i=1}^{q} x_i \cap \bigcap_{i=1}^{k} h_i\right) \subseteq \iota\left(\bigcap_{i=1}^{q} x_i \cap \bigcap_{i=1}^{k-1} h_i\right)$$

Therefore, any structure that satisfies the first formula would satisfy the second one. □

Lemma 4. *If T is Γ-sane, then for every $t_a \in T$ and $t_b \in \hat{T}$, there exists a number Q_{t_a,t_b} s.t. for every $q \geq Q_{t_a,t_b}$, $\Gamma \not\models \forall x_1 : g_{\geq t_a} \dots . \dots x_q : g_{\geq t_a} . g_{\geq t_b}(x_1 \cap \dots \cap x_q)$*

Proof. Let $t_a \in T, t_b \in \hat{T}$ be arbitrary guards. As T is Γ-sane, there is a structure $s_B = (\mathcal{D}, \mathcal{I})$ such that $s_B \models \Gamma$ and $s_B \models t_a \leq \mathbf{n} - 1 \wedge t_b > 0$ if $t_b \in T$, or $s_B \models t_a \leq \mathbf{n} - 1 \wedge t_a > 0$ otherwise. Either way, we have that $\mathcal{I}(t_b) > 0$ and $\mathcal{I}(t_a) \leq \mathcal{I}(\mathbf{n}) - 1$.

We define $Q_{t_a,t_b} = \mathcal{I}(\mathbf{n})$. Wlog the universal set of s_B is $U = \{e_1 \dots e_{Q_{t_a,t_b}}\}$. We define a valuation ι for s_B in which for every $1 \leq i \leq Q_{t_a,t_b}$ we set $\iota(x_i) = E_i = U \setminus \{e_i\}$. We then get that $|\iota(x_i)| \geq \mathcal{I}(t_a)$ for all $1 \leq i \leq Q_{t_a,t_b}$, and $|\cap_{1 \leq i \leq Q_{t_a,t_b}} \iota(x_i)| = 0 < \mathcal{I}(t_b)$. Hence, $\Gamma \not\models \forall x_1 : g_{\geq t_a} . \dots . x_{Q_{t_a,t_b}} : g_{\geq t_a} . g_{\geq t_b}(x_1 \cap \dots \cap x_{Q_{t_a,t_b}})$. By Lem. 3 this also holds for any $q \geq Q_{t_a,t_b}$. □

We are now ready to prove Thm. 2:

Proof (of Thm. 2). By Lem. 2, (1) implies (2). Then (2) implies (3) since whenever $\Gamma \models t_a > \mathbf{n}-1 \vee t_b \leq 0$, then any formula of the form $\forall x_1 : g_{\geq t_a} \ldots \forall x_q : g_{\geq t_a}. g_{\geq t_b}(x_1 \cap \ldots \cap x_q)$ is Γ-valid. Finally we show that (3) implies (1). Let $Q = \sum_{t_a \in T} \max_{t_b \in \hat{T}} Q_{t_a,t_b}$, where Q_{t_a,t_b} is the number defined by Lem. 4. Then every Γ-valid simple TIP formula α must have less than Q quantifiers, as otherwise there exists $t_a \in T$ such that at least $\max_{t_b \in \hat{T}} Q_{t_a,t_b}$ quantifiers have guard $g_{\geq t_a}$. This means that at least Q_{t_a,g_α} quantifiers have guard $g_{\geq t_a}$, and thus, from Lem. 3 and Lem. 4, we get that $\Gamma \not\models \alpha$. □

Thm. 2 ensures that there are finitely many Γ-valid (simple) TIP formulas. In fact, Lem. 3 implies a stronger property: that it is possible to identify that all Γ-valid simple formulas have been discovered. This is formulated by the following corollary:

Corollary 2. *If no simple TIP formula with q quantifiers is Γ-valid then no simple TIP formula with more than q quantifiers is Γ-valid.*

5.3 Efficiently Checking for Γ-Validity

At this point we have established that the space of Γ-valid (simple) TIP formulas is finite. Next, we address the issue of checking whether a single simple TIP formula is Γ-valid. Let $t_1 \ldots t_q \in T$, $h_1 \ldots h_k \in \hat{Prm}_S$, $t \in \hat{T}$, and consider the following simple TIP formula:

$$\alpha = \forall x_1 : g_{\geq t_1} \ldots \forall x_q : g_{\geq t_q}.\ g_{\geq t}(x_1 \cap \ldots \cap x_q \cap h_1 \cap \ldots \cap h_k)$$

It is possible to check for Γ-validity simply by using the $\mathcal{B}$ translation defined in Chapter 4 producing the formula:

$$\mathcal{B}(\alpha) = \forall x_1 \ldots x_q. \bigwedge_{i=1}^{q} (|x_i| \geq t_i) \rightarrow |x_1 \cap \ldots \cap x_q \cap h_1 \cap \ldots \cap h_k| \geq t.$$

Checking that α is Γ-valid amounts to ensuring that no model of Γ satisfies $\neg\mathcal{B}(\alpha)$. Notice that

$$\neg\mathcal{B}(\alpha) = \exists x_1 \ldots x_q. \bigwedge_{i=1}^{q} (|x_i| \geq t_i) \wedge |x_1 \cap \ldots \cap x_q \cap h_1 \cap \ldots \cap h_k| < t$$

is free of universal quantifiers, which makes it easy to check for unsatisfiability with standard BAPA solvers.

In fact, in the sequel we observe that as the guarded variables in α are universally quantified, we can equivalently check that Γ entails a BAPA formula that constrains relations between variables of sort int guarded by threshold expressions. To simplify the notation, in the following lemma we assume wlog that $k > 0$, otherwise we consider the equivalent formula with $k = 1$ and $h_1 = \emptyset^c$.

Lemma 5. *Given a simple TIP formula*

$$\alpha = \forall x_1 : g_{\geq t_1} \ldots \forall x_q : g_{\geq t_q}.\ g_{\geq t}(x_1 \cap \ldots \cap x_q \cap h_1 \cap \ldots \cap h_k),$$

we denote by $\{i_1, \ldots, i_m\} \subseteq \{1, \ldots, q\}$ a maximal subset of the indices whose thresholds are distinct (i.e., for each $1 \leq j \leq q$ there exists a unique $1 \leq \ell \leq m$ such that $t_j = t_{i_\ell}$). Let β be the following BAPA formula

$$\beta = \forall u_{t_{i_1}} \ldots u_{t_{i_m}}. \bigwedge_{j=1}^{m} (0 \leq u_{t_{i_j}} \leq \mathbf{n} \wedge u_{t_{i_j}} \geq t_{i_j}) \rightarrow \sum_{i=1}^{q} (u_{t_i} - \mathbf{n}) + \left| \bigcap_{i=1}^{k} h_i \right| \geq t,$$

where the $u_{t_{i_\ell}}$ variables are of sort int. *Then, it holds that $\Gamma \models \alpha$ iff $\Gamma \models t > 0 \rightarrow \beta$.*

Proof. **First direction** ($\Rightarrow$): Assume that $\Gamma \not\models t > 0 \to \beta$, and let $s_B = (\mathcal{D}, \mathcal{I})$ be such that $s_B \models \Gamma$ and $s_B \models t > 0 \wedge \neg\beta$, in other words $\mathcal{I}(t) > 0$ and there exist an assignment σ_β such that for every $j = 1 \dots m$, $0 \leq \sigma_\beta(u_{t_{i_j}}) \leq \mathcal{I}(\mathbf{n})$, $\sigma_\beta(u_{t_{i_j}}) \geq \mathcal{I}(t_{i_j})$ and

$$\sum_{i=1}^{q}(\sigma_\beta(u_{t_i}) - \mathcal{I}(\mathbf{n})) + \left|\mathcal{I}\left(\bigcap_{i=1}^{k} h_i\right)\right| < \mathcal{I}(t)$$

We build σ_α constructively with the following procedure: start by defining $\xi^{(0)} = \mathcal{I}(h_1 \cap \ldots \cap h_k)$. Then, for $i = 1 \dots q$:

1. If $\xi^{(i-1)} = \emptyset$, choose $\sigma_\alpha(x_i) = \emptyset^c$.
2. If $\left|\xi^{(i-1)}\right| + \sigma_\beta(u_{t_i}) \leq \mathcal{I}(\mathbf{n})$ then define $\sigma_\alpha(x_i) = \sigma_\alpha\left(\xi^{(i-1)}\right)^c$.
3. Otherwise, $\sigma_\beta(u_{t_i}) > \mathcal{I}(\mathbf{n}) - \left|\xi^{(i-1)}\right|$.
   Choose $\sigma_\alpha(x_i)$ such that $\sigma_\alpha\left(\xi^{(i-1)}\right)^c \subseteq \sigma_\alpha(x_i)$ and $|\sigma_\alpha(x_i)| = \sigma_\beta(u_{t_i})$.
4. Define $\xi^{(i)} = \sigma_\alpha(x_i) \cap \xi^{(i-1)}$ for the next step.

Note that for any $1 \leq i \leq q$, we have that $|\sigma_\alpha(x_i)| \geq \sigma_\beta(u_{t_i}) \geq \mathcal{I}(t_i)$. Also, by definition of $\xi^{(i)}$ we have that $\xi^{(i)} \subseteq \xi^{(i-1)}$. Take a look at $\xi^{(q)} = \sigma_\alpha(x_1 \cap \ldots \cap x_q \cap h_1 \cap \ldots \cap h_k)$. If $\xi^{(q)} = \emptyset$ then we have that $|\sigma_\alpha(x_1 \cap \ldots \cap x_q \cap h_1 \cap \ldots \cap h_k)| = |\emptyset| = 0 < \mathcal{I}(t)$.

