Towards Verifying Security Policies for Infinite-state Systems

Quentin Peyras[1,3], Ghada Gharbi[1], and Souheib Baarir[2]

[1] LRE, EPITA
[2] LIP 6, Sorbonne University
[3] IRIT CNRS UPS, Université de Toulouse
quentin.peyras@irit.fr, ghada.gharbi@epita.fr
souheib.baarir@lip6.fr

Abstract. Non-interference ensures no unauthorized data leaks during system execution. Verifying security policies is complex, requiring analysis of multiple execution paths. Hyperproperties provide a framework to describe security policies like non-interference. However, existing methods like HyperLTL are limited to finite-state models. This paper introduces a case study illustrating the use of HyperFOLTL, designed for infinite-state systems, and presents a formal approach to verify security policies in such systems.

Keywords: Non-Interference · Hyperproperties · HyperFOLTL · First-Order Logic · Distributed Systems.

1 Introduction

Distributed systems are crucial in modern computing, where security is vital for data integrity and privacy. Mechanisms such as non-interference and confidentiality policies are key defenses against potential threats. To ensure system correctness and security, formal methods are often used. While safety and liveness can be expressed using temporal logics like LTL [25], security policies require comparing multiple executions, leading to the introduction of HyperLTL for specifying hyperproperties. However, the complexity of distributed systems, with their dynamic interactions, necessitates more expressive logics. HyperFOLTL extends HyperLTL [6] with First-Order (FO) quantifiers, enabling the specification of hyperproperties in infinite-state systems.

In this paper, we propose to use HyperFOLTL logic as a potential solution for expressing security policies and a formal definition of security policy satisfiability problem, defined in terms of HyperFOLTL formulas. The paper is organized as follows. Section II introduces a case stydy, the leader election protocol. Section III outlines temporal logics for expressing hyperproperties. Section IV details our approach. We will study the formal proof of security property's satisfiability (i.e., non-interference). The proof is conducted on the case study. Section V formally defines the concept of a linking invariant, used to reason about multiple system traces. Section VI describes related works and finally we will conclude and give future works.

2 Problem statement

Non-interference, first introduced by Goguen and Meseguer [17], is a key security property in distributed systems, ensuring data integrity and confidentiality across different security domains. It prevents sensitive information from leaking to unauthorized users or processes.

Leader election protocol Let's explore a motivating example that demonstrates a non-interference security property in the context of a distributed system: the leader election protocol [4]. This will showcase our approach and highlight the challenges of verifying security policies.

The leader election protocol operates in a unidirectional ring network, where each node communicates with its neighbor in one direction. The goal is to elect the node with the largest identifier as the leader. Each node maintains a list of identifiers (represented by **msgs**), initially containing only its own, and asynchronously sends its identifier to its successor, adding any larger identifiers to the list. A node becomes the leader if it receives its own identifier back.
Key assumptions include asynchronous message transmission, a reliable network with no data loss, and the tolerance of transmission delays. Node identifiers are integers, allowing for total ordering, simplifying comparisons during the election process. It is important to note that the example being analyzed is modeled as an infinite-state system solely based on the number of nodes it comprises.
A key security concern arises from the assumption that an attacker can observe message sequences without accessing their content. This raises the question of whether the attacker can deduce the leader's identity through message patterns alone. Given the leader's pivotal role in decision-making and resource control, such disclosure could have serious consequences in distributed systems like distributed databases and file systems, or cloud services.

3 Formalizing Hyperproperties

In this section, we present the formalism of hyperproperties.

3.1 For finite-state systems

In this part, we consider the formalism of labeled transition systems, a standard method for modeling distributed systems [2, 16, 17, 27]. Labeled transition systems allows the specification of safety and liveness properties using temporal logics such as Linear Temporal Logic (LTL). However, when it comes to reasoning about security policies, a more sophisticated class of properties, known as hyperproperties, is necessary. Unlike trace properties, expressible in LTL, and branching properties, expressible in CTL, hyperproperties requires the comparison and linkage of multiple distinct traces. The primary logic used to specify hyperproperties is HyperLTL [5,6], achieved by extending LTL with path quantifiers that bind each atomic proposition (the set of atomic propositions is denoted

as **AP**) to a specific trace. This extension allows for the comparison of traces by connecting atomic propositions across different paths using LTL operators.

The general **HyperLTL** satisfiability problem is undecidable. Nevertheless, it is possible to define a specific fragment of **HyperLTL**, called $\mathbf{HyperLTL}_2$, that is both decidable and capable of expressing all the relevant security policies as discussed in [5].

3.2 Generalization to infinite-state systems

When specifying concurrent and infinite-state systems, our emphasis lies in defining initial states and transitions using first-order logic. This approach, which is standard for specifying and verifying such systems [7, 18–20, 23, 24], necessitates augmenting first-order logic with a unique prime operator. This operator enables referencing the values of relations after a transition has been executed.

Definition 1 (Prime operator). *If r is a relation symbol, r' represents the value of r at the subsequent time instant (using LTL notation, $r'(\boldsymbol{x}) \equiv \mathbf{X}(r(\boldsymbol{x}))$). For a signature Σ, the signature of the primed relations and functions derived from this Σ is denoted as Σ'.*

Furthermore, a primed first-order (FO) formula on Σ refers to an FO formula on the signature $\Sigma \cup \Sigma'$. Such a formula is interpreted using a pair (m, m') of FO structures over Σ : one interprets the non-primed symbols, and the other interprets the primed symbols, thus defining the temporal evolution of the system between two successive states. If this interpretation satisfies the formula ϕ, we denote it as $m, m' \vDash \phi$.

Then, systems can be specified using FO formulas [22].

Definition 2 (FO transition systems). *A FO transition system is a tuple* **Spec** $= (\Sigma, \iota, \tau)$ *where: Σ is an FO signature; ι is an FO formula defining initials states; τ is a primed FO formula defining possible transitions. Let t be a sequence of FO structures sharing the same domain D, we say that t is a trace of* **Spec** *if $t_0 \vDash \iota$ and $\forall i \in \mathbb{N}$ we have $t_i, t_{i+1} \vDash \tau$.*

Now that systems can be specified, a logic for reasoning about them is required. However, **HyperLTL** only supports finite-state systems. To handle infinite-state systems, one solution is to extend **HyperLTL** with FO quantifiers [13] [10] [12], leading to HyperFOLTL.

Definition 3 (HyperFOLTL). *HyperFOLTL syntax is given by adding FO quantifiers to the syntax of* **HyperLTL**, *the syntax of a HyperFOLTL formula ψ follows the grammar:*

- $\psi ::= \phi \mid \exists \pi \cdot \psi \mid \forall \pi \cdot \psi$;
- $\phi ::= r_\pi(\boldsymbol{x})^4 \ (r \in \Sigma) \mid \neg\phi \mid \phi \vee \phi \mid \mathbf{X}\,\phi \mid \phi\,\mathbf{U}\,\phi \mid \forall x \cdot \phi$.

[4] r_π denotes the relation r interpreted within the trace π. In order to simplify HyperFOLTL formulas, we sometimes write r_i to denotes r_{π_i}.

Let T be a set of traces sharing the same domain D, Π a set of trace variables, $\mathcal{C}_\Pi : \Pi \rightharpoonup T$ a partial assignment of those variables to FO traces, $\mathcal{V}$ a set of variables and