Otherwise, $\xi^{(q)} \neq \emptyset$, which means each $\sigma_\alpha(x_i)$ was defined using step 3. We prove that $\left|\xi^{(i)}\right| = \sum_{j=1}^{i}(\sigma_\beta(u_{t_j}) - \mathcal{I}(\mathbf{n})) + \left|\xi^{(0)}\right|$ for every $0 \leq i \leq q$ by using the inclusion-exclusion principle:

$$\begin{aligned}
\left|\xi^{(i)}\right| - \left|\xi^{(i-1)}\right| &= \left|\sigma_\alpha(x_i) \cap \xi^{(i-1)}\right| - \left|\xi^{(i-1)}\right| \\
&= |\sigma_\alpha(x_i)| + \left|\xi^{(i-1)}\right| - \left|\sigma_\alpha(x_i) \cup \xi^{(i-1)}\right| - \left|\xi^{(i-1)}\right| \\
&= |\sigma_\alpha(x_i)| - \left|\sigma_\alpha(x_i) \cup \xi^{(i-1)}\right| \\
&= |\sigma_\alpha(x_i)| - \mathcal{I}(\mathbf{n}) \\
&= \sigma_\beta(u_{t_i}) - \mathcal{I}(\mathbf{n})
\end{aligned}$$

Using induction on i we get the result we needed, and we conclude that

$$\begin{aligned}
|\sigma_\alpha(x_1 \cap \ldots \cap x_q \cap h_1 \cap \ldots \cap h_k)| &= \left|\xi^{(q)}\right| \\
&= \sum_{i=1}^{q}(\sigma_\beta(u_{t_i}) - \mathcal{I}(\mathbf{n})) + \left|\xi^{(0)}\right| \\
&= \sum_{i=1}^{q}(\sigma_\beta(u_{t_i}) - \mathcal{I}(\mathbf{n})) + \left|\mathcal{I}\left(\bigcap_{i=1}^{k} h_i\right)\right| < \mathcal{I}(t)
\end{aligned}$$

σ_α therefore shows that $s_B \models \neg\alpha$, and thus $\Gamma \not\models \alpha$

Second direction ($\Leftarrow$): First notice that for any two sets $A, B \subseteq \mathbf{U}$ from the inclusion-exclusion principle we have that

$$|A \cap B| = |A| + |B| - |A \cup B| \geq |A| + |B| - |\mathbf{U}|$$

or in the general case using induction,

$$(*) \quad \left|\bigcap_{i=1}^{m} A_i\right| \geq \sum_{i=1}^{m} |A_i| - (m-1)|\mathbf{U}|$$

Assume that $\Gamma \not\models \alpha$, and let $s_B = (\mathcal{D}, \mathcal{I})$ be such that $s_B \models \Gamma$ and $s_B \models \neg\alpha$, in other words there exist an assignment σ_α such that

$$\sigma_\alpha(|x_1 \cap \cdots \cap x_q \cap h_1 \cap \cdots \cap h_q|) = |\sigma_\alpha(x_1) \cap \cdots \cap \sigma_\alpha(x_q) \cap \mathcal{I}(h_1 \cap \cdots \cap h_q)| < \mathcal{I}(t)$$

Define a new assignment σ_β with $\sigma_\beta(u_{t_{i_j}}) = \min_{\substack{i=1 \\ t_i = t_{i_j}}}^{q} |\sigma_\alpha(x_i)|$ for any $0 \leq j \leq m$. Note that indeed $0 \leq \sigma_\beta(u_{t_{i_j}}) \leq \mathcal{I}(\mathbf{n})$, $\sigma_\beta(u_{t_{i_j}}) \geq \mathcal{I}(t_{i_j})$ and from $(*)$ we get that

$$\begin{aligned}
&\sum_{i=1}^{q}(\sigma_\beta(u_{t_i}) - \mathbf{n}) + \mathcal{I}\left(\left|\bigcap_{i=1}^{k} h_i\right|\right) \\
=&\sum_{i=1}^{q}\sigma_\beta(u_{t_i}) + \left|\mathcal{I}\left(\bigcap_{i=1}^{k} h_i\right)\right| - q\mathbf{n} \\
\leq&\sum_{i=1}^{q}|\sigma_\alpha(x_i)| + |\mathcal{I}(h_1 \cap \cdots \cap h_k)| - q\mathbf{n} \\
\leq&|\sigma_\alpha(x_1) \cap \cdots \cap \sigma_\alpha(x_q) \cap \mathcal{I}(h_1 \cap \cdots \cap h_k)| < \mathcal{I}(t)
\end{aligned}$$

and therefore, $s_B \models \neg\beta$. Also, because $s_B \not\models \alpha$ we have that $s_B \models t > 0$, and in total $s_B \models t > 0 \wedge \neg\beta$. This concludes that $\Gamma \not\models t > 0 \rightarrow \beta$ as needed. □

Lem. 5 defines an alternative BAPA formula, β, that may be used instead of $\mathcal{B}(\alpha)$ to check the Γ-validity of α. As before, the formula is universally quantified, which translates to an existentially quantified formula in the corresponding unsatisfiability check. The alternative formula, β, is preferable to $\mathcal{B}(\alpha)$ for a couple of reasons. First, it uses reasoning over integers instead of sets (with the exception of the set parameters), which is easier for automatic solvers, but mainly, it uses less quantified variables than $\mathcal{B}(\alpha)$ (that inherits its quantifies from the original formula α). Moreover, no matter how many quantifiers are in α, the number of quantifiers in β is bounded by $|T|$. This means that even though the maximal number of quantifiers in α depends on Γ (see Lem. 4), the number of quantifiers in β does not.

5.4 From TIP to Axioms in EPR

In order to use a set of Γ-valid simple formulas, Δ, for verifying the discussed protocols, it needs to be translated to FOL axioms as described in Sec. 4.2. Next, we show how to ensure that the quantifier alternation graph (see Chapter 2) of $\mathcal{FO}(\Delta)$ is acyclic.

Observation 1. *A simple formula φ with atomic guard $g_{\geq t}$ induces quantifier alternation edges in $QA(\mathcal{FO}(\varphi))$ from the threshold sorts of the guards of its universal quantifiers (these are thresholds in T) to the threshold sort of t if $t \in T$ or to the* node *sort if $t = 1$. If $t = \mathbf{n}$, no quantifier alternation edges are induced by φ.*

Therefore, 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 Δ.

Definition 7. *T is* Γ-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 Γ-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.

To facilitate elimination of cyclic formulas, we also need to strengthen the saneness requirement (Def. 6) to refer to the set parameters as well:

Definition 8. (T, Prm_S) *is* Γ-sane *if* T *is* Γ*-sane and, in addition, for every* $t \in T$ *and* $a \in \hat{Prm}_S$, $\Gamma \not\models t \leq 0 \vee |a| = \mathbf{n}$.

Theorem 3. *Assume that* T *is* Γ*-feasible and* Γ*-acyclic and that* (T, Prm_S) *is* Γ*-sane. Let* Δ *be a set of* Γ*-valid simple formulas, and let* $\Delta' = \{\varphi \in \Delta \mid \varphi \text{ is not cyclic}\}$. *Then the verification conditions of the first-order transition system with axioms* $\mathcal{FO}(\Delta)$ *are valid if and only if they are valid with axioms* $\mathcal{FO}(\Delta')$. *Further, the quantifier alternation graph of* $\mathcal{FO}(\Delta')$ *is acyclic.*

To prove the theorem we first prove the following lemma:

Lemma 6. *If* T *is* Γ*-feasible and* (T, Prm_S) *is* Γ*-sane, then for every* Γ*-valid simple formula* φ, *if* φ *is cyclic, then* $\mathcal{FO}(\varphi) \equiv true$.

Proof. Let $\varphi = \forall x_1 : g_{\geq t_1} \ldots \forall x_q : g_{\geq t_q}.\ g_{\geq t}(x_1 \cap \ldots \cap x_q \cap h_1 \ldots \cap h_k)$ be a cyclic TIP formula. Following Observation 1, because φ is cyclic, there must exist t_i such that $t_i = t$. Assume without loss of generally that this is t_1 (the threshold associated with the first quantifier).

If $q > 1$, then we get from Lem. 3 that $\varphi \models \forall x_1 : g_{\geq t}.\ \forall x_2 : g_{\geq t_2}.g_{\geq t}(x_1 \cap x_2)$. Because T is Γ-sane, there is $s_B = (\mathcal{D}, \mathcal{I})$ such that $s_B \models \Gamma$ but $s_B \models t > 0$ and $s_B \models t_2 \leq \mathbf{n} - 1$. Because T is Γ-feasible, we choose $\lceil \mathcal{I}(t) \rceil$ elements in $\mathcal{I}(\emptyset^c)$ (this is well-defined since Γ-feasibility ensures that $\lceil \mathcal{I}(t) \rceil \leq \mathcal{I}(\mathbf{n})$), let X_1 be the set of elements, with $e \in X_1$ an arbitrary element in the set. Let $X_2 = \mathcal{I}(\emptyset^c)/\{e\}$. Notice that $|X_1| = \lceil \mathcal{I}(t) \rceil \geq \mathcal{I}(t)$ and $|X_2| = \mathcal{I}(\mathbf{n}) - 1 \geq \mathcal{I}(t_2)$. Following that, a valuation ι over s_B exists such that $\iota(x_1) = X_1$, $\iota(x_2) = X_2$, hence the guards of the quantifiers are satisfied, but $|\iota(x_1) \cap \iota(x_2)| = \lceil \mathcal{I}(t) \rceil - 1 < \mathcal{I}(t)$, from which we conclude that φ is not Γ-valid. This means that $q = 1$.

Similarly, assume $k > 0$, then we have that $\varphi \models \forall x : g_{\geq t}.\ g_{\geq t}(x \cap h_1)$, and because (T, Prm_S) is Γ-sane (which ensures that there is s_B such that $s_B \models \Gamma$ but $s_B \models t > 0$ and $s_B \models |h_1| < \mathbf{n}$), we again conclude that φ is not Γ-valid. This means that $k = 0$. The conclusion is that $\varphi = \forall x : g_{\geq t}.\ g_{\geq t}(x)$, for which $\mathcal{FO}(\varphi) \equiv$ *true*. □

Proof (of Thm. 3). Let Δ be a set of Γ-valid simple TIP formulas, and let Δ' be defined as in the theorem. Lem. 6 shows that $\mathcal{FO}(\Delta') \models \mathcal{FO}(\Delta)$ as required. Assume $QA(\mathcal{FO}(\Delta'))$ is cyclic with a cycle α. Following Observation 1, the sort node cannot be in α. Let $t_1 \ldots t_m \in T$ be such that $\alpha = \mathsf{set}_{\mathsf{t}_1} \to \ldots \to \mathsf{set}_{\mathsf{t}_m} \to \mathsf{set}_{\mathsf{t}_1}$. Every edge $(\mathsf{set}_{\mathsf{t}_i}, \mathsf{set}_{\mathsf{t}_j})$ in α is obtained from a simple formula $\varphi \in \Delta'$ for which, from Lem. 3, we get that $\varphi \models \forall x : g_{\geq t_i}.\ g_{\geq t_j}(x)$, and therefore $\Gamma \models \forall x : g_{\geq t_i}.\ g_{\geq t_j}(x)$, which means $\Gamma \models t_i \geq t_j$. Finally, from transitivity, we get that $\Gamma \models t_i = t_j$ and because T is Γ-acyclic, we get that $t_i = t_j$, i.e., α corresponds to a self loop over a single sort $\mathsf{set}_{\mathsf{t}_1}$. By Observation 1, a self loop may only arise from a cyclic formula. Hence, each cycle in $QA(\mathcal{FO}(\Delta'))$ is induced by a single cyclic formula. As Δ' contains no cyclic formulas, we conclude that $QA(\mathcal{FO}(\Delta'))$ contains no cycles. □

5.5 Finding Minimal Properties Required for a Protocol

If Δ consists of *all* (simple) Γ-valid TIP formulas (except cyclic formulas as shown in Thm. 3), using $\mathcal{FO}(\Delta)$ as FO axioms is likely to lead to divergence of the verifier. To overcome this, we propose two variants.

Minimal Equivalent Δ_{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 φ_i from Δ. 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 Δ_{min} satisfies this condition. That is, Δ_{min} is a local minimum for that property.

Interpolant Δ_{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 Δ_{int} iteratively. Initially, $\Delta_{int} = \emptyset$. In each iteration, we check the VCs. If a counterexample to induction (CTI) is found, we add to Δ_{int} a formula from Δ not satisfied by the CTI. In this approach, Δ is not pre-computed, but computed lazily to generate candidate formulas in reaction to CTIs (assuming the implemented algorithm used to find Δ supports such behavior).

Chapter 6

Efficient Enumeration of TIP Formulas

To complete the task of finding a suitable set of threshold properties for the verification of a given protocol, it remains to find all Γ-valid simple TIP formulas, a subset of which will be used to prove the FO verification conditions (as explained in Sec. 5.5). In this chapter we address the problem of efficiently enumerating all Γ-valid simple formulas, for a given set of resilience conditions, Γ. In order to do so, we first define a subsumption relation between simple formulas that implies logical implication, and can be easily computed. We then present two algorithms that use this relation to prune the search space over which the enumeration is performed: the first one is rather simple, and the second one uses the Marco algorithm [26].

We notice that each simple formula φ is characterized (up to equivalence) by its atomic guard ($g_\varphi \in \hat{T}$), the number of quantified variables in φ that are guarded by every threshold $t \in T$ and, for each $h \in \hat{Prm}_S$, whether or not it appears as a term in φ. Therefore it is useful to represent φ by g_φ together with the following function:

Definition 9. *For a simple TIP formula $\varphi = \forall x_1 : g_{\geq t_1} \ldots \forall x_q : g_{\geq t_q}.\ g_{\geq g_\varphi}(x_1 \cap \ldots \cap x_q \cap h_1 \cap \ldots \cap h_k)$, we define χ_φ to be the function $\chi_\varphi : T \cup \hat{Prm}_S \to \mathbb{N}$ such that $\chi_\varphi(t) = |\{1 \leq i \leq q \mid t_i = t\}|$ for every $t \in T$, and for any $a \in \hat{Prm}_S$, $\chi_\varphi(a) = 1$ if $a = h_i$ for some $1 \leq i \leq k$, otherwise $\chi_\varphi(a) = 0$.*

Note that for any simple formulas φ_1 and φ_2, if $(g_{\varphi_1}, \chi_{\varphi_1}) = (g_{\varphi_2}, \chi_{\varphi_2})$ then φ_1 and φ_2 are **equivalent**, but they are not necessarily **equal**.

6.1 Simple Formula Subsumption

In this section we define a syntactic subsumption relation between formulas, which allows deducing Γ-(in)validity of formulas from the Γ-(in)validity of other formulas, while avoiding Γ-validity checks. We start by defining a subsumption relation between thresholds and parameters.

Definition 10 (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 $\Gamma \models |h_1| \geq h_2$.*

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

Definition 11. *Given simple formulas α and β we say that $\alpha \sqsubseteq^{(1)}_{\Gamma} \beta$ if one of the following holds:*

1. ***Atomic guard subsumption.*** *$\chi_\alpha = \chi_\beta$ and $g_\alpha \sqsubseteq_\Gamma g_\beta$.*
2. ***Quantifier guard subsumption.*** *$g_\alpha = g_\beta$ and there exist $h, h' \in T \cup \hat{Prm}_S$ such that $h \sqsubseteq_\Gamma h'$, and χ_α and χ_β agree on every input except $\chi_\beta(h) = \chi_\alpha(h) + 1$ and $\chi_\beta(h') = \chi_\alpha(h') - 1$.*
3. ***Term reduction.*** *$g_\alpha = g_\beta$ and there exist $h \in T \cup \hat{Prm}_S$ such that χ_α and χ_β agree on every input except $\chi_\beta(h) = \chi_\alpha(h) - 1$.*

We say that $\alpha \sqsubseteq_\Gamma \beta$ if there exist a sequence of simple formulas $\varphi_0 \dots \varphi_m$ such that $\varphi_0 = \alpha$, $\varphi_m = \beta$ with $\varphi_i \sqsubseteq^{(1)}_{\Gamma} \varphi_{i+1}$ for every $0 \leq i < m$.

Lemma 7 (Soundness)**.** *Let α, β be simple formulas such that $\alpha \sqsubseteq_\Gamma \beta$. Then, $\Gamma \models \alpha$ implies that $\Gamma \models \beta$.*

Proof. To prove this lemma we use induction on the length of the sequence of TIP formulas, m. When $m = 0$ we have that $\alpha = \beta$ and the result is immediate.

Induction step: We have that if $\Gamma \models \alpha$ then $\Gamma \models \varphi_{m-1}$, and we prove that if $\Gamma \models \varphi_{m-1}$ then $\Gamma \models \varphi_m = \beta$ by splitting to three cases, each corresponding to another case in the definition of $\varphi_{m-1} \sqsubseteq^{(1)}_{\Gamma} \varphi_m$. To simplify the notation, in the remainder of the proof we will denote $\alpha = \varphi_{m-1}$:

(i) **Atomic guard subsumption.** Let

$$\alpha \equiv \forall x_1 : g_{\geq h_1} \dots \forall x_q : g_{\geq h_q}.\ g_{\geq g_\alpha}(x_1 \cap \dots \cap x_q \cap h_{q+1} \dots \cap h_k)$$
$$\beta \equiv \forall x_1 : g_{\geq h_1} \dots \forall x_q : g_{\geq h_q}.\ g_{\geq g_\beta}(x_1 \cap \dots \cap x_q \cap h_{q+1} \dots \cap h_k)$$

Assume that $\Gamma \models \alpha$ and let $s_B = (\mathcal{D}, \mathcal{I})$ be such that $s_B \models \Gamma$. Because $g_\alpha \sqsubseteq_\Gamma g_\beta$ we get that $\mathcal{I}(g_\alpha) \geq \mathcal{I}(g_\beta)$. Therefore for any set term b if $|\mathcal{I}(b)| \geq \mathcal{I}(g_\alpha)$ then $|\mathcal{I}(b)| \geq \mathcal{I}(g_\beta)$, and we get that because $s_B \models \alpha$, $s_B \models \beta$ as well.

(ii) **Quantifier guard subsumption.** Separate to two cases: if $h_1, h_2 \in T$ then let α and β be such that

$$\alpha \equiv \forall x_1 : g_{\geq h_1} \dots \forall x : g_{\geq h'} \dots \forall x_q : g_{\geq h_q}.\ g_{\geq t}(x_1 \cap \dots x \dots \cap x_q \cap h_{q+1} \dots \cap h_k)$$
$$\beta \equiv \forall x_1 : g_{\geq h_1} \dots \forall x : g_{\geq h} \dots \forall x_q : g_{\geq h_q}.\ g_{\geq t}(x_1 \cap \dots x \dots \cap x_q \cap h_{q+1} \dots \cap h_k)$$

Assume that $\Gamma \models \alpha$ and let $s_B = (\mathcal{D}, \mathcal{I})$ be such that $s_B \models \Gamma$. Because $h \sqsubseteq_\Gamma h'$ we get that $\mathcal{I}(h) \geq \mathcal{I}(h')$. Therefore for any formula $\varphi(x)$ if $s_B \models \forall x : g_{\geq h'}.\ \varphi(x)$ then $s_B \models \forall x : g_{\geq h}.\ \varphi(x)$, and we get that because $s_B \models \alpha$, $s_B \models \beta$ as well.

Otherwise, $h \in \hat{Prm}_S$ and $h' \in T$. Then let α and β be such that

$$\alpha \equiv \forall x_1 : g_{\geq h_1} \dots \forall x : g_{\geq h'} \dots \forall x_q : g_{\geq h_q}.\ g_{\geq t}(x_1 \cap \dots x \dots \cap x_q \cap h_{q+1} \dots \cap h_k)$$
$$\beta \equiv \forall x_1 : g_{\geq h_1} \dots \forall x_q : g_{\geq h_q}.\ g_{\geq t}(x_1 \cap \dots h \dots \cap x_q \cap h_{q+1} \dots \cap h_k)$$

Assume that $\Gamma \models \alpha$ and let $s_B = (\mathcal{D}, \mathcal{I})$ be such that $s_B \models \Gamma$. Because $h' \sqsubseteq_\Gamma h$ we get that $|\mathcal{I}(h)| \geq \mathcal{I}(h')$. Therefore for any formula $\varphi(x)$ if $s_B \models \forall x : g_{\geq h'}.\ \varphi(x)$ then $s_B \models \varphi(h)$, and we get that because $s_B \models \alpha$, $s_B \models \beta$ as well.

(iii) **Term reduction.** Follows from Lem. 3.

□

The subsumption relation $\sqsubseteq_\Gamma$ allows us to avoid checking every simple TIP formula for Γ-validity. Namely, if $\alpha \sqsubseteq_\Gamma \beta$, then having checked that $\Gamma \models \alpha$, we can deduce that $\Gamma \models \beta$, and vice versa, having checked that $\Gamma \not\models \beta$, we can deduce that $\Gamma \not\models \alpha$.

Algorithm 1: Algorithm for Inferring Intersection Properties (AIP)

Input: Prm_S, T, Γ
set checked_true = checked_false = [ ] ;
foreach $q = 0, 1, \ldots$ **do**
  foreach *simple formula* φ *over* T *and* Prm_S *with* q *quantifiers* **do**
    if *exists* $\psi \in$ *checked_true s.t.* $\psi \sqsubseteq_\Gamma \varphi$ **then** yield φ ;
    else if *exists* $\psi \in$ *checked_false s.t.* $\varphi \sqsubseteq_\Gamma \psi$ **then** continue ;
    else if $\Gamma \models \varphi$ **then** yield φ ; add φ to checked_true ;
    else add φ to checked_false ;
  if *no formulas were added to checked_true* **then** terminate ;

Efficiently checking for subsumption. To make the use of subsumption worthwhile, it is left for us to show that checking $\alpha \sqsubseteq_\Gamma \beta$ is easy. This is guaranteed by the following lemma.

Lemma 8 (Checking for subsumption)**.** *Let α, β be simple formulas. Then $\alpha \sqsubseteq_\Gamma \beta$ iff (i) $g_\alpha \sqsubseteq_\Gamma g_\beta$, and (ii) for every $h, h' \in T \cup \hat{Prm}_S$ where $h' \sqsubseteq_\Gamma h$ there exists $e_{(h',h)} \in \mathbb{N}$ such that for every $r \in T \cup \hat{Prm}_S$ the following inequality holds:*

$$\chi_\beta(r) \leq \chi_\alpha(r) - \sum_{h \in T \cup \hat{Prm}_S} e_{(h,r)} + \sum_{h \in T \cup \hat{Prm}_S} e_{(r,h)}.$$

Proof. ($\Rightarrow$): Easy to prove using induction on the length of the sequence constructing $\alpha \sqsubseteq_\Gamma \beta$.

($\Leftarrow$): Constructively build the sequence of simple formulas showing that $\alpha \sqsubseteq_\Gamma \beta$, using $e_{(h,h')}$ as the number of times to use the *quantifier guard subsumption* step. □

Lem. 8 shows that checking subsumption between simple TIP formulas reduces to checking subsumption between thresholds and parameters, and checking satisfiability of a PA formula defined as follows (to establish the second condition). First, for any $h, h' \in T \cup \hat{Prm}_S$ such that $h' \sqsubseteq_\Gamma h$, a variable $u_{(h',h)}$ of sort int is defined. The formula whose satisfiability characterizes the second condition is then the following formula, where $\bar{u}$ denotes the sequence of all the $u_{(h',h)}$ variables.

$$\exists \bar{u}. \bigwedge_{\substack{h,h' \in T \cup \hat{Prm}_S \\ h' \sqsubseteq_\Gamma h}} (u_{(h',h)} \geq 0) \wedge \bigwedge_{r \in T \cup \hat{Prm}_S} \left(\chi_\beta(r) \leq \chi_\alpha(r) - \sum_{\substack{h \in T \cup \hat{Prm}_S \\ h \sqsubseteq_\Gamma r}} u_{(h,r)} + \sum_{\substack{h \in T \cup \hat{Prm}_S \\ r \sqsubseteq_\Gamma h}} u_{(r,h)} \right)$$

6.2 Iterative Enumeration

Alg. 1 depicts AIP that generates all Γ-valid simple formulas, relying on Lem. 7. Alg. 1 is designed to emit all Γ-valid simple formulas, including ones that are subsumed or entailed by others, since BAPA entailment does not imply entailment between the FO translations [1]. AIP uses a naïve search strategy; in Sec. 6.3 we show how different and more sophisticated strategies can be used. AIP terminates if for some number of quantifiers no Γ-valid formula is discovered. This is justified by Cor. 2.

Lemma 9 (Soundness)**.** *Every formula φ that is returned by* AIP *is Γ-valid.*

[1] It is worth mentioning that if the FO formulas that encode subsumption over $\hat{T} \cup \hat{Prm}_S$ (Def. 10) are included in the FO translations (these are the formulas $\mathcal{FO}(\forall x : g_{\geq h}.\ g_{\geq t}(x))$ for $h \sqsubseteq_\Gamma t$ and $\mathcal{FO}(g_{\geq t}(a))$ for $a \sqsubseteq_\Gamma t$ where $h, t \in \hat{T}$ and $a \in \hat{Prm}_S$), then subsumption between simple formulas does actually imply entailment between the FO translations.

Proof. It follows from Lem. 7 that in each step of AIP, the set *checked_true* consists only of TIP formulas that are Γ-valid. Therefore, all the formulas returned by AIP are Γ-valid as well. □

Lemma 10 (Completeness)**.** *If T is Γ-feasible and Γ-non-degenerate, then for every Γ-valid TIP formula φ there exist $\varphi_1 \dots \varphi_m$ s.t. $\varphi \equiv \bigwedge_{i=1}^{m} \varphi_i$ and* AIP *yields every φ_i.*

Proof. Let φ be a Γ-valid TIP formula. From Lem. 2 it follows that there exist simple formulas $\varphi_1 \dots \varphi_m$ such that $\varphi \equiv \bigwedge_{i=1}^{m} \varphi_i$, all Γ-valid.

Assume that some φ_i was not reached by the algorithm before it has terminated. Let q be the number of quantifiers in φ_i, then there exists a number $q' < q$ for which no formula with q' quantifier is Γ-valid. But this contradicts Cor. 2.

This means that at some point the algorithm would visit φ_i at the inner loop, and as *checked_true* holds at every point only formulas that were proved to be Γ-valid, according to lem. 7, the algorithm would yield φ_i. □

Lemma 11 (Termination)**.** *If T is Γ-feasible and Γ-sane,* AIP *terminates.*

Proof. From Thm. 2 we conclude that there are finitely many Γ-valid simple TIP formulas, which means at some point the algorithm would reach the maximum number of quantifier that yield Γ-valid formulas, and terminate. □

6.3 Finding a Cut in the Subsumption Graph

The key observation for this section, based on Lem. 7, is that the relation $\sqsubseteq_\Gamma$ between simple TIP formulas is actually a Γ-validity separator for simple TIP formulas: for any two simple TIP formulas φ_1 and φ_2 such that $\varphi_1 \sqsubseteq_\Gamma \varphi_2$, we have that $\Gamma \models \varphi_1 \Rightarrow \Gamma \models \varphi_2$. This property can be used to treat the problem of inferring Γ-valid TIP formulas as a problem of *Minimal Unsatisfiable Subset (MUS)* [26] enumeration. In this section we present the TIP inference problem as a MUS problem, which we then solve with the MARCO algorithm. To this end, we first formulate the MUS problem in a more general way that is suitable for our purpose.

The MUS problem. We first lay the grounds for discussing the problem of MUS enumeration. Let Σ be a set of elements (which we call the *search space*), where each element is classified either as "positive" or "negative", as given by a predicate $\Psi \subseteq \Sigma$ that consists of all the positive elements. Further, let $\preceq^{(1)}$ be a relation over Σ with the following properties:

- **Separation** For any two elements $\sigma_1, \sigma_2 \in \Sigma$ if $\sigma_1 \preceq^{(1)} \sigma_2$ then $\sigma_1 \in \Psi \Rightarrow \sigma_2 \in \Psi$.
- **Finite Branching** For any $\sigma_1 \in \Sigma$, the sets $\{\sigma_2 \in \Sigma \mid \sigma_1 \preceq^{(1)} \sigma_2\}$ and $\{\sigma_2 \in \Sigma \mid \sigma_1 \succeq^{(1)} \sigma_2\}$ are finite.
- **A-cyclicity** For any path $\sigma_0 \preceq^{(1)} \dots \preceq^{(1)} \sigma_m$ with $m > 0$ it holds that $\sigma_0 \neq \sigma_m$ (note that with $m = 1$, this implies anti-reflexivity).

For every $\sigma \in \Sigma$ we denote $\sigma \preceq^{(0)} \sigma$, and we lift $\preceq^{(1)}$ to $i > 1$ by denoting $\sigma_1 \preceq^{(i)} \sigma_2$ if there exists an element $\sigma' \in \Sigma$ such that $\sigma_1 \preceq^{(i-1)} \sigma'$ and $\sigma' \preceq^{(1)} \sigma_2$. Finally, denote $\sigma_1 \preceq \sigma_2$ if for some $i \geq 0$

it holds that $\sigma_1 \preceq^{(i)} \sigma_2$. Namely, $\preceq$ is the reflexive transitive closure of $\preceq^{(1)}$, and with the a-cyclicity property, defines a partial order over Σ.

In this setting, we identify elements of Σ that have particular properties concerning Ψ:

- A *minimal negative element* (MNE) is an element $\sigma \in \Sigma$ such that $\sigma \notin \Psi$ and $\forall \sigma' \in \Sigma : \sigma \preceq \sigma' \Rightarrow \sigma' \in \Psi$.
- A *maximal positive element* (MPE) is an element $\sigma \in \Sigma$ such that $\sigma \in \Psi$ and $\forall \sigma' \in \Sigma : \sigma' \preceq \sigma \Rightarrow \sigma' \notin \Psi$.

Definition 12. *For a given search space Σ, predicate Ψ and a relation $\preceq^{(1)}$ following the above requirements, the* MUS enumeration problem *is to find all MPEs and MNEs as defined above.*

In typical instances of the problem (and in particular, in the original specification [26]), Σ would be exponential in the size of the problem, and would thus require high complexity to iterate over. Moreover, the calculation of Ψ might demand high resources as well. In such cases, we would need an efficient way of finding the MNEs and MPEs in Σ, using the relation $\preceq$ instead of checking whether an element is in Ψ or not. [2]

6.3.1 The MARCO Algorithm

In this section we describe the *MARCO* algorithm that solves the MUS enumeration problem for a given Σ, Ψ and $\preceq^{(1)}$. MARCO requires an efficient way to explore the search space Σ, or more particularly, explore the directed graph obtained by using the elements of Σ as vertices and the relation $\preceq^{(1)}$ as edges. To this end, MARCO uses an exploration map *Ex*, which maps elements in Σ to whether or not their membership in Ψ has already been determined:

$$Ex : \Sigma \to \{EXPLORED,\ UNEXPLORED\}$$

There are two primitive operations needed for MARCO: retrieving an unexplored element from *Ex* and marking as explored all elements related by $\preceq$ to an arbitrary element σ after computing whether σ is in Ψ or not (while concluding the same classification for every σ' such that $\sigma \preceq \sigma'$ if $\sigma \in \Psi$, or for every σ' such that $\sigma' \preceq \sigma$ if $\sigma \notin \Psi$). In the following discussion, $MarkExplored_{\preceq}(\sigma)$ is the operation of marking every $\sigma' \in \Sigma$ for which $\sigma' \preceq \sigma$ as explored, and $MarkExplored_{\succeq}(\sigma)$ marks every $\sigma' \in \Sigma$ for which $\sigma' \succeq \sigma$ as explored.

With Σ, Ψ, $\preceq^{(1)}$ and *Ex* defined as above, MARCO, presented in Alg. 2, finds all MPEs and MNEs. It does so by iteratively getting an unexplored element from *Ex*, checking if it is in Ψ, and traversing the graph obtained by $\preceq^{(1)}$ up or down until an MPE or an MNE is reached, as determined by a Ψ check, in which case all greater, respectively, smaller, elements are marked as explored.

The requirements of $\preceq^{(1)}$ and *Ex* ensure the correctness of the algorithm, as summarized next.

Lemma 12. *With Σ, Ψ, $\preceq^{(1)}$ and Ex as described above, we have that:*

1. ***Soundness*** *Every element returned by MARCO as an MPE or an MNE is in fact an MPE or an MNE, respectively.*
2. ***Completeness for MPEs*** *For every element $\sigma \in \Sigma$ such that $\sigma \in \Psi$ there exists an MPE σ' returned by MARCO such that $\sigma' \preceq \sigma$.*

[2] The MUS enumeration problem is described in [26] as a special case, where a set of constraints C is spanned such that $\Sigma = \mathcal{P}(C)$, Ψ consists of all subsets satisfying a given CNF formula, and $\preceq$ is simply the relation $\subseteq$.

Algorithm 2: MARCO Algorithm

```
   Input: Σ, Ψ, ≺^(1)
 9 set Ex = { } ;
10 set MPE, MNE = [ ] ;
11 while Ex.HasUnexplored do
12     set σ = Ex.GetUnexplored() ;
13     if σ ∈ Ψ then
14         for σ' ∈ Σ such that σ' ≺^(1) σ do
15             if σ' ∈ Ψ then
16                 set σ = σ' ;
17                 goto Line 13 ;
18         add σ to MPE ;
19         Ex.MarkExplored_≥(σ) ;
20     else
21         for σ' ∈ Σ such that σ' ≻^(1) σ do
22             if σ' ∉ Ψ then
23                 set σ = σ' ;
24                 goto Line 20 ;
25         add σ to MNE ;
26         Ex.MarkExplored_≺(σ) ;
27 return MPE, MNE
```

3. ***Completeness for MNEs*** *For every element $\sigma \in \Sigma$ such that $\sigma \notin \Psi$ there exists an MNE σ' returned by MARCO such that $\sigma' \succeq \sigma$.*

Proof. Immediate from the requirements of $\prec^{(1)}$ and *Ex*. □

Termination, on the other hand, is not that immediate. Of course, if Σ is finite then the algorithm is guaranteed to terminate. To guarantee termination otherwise, we require that Σ has a finite basis with regard to $\preceq$, as defined in the following definition:

Definition 13. *Let Σ be a set of elements, and $\preceq$ be a partial order defined over Σ. We say that $B \subseteq \Sigma$ is a* basis of Σ with regard to $\preceq$ *if for any $\sigma \in \Sigma$ there exists $b \in B$ with $\sigma \preceq b$.*

Note that if Σ has a finite basis both w.r.t. $\preceq$ and w.r.t. $\succeq$ then Σ is finite, as follows from the finite branching of the relation $\prec^{(1)}$. Lem. 13 shows that instead of requiring that Σ is finite, a finite basis of Σ w.r.t. $\preceq$ is sufficient to guarantee termination of MARCO, provided that Ψ is finite.

Lemma 13. *If Σ has a finite basis w.r.t. $\preceq$ and Ψ is finite, then the MARCO algorithm terminates.*

Proof. Followed by Def. 13, let $B \subseteq \Sigma$ be the finite basis w.r.t. $\preceq$ of Σ. We assume wlog that every $b \in B$ is maximal with relation to $\preceq$: if there exist an element $b_1 \in B$ that is not maximal, then there exist some $\sigma \neq b_1$ in Σ with $b_1 \preceq \sigma$. By Def. 13 there exists $b_2 \in B$ with $\sigma \preceq b_2$. From transitivity $b_1 \npreceq b_2$, and therefore we can eliminate b_1 from B.

Note that it is not necessary that $B \subseteq \Psi$, so for the proof we need to separate it into $B_1 = B \cap \Psi$ and $B_2 = B \setminus B_1$. All elements in B_2 are of course MNEs. For any $\sigma_1, \sigma_2 \in \Sigma$ such that $\sigma_1 \preceq \sigma_2$ we define the distance between σ_1 and σ_2, $d(\sigma_1, \sigma_2)$, as the minimal d such that $\sigma_1 \prec^{(d)} \sigma_2$.

Look at the maximal distance of element in Ψ from element in B_1, $D = \max_{\substack{\sigma \in \Psi, b \in B_1 \\ \sigma \preceq b}} d(\sigma, b)$ (which is well defined as Ψ is finite). From finite branching of $\prec^{(1)}$, we have that $\mathbb{D} = \{\sigma \in \Sigma \mid \max_{b \in B_1, \sigma \preceq b} d(\sigma, b) \leq$

$D+1\}$ is finite. Also, every iteration of the algorithm must terminate eventually with an MPE or MNE as a result: if the initial element is in Ψ, then the iteration would terminate as Ψ is finite and eventually an MPE would be reached. Otherwise, the initial element is not in Ψ and the iteration would terminate when it reaches B (or sooner) with an MNE, as B only consists of maximal elements.

Finally, notice that every MPE is in $\Psi \subseteq \mathbb{D}$ and every MNE is either in B_2 or in $\mathbb{D}$, which means that after at most $|\mathbb{D}| + |B_2|$ iterations of the algorithm, every element in Σ would be marked as explored in *Ex*, and the algorithm would terminate. □

Similarly, requiring instead that Σ has a finite basis w.r.t. $\succeq$ and $\Sigma \setminus \Psi$ is finite, also guarantees termination of the MARCO algorithm.

6.3.2 TIP Enumeration with the Marco Algorithm

In this section we use the MARCO algorithm in order to find all Γ-valid simple formulas. First, we specify Σ, Ψ and $\preceq^{(1)}$ as required in the beginning of the section. Then, we specify the exploration map *Ex* and implement the required primitive operations of the MARCO algorithm: retrieving an unexplored element and marking additional elements as explored based on a single "root" element from Σ.

TIP enumeration as a MUS problem We define Σ to be the set of all simple TIP formulas up to equivalence where each formula φ is represented by $(g_\varphi,\ \chi_\varphi)$. The classification predicate corresponds to Γ-validity, i.e., $\Psi = \{\varphi \in \Sigma \mid \Gamma \models \varphi\}$, and the relation $\preceq^{(1)}$ is defined by $\sqsubseteq^{(1)}_\Gamma$, except when both formulas are equivalent: given the precomputed relation $\sqsubseteq_\Gamma$ between elements in $\hat{T} \cup \hat{Prm}_S$ we define for every simple TIP formulas φ_1, φ_2:

$$\varphi_1 \preceq^{(1)} \varphi_2 \text{ iff } (g_{\varphi_1} \neq g_{\varphi_2} \vee \chi_{\varphi_1} \neq \chi_{\varphi_2}) \wedge \varphi_1 \sqsubseteq^{(1)}_\Gamma \varphi_2.$$

Recall the three properties required above from $\preceq^{(1)}$: **separation** is shown in Lem. 7, **finite branching** is immediate from the definition of $\sqsubseteq^{(1)}_\Gamma$, and **a-cyclicity** is obtained by assuming that T is Γ-acyclic (Sec. 5.4), as shown in the following lemma.

Lemma 14. *Assume that T is Γ-acyclic. Then if $\varphi_0, \ldots, \varphi_m$ are simple TIP formulas with $\varphi_0 \preceq^{(1)} \ldots \preceq^{(1)} \varphi_m$ for $m > 0$, then $(g_{\varphi_0}, \chi_{\varphi_0}) \neq (g_{\varphi_m}, \chi_{\varphi_m})$.*

Proof. Assume to the contrary that $g_{\varphi_0} = g_{\varphi_m}$ and $\chi_{\varphi_0} = \chi_{\varphi_m}$. We prove the lemma by showing that no transition in the definition of $\sqsubseteq^{(1)}_\Gamma$ is a part of the sequence $\varphi_0 \preceq^{(1)} \ldots \preceq^{(1)} \varphi_m$, in contradiction to $m > 0$.

First, we handle steps of type **Atomic guard subsumption**. Look at the atomic guard sequence: $g_{\varphi_0} \sqsubseteq_\Gamma \ldots \sqsubseteq_\Gamma g_{\varphi_m} = g_{\varphi_0}$. Because $\sqsubseteq_\Gamma$ over $\hat{T}$ is transitive, for every $0 \leq i < m$ we have that $g_{\varphi_0} \sqsubseteq_\Gamma g_{\varphi_i} \sqsubseteq_\Gamma g_{\varphi_m} = g_{\varphi_0}$, and therefore, as T is Γ-acyclic we get that $g_{\varphi_0} = g_{\varphi_i}$. That means that if an *atomic guard subsumption* step is used we get that both g_{φ_i} and χ_{φ_i} are preserved which contradicts the definition of $\preceq^{(1)}$.

Denote $\tau_\varphi = \sum_{h \in T \cup \hat{Prm}_S} \chi_\varphi(h)$ for any simple TIP formula φ, and note that for a pair $\varphi_i \preceq^{(1)} \varphi_{i+1}$, if a *quantifier guard subsumption* step was applied then $\tau_{\varphi_i} = \tau_{\varphi_{i+1}}$, and if a *term reduction* step was applied then $\tau_{\varphi_i} > \tau_{\varphi_{i+1}}$. Therefore, because $\tau_{\varphi_0} = \tau_{\varphi_m}$ we get that the *term reduction* step is not applied as well.

That leaves us with the *quantifier guard subsumption*. Note that because T is Γ-acyclic, we can sort the elements in $T \cup \hat{Prm}_S$ by topological order, $h_1 \dots h_k$, meaning that for every i, j if $h_i \sqsubseteq_\Gamma h_j$ then $i \leq j$.

Look at h_1. Any *quantifier guard subsumption* either preserves $\chi_{\varphi_i}(h_1)$ or increases it by 1. Similarly to the τ_φ reasoning, because $\chi_{\varphi_0}(h_1) = \chi_{\varphi_m}(h_1)$, that means that it is not possible for such a step to replace a quantifier guarded by h_1. Now look at h_2. It is not possible to decrease $\chi_{\varphi_i}(h_2)$ with *quantifier guard subsumption* steps as we have established that no such step can be used with h_1. So, using a similar argument, that step cannot use h_2 as well. This procedure continues until h_k.

This concludes that no subsumption step can be used, and thus we get that $m = 0$, in contradiction. □

Note that with $\preceq^{(1)}$ defined as above, for any two simple TIP formula σ_1 and σ_2 we have that $\sigma_1 \preceq \sigma_2 \Leftrightarrow \sigma_1 \sqsubseteq_\Gamma \sigma_2$. Further, the set $B = \{\forall x : g_{\geq t_1}.\, g_{\geq t}(x) \mid t_1 \in T, t \in \hat{T}\} \cup \{g_{\geq t}(h) \mid h \in \hat{Prm}_S, t \in \hat{T}\}$ is a finite basis of Σ w.r.t. $\preceq$, and assuming that T is Γ-sane, we have that Ψ is finite (as shown in Thm. 2). Thus, the conditions listed in Lem. 13 for ensuring termination of MARCO are met.

With these definitions, the problem of finding all Γ-valid TIP formulas reduces to the problem of finding all MPEs: The reason is that, in general, $\Psi = \bigcup_{\sigma \text{ is MPE}} \{\sigma' \in \Sigma \mid \sigma \preceq \sigma'\}$. Therefore, the set of MPEs allows us to syntactically find *all* of the Γ-valid simple TIP formulas (using the subsumption relation). Further, recall that using the entire set of Γ-valid simple formulas as FO axioms when discharging the VCs often leads to divergence of the verifier (as discussed in Sec. 5.5). The following lemma shows that MPEs can be used to drastically decrease this set.

Lemma 15. *Assume that T is Γ-feasible, and denote*

$$\begin{aligned}\Delta = \{&\forall x : g_{\geq t_1}.\, g_{\geq t}(x) \mid t_1 \in T,\ t \in \hat{T},\ t_1 \sqsubseteq_\Gamma t\} \cup \\ &\{g_{\geq t}(h) \mid h \in \hat{Prm}_S, t \in \hat{T},\ h \sqsubseteq_\Gamma t\} \cup \\ &\{\varphi \in \Sigma \mid \varphi \text{ is an MPE}\}\end{aligned}$$

Then $\mathcal{FO}(\Delta) \models \mathcal{FO}(\Psi)$.

Proof. Let $\sigma \in \Psi$ be a Γ-valid simple TIP formula, which means there exists an MPE $\varphi \in \Delta$ with $\varphi \preceq \sigma$. In other words, $\varphi \sqsubseteq_\Gamma \sigma$, and there exists a sequence of simple TIP formulas with $\varphi = \sigma_0 \sqsubseteq_\Gamma^{(1)} \dots \sqsubseteq_\Gamma^{(1)} \sigma_m = \sigma$. For every $0 \leq i < m$, we describe how to deduce $\mathcal{FO}(\varphi_{i+1})$ from $\mathcal{FO}(\varphi_i)$ together with the atomic TIP formulas in Δ, based on the definition of $\sqsubseteq_\Gamma^{(1)}$.

1. **Atomic guard subsumption** In that case, $\chi_{\varphi_i} = \chi_{\varphi_{i+1}}$ and $g_{\varphi_i} \sqsubseteq_\Gamma g_{\varphi_{i+1}}$, in which case by definition of $\mathcal{FO}$:
$$\mathcal{FO}(\varphi_i) \wedge \mathcal{FO}\left(\forall x : g_{\geq g_{\varphi_i}}.\, g_{\geq g_{\varphi_{i+1}}}(x)\right) \models \mathcal{FO}(\varphi_{i+1})$$
2. **Quantifier guard subsumption** Here $g_{\varphi_i} = g_{\varphi_{i+1}}$ and there exist $h_1, h_2 \in T \cup \hat{Prm}_S$ with $h_1 \sqsubseteq_\Gamma h_2$ that form the subsumption between φ_i and φ_{i+1}. If $h_1 \in T$ then:
$$\mathcal{FO}(\varphi_i) \wedge \mathcal{FO}\left(\forall x : g_{\geq h_1}.\, g_{\geq h_2}(x)\right) \models \mathcal{FO}(\varphi_{i+1})$$
Otherwise, $h_1 \in \hat{Prm}_S$ and therefore:
$$\mathcal{FO}(\varphi_i) \wedge \mathcal{FO}\left(g_{\geq h_2}(h_1)\right) \models \mathcal{FO}(\varphi_{i+1})$$
3. **Term reduction** We simply get that
$$\mathcal{FO}(\varphi_i) \models \mathcal{FO}(\varphi_{i+1})$$

In conclusion, $\mathcal{FO}(\Delta) \models \mathcal{FO}(\sigma)$, as required. □

Exploration map. As mentioned above, in order to use the MARCO algorithm for enumerating all MPEs, we need to find a way to represent an exploration map of (simple) TIP formulas, for which it is easy to (i) retrieve an arbitrary formula that was not explored yet, and (ii) given a formula φ, mark every simple TIP formula ψ such that $\psi \sqsubseteq_\Gamma \varphi$ (or $\varphi \sqsubseteq_\Gamma \psi$) as explored. To this end, we encode the set of unexplored simple TIP formulas by a PA formula, U.

Recall that we encode a simple TIP formula φ by $(g_\varphi, \chi_\varphi)$ (see Def. 9). Therefore, to define U, we introduce a set C of uninterpreted constants that consists of:

- A boolean constant R_t for every $t \in \hat{T}$, used to capture g_φ.
- An integer constant Q_h for every $h \in T \cup \hat{Prm}_S$, used to capture χ_φ.

Note that, technically, PA as defined in Chapter 2 does not include boolean constants. However, it may easily be extended to include such constants (for example, by representing a boolean constant by a constrained integer constant).

Given C as above, a simple TIP formula φ may be represented (up to equivalence) by a structure $s_\varphi = (\mathbb{Z}, \mathcal{I})$ over C in which (i) for every $t \in \hat{T}$, $\mathcal{I}(R_t) = \mathit{true} \iff g_\varphi = t$, and (ii) for every $h \in T \cup \hat{Prm}_S$, $\mathcal{I}(Q_h) = \chi_\varphi(h)$. Similarly, in the other direction, a structure $s = (\mathbb{Z}, \mathcal{I})$ represents a simple TIP formula φ_s, provided that it satisfies the following formula

$$\rho = \bigvee_{t \in \hat{T}} \left(R_t \wedge \bigwedge_{t' \in \hat{T} \setminus \{t\}} \neg R_{t'} \right) \wedge \bigwedge_{h \in T} Q_h \geq 0 \wedge \bigwedge_{h \in \hat{Prm}_S} 0 \leq Q_h \leq 1.$$

$s \models \rho$ ensures that (i) $\mathcal{I}(R_t) = \mathit{true}$ for exactly one $t \in T$, (ii) $\mathcal{I}(Q_h) \geq 0$ for every $h \in T$, and (iii) $0 \leq \mathcal{I}(Q_h) \leq 1$ for every $h \in \hat{Prm}_S$. Hence, such s induces a formula φ_s in which g_{φ_s} and χ_{φ_s} are defined according to $\mathcal{I}$.

Using Lem. 8, given a simple TIP formula φ, we may further encode the set of all simple TIP formulas that are $\sqsubseteq_\Gamma \varphi$, respectively $\sqsupseteq_\Gamma \varphi$, by the following PA formulas over C:

$$Ex^{\sqsubseteq_\Gamma \varphi} = \exists \bar{u}. \bigvee_{\substack{t \in \hat{T} \\ t \sqsubseteq_\Gamma g_\varphi}} R_t \wedge \bigwedge_{r \in T \cup \hat{Prm}_S} \left(\chi_\varphi(r) \leq Q_r - \sum_{\substack{h \in T \cup \hat{Prm}_S \\ h \sqsubseteq_\Gamma r}} u_{(h,r)} + \sum_{\substack{h \in T \cup \hat{Prm}_S \\ r \sqsubseteq_\Gamma h}} u_{(r,h)} \right)$$

$$Ex^{\sqsupseteq_\Gamma \varphi} = \exists \bar{u}. \bigvee_{\substack{t \in \hat{T} \\ g_\varphi \sqsubseteq_\Gamma t}} R_t \wedge \bigwedge_{r \in T \cup \hat{Prm}_S} \left(\chi_\varphi(r) \geq Q_r + \sum_{\substack{h \in T \cup \hat{Prm}_S \\ h \sqsubseteq_\Gamma r}} u_{(h,r)} - \sum_{\substack{h \in T \cup \hat{Prm}_S \\ r \sqsubseteq_\Gamma h}} u_{(r,h)} \right)$$

These definitions ensure that $s \models \rho \wedge Ex^{\sqsubseteq_\Gamma \varphi}$ if and only if s defines a simple TIP formula, φ_s, such that $\varphi_s \sqsubseteq_\Gamma \varphi$, and similarly for $Ex^{\sqsupseteq_\Gamma \varphi}$.

Therefore, the map of unexplored simple TIP formulas is given by the PA formula U defined as follows:

1. U is initialized to ρ, indicating that all simple TIP formulas are initially unexplored.
2. Whenever $\mathit{MarkExplored}_\preceq(\varphi)$ is called, U is updated by conjoining it with $\neg Ex^{\sqsubseteq_\Gamma \varphi}$.
3. Whenever $\mathit{MarkExplored}_\succeq(\varphi)$ is called, U is updated by conjoining it with $\neg Ex^{\sqsupseteq_\Gamma \varphi}$.

The procedures *HasUnexplored* and *GetUnexplored* are performed by checking the satisfiability of U. If it is unsatisfiable then all elements in Σ are explored. Otherwise, the model $s = (\mathbb{Z}, \mathcal{I})$ for which $s \models U$ defines an unexplored formula φ_s.

Chapter 7

Evaluation

We evaluate the approach by verifying several challenging threshold-based distributed protocols that use sophisticated thresholds: Bosco [39] (presented in Chapter 3), Hybrid Reliable Broadcast [40], and Byzantine Fast Paxos [23].

7.1 Protocols

Bosco

Bosco was explained in detail in Chapter 3. We verified it under 3 different resilience conditions. The condition $\mathbf{n} > 3\mathbf{t}$ is required in order to ensure correctness. The condition $\mathbf{n} > 5\mathbf{t}$ allows Bosco to guarantee that if there are no Byzantine processes and all processes have the same input value, then every processor would reach a decision in a single network step (*weakly one-step*). If $\mathbf{n} > 7\mathbf{t}$ then Bosco ensures that even if there are some faulty processes, when all non-faulty processes start with the same initial value, they would reach a decision within a single network step (*strongly one-step*). To evaluate our approach, we verify the safety and liveness (using the liveness to safety reduction presented in [30]) of Bosco.

Hybrid Reliable Broadcast

We consider the asynchronous reliable broadcast from [40], that is designed to tolerate $\mathbf{t}_b < \frac{\mathbf{n}}{3}$ Byzantine faulty processes. Hybrid reliable broadcast, in its extended version, tolerates four different types of faults (namely Byzantine faults, symmetric faults, clean crash, and crash faults), with associated constants denoted in Fig. 7.1 by f_b, f_s, f_c, and f_i, respectively. The protocol constitutes the core of the clock synchronization algorithm presented in [43]. Interestingly, the protocol is correct under several different threshold conditions [24]. Nevertheless, the threshold intersection properties are same in all cases, which confirms that we capture the essence of thresholds, independently of their arithmetic representation. We verify the safety and liveness of Hybrid Reliable Broadcast.

Byzantine Fast Paxos

Byzantine Fast Paxos [23] is a fast-learning [21, 22] Byzantine fault-tolerant consensus protocols for an asynchronous system equipped with a leader-election module; "fast-learning" means that under

favourable timing, a decision can be reached in a single round of communication (when each node is simultaneously a proposer, acceptor, and learner) despite q crash failures. Moreover, Byzantine Fast Paxos is optimal in the sense that, in general, no fast-learning protocol can improve the bounds on n [21]. Byzantine Fast Paxos is safe if at most t Byzantine failures occur in a system of n nodes where $n > 3t + 2q$ for some parameter $q \leq t$ (Lamport [23] assumes that at most m out of the t failures are Byzantine; we only consider the case $m = t$). We verified the safety of Byzantine Fast Paxos, i.e., agreement.

Note that Byzantine fault-tolerant fast-learning consensus protocols are notoriously tricky to develop and verify. For example, two such algorithms, Zyzzyva [17] and FaB [28], were recently revealed incorrect [1] despite having been published at major systems conferences. The verification of Fast Byzantine Paxos therefore shows that the methodology presented in this thesis is able to handle some of the most intricate distributed protocols.

7.2 Implementation

We implemented both algorithms described in Sec. 5.5, where for the computation of Γ-valid TIP formulas we used AIP (Sec. 5.4). $\text{AIP}_{\text{EAGER}}$ eagerly constructs Δ 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 Sec. 6.1. AIP_{LAZY} computes a subset of Δ 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.

7.3 Results

Fig. 7.1 lists the protocols we verified and the details of the evaluation. Each experiment was repeated 10 times, and we report the mean time (μ) and standard deviation (σ). The caption of the figure explains the presented information, and we discuss the results below.

7.3.1 Eagerly Calculating Δ_{min} with $\text{Aip}_{\text{Eager}}$

For all protocols, running AIP took less than 1 minute (column $\mathbf{t_C}$), and generated all Γ-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, Δ_{EAGER}, contains 3–5 formulas, compared to 39–79 before minimization.

Due to the method we use for checking Γ-validity (Sec. 5.3), the number of quantifiers in the TIP formulas that are checked by AIP does not affect the time needed to compute the full Δ. For example, Bosco under the Strongly One-step resilience condition contains Γ-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 than the simple variant of Bosco (where the number of quantifiers is at most 3) to find Δ. Interestingly, in this example the Γ-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 $\Delta_{\text{Eager}}^{\text{Bosco Strongly One-step}}$.

7.3.2 Lazily Calculating Δ_{int} with Aip_{Lazy}

With the lazy approach based on CTIs, the time for finding the set of TIP formulas, Δ_{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 Δ_{Lazy} has a significant benefit: it lets Ivy prove the protocol much faster compared to using Δ_{Eager}. Comparing $\mathbf{t_V}$ in $\text{Aip}_{\text{Eager}}$ vs. Aip_{Lazy} shows that when the former takes a minute, the latter takes a few seconds, and when the former times out after 1 hour, the latter terminates, usually in under 1 minute. Comparing the formulas of Δ_{Eager} and Δ_{Lazy} reveals the reason. While the FO translation of both yields EPR formulas, the formulas resulting from Δ_{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_{Lazy} is not sensitive to the largest number of quantifiers that may appear in a Γ-valid simple TIP formula. The downside is that Aip_{Lazy} performs many Ivy checks in order to compute the final Δ_{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, Δ_{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.

Protocol	T	Γ	AIP$_{\text{EAGER}}$						AIP$_{\text{LAZY}}$						
			V	**I**	**Q**	$\mathbf{t_C}$	$\mathbf{t_V}$	$\Delta^{\text{Protocol}}_{\text{Eager}}$	$\Delta^{\text{Protocol}}_{\text{Lazy}}$	**V**	**I**	**CTI**	**Q**	$\mathbf{t_I}$	$\mathbf{t_V}$
Bosco	t_1=**n**$-t$ t_2=$\frac{\mathbf{n}+3t+1}{2}$ t_3=$\frac{\mathbf{n}-t+1}{2}$	**n** $> 3t$ $\lvert f\rvert \leq t$	$\frac{23}{39}$	$\frac{21}{1216}$	6	$3s$	$\mu(12s)$ $\sigma(4s)$	$g_1(f^c)$ $\forall x{:}g_1, y{:}g_1, z{:}g_2.g_3(x\cap y\cap z)$ $\forall x{:}g_1, y{:}g_2, z{:}g_3.x\cap y\cap z\neq\emptyset$	$g_1(f^c)$ $\forall x{:}g_1, y{:}g_2.g_3(x\cap y\cap f^c)$ $\forall x{:}g_2, y{:}g_3.x\cap y\cap f^c\neq\emptyset$	24	6	18	2	$\mu(3m)$ $\sigma(1m)$	$\mu(4s)$ $\sigma(0.4s)$
Bosco Weakly One-step	〃	**n** $> 5t$ $\lvert f\rvert \leq t$	$\frac{16}{51}$	$\frac{24}{1204}$	6	$3s$	$\mu(13m)$ $\sigma(14m)$	$\Delta^{\text{Bosco}}_{\text{EAGER}}$ $\forall x{:}g_1.g_2(x)$	$\Delta^{\text{Bosco}}_{\text{LAZY}}$ $\forall x{:}g_1.g_2(x)$	32	7	19	2	$\mu(13m)$ $\sigma(9m)$	$\mu(9s)$ $\sigma(2s)$
Bosco Strongly One-step	〃	**n** $> 7t$ $\lvert f\rvert \leq t$	$\frac{26}{63}$	$\frac{24}{2407}$	8	$8s$	T.O.	$\Delta^{\text{Bosco}}_{\text{EAGER}}$ $\forall x{:}g_1, y{:}g_1.g_2(x\cap y)$	$\Delta^{\text{Bosco}}_{\text{LAZY}}$ $\forall x{:}g_1.g_2(x\cap f^c)$	34	9	20	2	$\mu(23m)$ $\sigma(8m)$	$\mu(16s)$ $\sigma(13s)$
Hybrid Reliable Broadcast	t_1=t_a+t_s+1 t_2=**n**$-t_c-t_a$ $-t_s-t_i$	**n** $> t_c+3t_a+$ $2t_s+2t_i$ $\lvert f_x\rvert \leq t_x$ $f_x\cap f_y=\emptyset$ for $x\neq y$ $x,y\in\{a,c,i,s\}$	$\frac{25}{63}$	$\frac{34}{1877}$	2	$37s$	$\mu(35s)$ $\sigma(0.3s)$	$g_2(f^c_c\cap f^c_a\cap f^c_s\cap f^c_i)$ $\forall x{:}g_1.x\cap f^c_a\cap f^c_s\neq\emptyset$ $\forall x{:}g_2.g_1(x\cap f^c_a\cap f^c_i)$	$\Delta^{\text{Hybrid Reliable Broadcast}}_{\text{EAGER}}$	63	15	45	1	$\mu(15m)$ $\sigma(1.5m)$	$\mu(43s)$ $\sigma(1s)$
Byzantine Fast Paxos	t_1=**n**$-t$ t_2=**n**$-q$ t_3=**n**$-2t-q$ t_4=t+1	**n** $> 2q+3t$ $t\geq q$ $q\geq 0$ $\lvert b\rvert \leq t$	$\frac{22}{79}$	$\frac{44}{3695}$	6	$6s$	T.O.	$g_1(b^c)$ $\forall x{:}g_2.g_1(x)$ $\forall x{:}g_1, y{:}g_4.x\cap y\neq\emptyset$ $\forall x{:}g_2, y{:}g_3.g_4(x\cap y)$ $\forall x{:}g_1, y{:}g_1, z{:}g_2.g_3(x\cap y\cap z)$	$g_1(b^c)$ $\forall x{:}g_2.g_1(x)$ $\forall x{:}g_3.x\neq\emptyset$ $\forall x{:}g_4.x\neq\emptyset$ $\forall x{:}g_1.g_3(x\cap b^c)$ $\forall x{:}g_1, y{:}g_1.g_4(x\cap y)$	44	11	19	2	$\mu(36m)$ $\sigma(21m)$	$\mu(28m)$ $\sigma(19m)$

Figure 7.1: Protocols verified using our technique. For each protocol, T is the set of thresholds and Γ is the resilience condition. AIP$_{\text{EAGER}}$ lists metrics for the procedure of finding all Γ-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 $\Delta^{\text{Protocol}}_{\text{Eager}}$, where g_i denotes $g_{\geq t_i}$. In addition to the run times, **V** shows $\frac{c}{v}$, where c is the number of Γ-valid simple formulas that were checked using the BAPA solver (CVC4), and v is the total number of Γ-valid simple formulas. Namely $v-c$ simple formulas were inferred to be valid via subsumption. **I** reports the analogous metric for Γ-invalid simple formulas. Finally, **Q** reports the maximal number of quantifiers considered (for which all formulas were Γ-invalid). AIP$_{\text{LAZY}}$ lists metrics for the procedure of finding a set of Γ-valid TIP formulas sufficient to prove the protocol based on counterexamples. The resulting set is listed in $\Delta^{\text{Protocol}}_{\text{Lazy}}$, and $\mathbf{t_I}$ lists the total Ivy runtime, with the standard deviation specified below. **V** (resp. **I**) lists the number of Γ-valid (resp. Γ-invalid) simple formulas considered before the final set was reached. **CTI** lists the number of counterexample iterations required, and **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 1 hour was reached.

Chapter 8

Related Work

Fully automatic verification of threshold-based protocols. Algorithms modeled as Threshold automata (TA) [14] have been studied in [16, 13], 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 unboundedly many rounds [3]. 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 [10, 8]. 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 $\frac{\mathbf{n}}{2}$ nodes—requires a proof by induction over $\mathbf{n}$. 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 thesis, 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., [44, 38, 27, 36, 12, 37] (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 [45].

Chapter 9

Conclusion

This thesis 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 and (2) sufficient to discharge the VCs. Soundness is automatically checked in (quantifier-free) BAPA, and the VCs are automatically discharged using EPR. Both logics are decidable, supported by mature solvers, and allow providing the user with an understandable counterexample in case verification fails.

We evaluate the approach by formally verifying the correctness (both safety and liveness) of intricate protocols, including notoriously tricky fast-learning consensus protocols such as Byzantine Fast Paxos. The experimental results show 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.

Bibliography

[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: International Joint Conference on Automated Reasoning. pp. 82–98. Springer (2016)

[3] Bertrand, N., Konnov, I., Lazic, M., Widder, J.: Verification of Randomized Distributed Algorithms under Round-Rigid Adversaries. HAL **hal-01925533** (Nov 2018), https://hal.inria.fr/hal-01925533

[4] Cooper, D.C.: Theorem proving in arithmetic without multiplication. Machine intelligence **7**(91-99), 300 (1972)

[5] De Moura, L., Bjrner, N.: Z3: An efficient SMT solver pp. 337–340 (2008)

[6] Dragoi, C., Henzinger, T.A., Veith, H., Widder, J., Zufferey, D.: A logic-based framework for verifying consensus algorithms. In: VMCAI. Lecture Notes in Computer Science, vol. 8318, pp. 161–181. Springer (2014)

[7] Drgoi, C., Henzinger, T.A., Zufferey, D.: PSync: A partially synchronous language for fault-tolerant distributed algorithms **51**(1), 400–415 (2016)

[8] Dutertre, B., Jovanović, D., Navas, J.A.: Verification of fault-tolerant protocols with sally. In: Dutle, A., Muñoz, C., Narkawicz, A. (eds.) NASA Formal Methods. pp. 113–120. Springer International Publishing, Cham (2018)

[9] Ge, Y., De Moura, L.: Complete instantiation for quantified formulas in satisfiabiliby modulo theories. In: Computer Aided Verification. pp. 306–320. Springer (2009)

[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. pp. 599–613. PLDI '16, 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, October 4-7, 2015. pp. 1–17 (2015). https://doi.org/10.1145/2815400.2815428, https://doi.org/10.1145/2815400.2815428

[13] Konnov, I., Lazic, M., Veith, H., Widder, J.: Para^2: Parameterized path reduction, acceleration, and SMT for reachability in threshold-guarded distributed algorithms. Formal Methods in System Design **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. Information and Computation **252**, 95–109 (2017)

[15] Konnov, I., Widder, J.: Bymc: Byzantine model checker. In: ISoLA (3). LNCS, vol. 11246, pp. 327–342. Springer (2018)

[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, January 18-20, 2017. pp. 719–734 (2017)

[17] Kotla, R., Alvisi, L., Dahlin, M., Clement, A., Wong, E.: Zyzzyva: Speculative byzantine fault tolerance **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.C.: An algorithm for deciding BAPA: boolean algebra with presburger arithmetic. In: CADE. Lecture Notes in Computer Science, vol. 3632, pp. 260–277. Springer (2005)

[20] Lamport, L.: The Part-time Parliament **16**(2), 133–169 (1998-05). https://doi.org/10.1145/279227.279229

[21] Lamport, L.: Lower Bounds for Asynchronous Consensus. In: Future Directions in Distributed Computing, pp. 22–23. Lecture Notes in Computer Science, Springer, Berlin, Heidelberg (2003)

[22] Lamport, L.: Lower bounds for asynchronous consensus **19**(2), 104–125 (2006)

[23] Lamport, L.: Fast byzantine paxos (Nov 17 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), http://forsyte.at/wp-content/uploads/opodis17.pdf, (to appear)

[25] Lewis, H.R.: Complexity results for classes of quantificational formulas **21**(3), 317–353 (1980)

[26] Liffiton, M.H., Previti, A., Malik, A., Marques-Silva, J.: Fast, flexible mus enumeration. Constraints **21**(2), 223–250 (Apr 2016)

[27] Liu, Y.A., Stoller, S.D., Lin, B.: From clarity to efficiency for distributed algorithms. ACM Trans. Program. Lang. Syst. **39**(3), 12:1–12:41 (2017). https://doi.org/10.1145/2994595, https://doi.org/10.1145/2994595

[28] Martin, J.P., Alvisi, L.: Fast byzantine consensus **3**(3), 202–215 (2006)

[29] McMillan, K.L., Padon, O.: Deductive verification in decidable fragments with Ivy. In: SAS. Lecture Notes in Computer Science, vol. 11002, pp. 43–55. Springer (2018)

[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), 108:1–108:31 (2017)

[33] Padon, O., McMillan, K.L., Panda, A., Sagiv, M., Shoham, S.: Ivy: safety verification by interactive generalization. In: Krintz, C., Berger, E. (eds.) Proceedings of the 37th ACM SIGPLAN Conference on Programming Language Design and Implementation, PLDI 2016, Santa Barbara, CA, USA, June 13-17, 2016. pp. 614–630. ACM (2016)

[34] Piskac, R.: Decision procedures for program synthesis and verification (2011). https://doi.org/10.5075/epfl-thesis-5220, http://infoscience.epfl.ch/record/168994

[35] Piskac, R., de Moura, L., Bjrner, N.: Deciding effectively propositional logic using DPLL and substitution sets **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, https://doi.org/10.14279/tuj.eceasst.72.1013

[37] Rahli, V., Vukotic, I., Völp, M., Veríssimo, P.J.E.: Velisarios: Byzantine fault-tolerant protocols powered by coq. In: Programming Languages and Systems - 27th European Symposium on Programming, ESOP 2018, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2018, Thessaloniki, Greece, April 14-20, 2018, Proceedings. pp. 619–650 (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). https://doi.org/10.1145/3158116, https://doi.org/10.1145/3158116

[39] Song, Y.J., van Renesse, R.: Bosco: One-step Byzantine asynchronous consensus. In: DISC. LNCS, vol. 5218, pp. 438–450 (2008)

[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 bounded model checking. In: Tools and Algorithms for the Construction and Analysis of Systems - 25th International Conference, TACAS 2019, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2019, Prague, Czech Republic, April 6-11, 2019, Proceedings, Part II. pp. 357–374 (2019). https://doi.org/10.1007/978-3-030-17465-1_20, https://doi.org/10.1007/978-3-030-17465-1_20, a pre-print including the proofs is available at: https://hal.inria.fr/hal-01925653

[42] Taube, M., Losa, G., McMillan, K.L., Padon, O., Sagiv, M., Shoham, S., Wilcox, J.R., Woos, D.: Modularity for decidability of deductive verification with applications to distributed systems. In: PLDI. pp. 662–677. ACM (2018)

[43] Widder, J., Schmid, U.: Booting clock synchronization in partially synchronous systems with hybrid process and link failures. Dist. Comp. **20**(2), 115–140 (2007)

[44] Wilcox, J.R., Woos, D., Panchekha, P., Tatlock, Z., Wang, X., Ernst, M.D., Anderson, T.E.: 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, June 15-17, 2015. pp. 357–368 (2015). https://doi.org/10.1145/2737924.2737958, https://doi.org/10.1145/2737924.2737958

[45] 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, January 20-22, 2016. pp. 154–165 (2016). https://doi.org/10.1145/2854065.2854081, https://doi.org/10.1145/2854065.2854081

תקציר

אימות של אלגוריתמים מבוזרים מבוססי תנאי-סף על ידי פירוק ללוגיקות כריעות

עידן ברקוביץ'
תואר מוסמך (.B.Sc)
בית הספר למדעי המחשב
אוניברסיטת תל-אביב

אימות של פרוטוקולים מבוזרים אשר סופגים התנהגויות בלתי צפויות של תהליכים הוא משימה קשה ביותר. לעתים קרובות, בפרוטוקולים אלה נעשה שימוש בתנאי-סף על גדלי קבוצות אשר מוגדרים הן בקוד הפרוטוקול והן בהוכחת נכונותו. למשל, תהליך יכול לבצע פעולה רק אם קיבל אישור לפחות ממחצית מעמיתיו. אימות פרוטוקולים מבוססי תנאי-סף מאתגר במיוחד מכיוון שהוא כולל שני סוגים של טיעונים: טיעונים מסדר ראשון העוסקים בכמות בלתי מוגבלת של מצבים בהם יכול להימצא הפרוטוקול, יחד עם טיעונים הנוגעים לקבוצות וגודלן.

בעבודה זו אנו מפתחים שיטה לחלוקת משימת האימות של פרוטוקולים מסוג זה לשתי לוגיקות כריעות: EPR ו-BAPA. התובנה העיקרית שלנו היא שפרוטוקולים כאלה משתמשים בתנאי-סף בצורה מוגבלת כאמצעי להשגת מאפיינים מסוימים של "חיתוך" בין הקבוצות. אנו מגדירים שפה לביטוי מאפיינים כאלה ומציגים שני תרגומים של טענות בשפה זו: ל-EPR ול-BAPA. התרגום ל-EPR מאפשר לאמת את הפרוטוקול תוך הנחת מאפיינים מסוג זה, והתרגום ל-BAPA מאפשר לאמת את נכונות המאפיינים הללו בהנחת האקסיומות המוגדרות בפרוטוקול.

אנו מפתחים אלגוריתם למציאה אוטומטית של המאפיינים הדרושים לאימות פרוטוקול נתון, מה שמאפשר תהליך אימות אוטומטי לחלוטין. באמצעות טכניקה זו אימתנו מספר פרוטוקולים מאתגרים, כולל הסכמת קונצנזוס בצעד אחד עמידה בפני תהליכים תקולים (Byzantine one-step consensus), שידור אמין היברידי (Hybrid reliable broadcast) ו-Paxos מהיר עמיד בפני תהליכים תקולים (Fast byzantine paxos).

הפקולטה למדעים
מדויקים ע"ש ריימונד
ובברלי סאקלר
אוניברסיטת תל אביב

בית הספר למדעי המחשב ע"ש בלבטניק

אימות של אלגוריתמים מבוזרים מבוססי תנאי-סף על ידי פירוק ללוגיקות בריעות

מאת

עידן ברקוביץ'

עבודת המחקר בוצעה בהנחייתה של

פרופ' שרון שוהם

חיבור זה הוגש כחלק מהדרישות לקבלת תואר מוסמך אוניברסיטה (.M.Sc)

2020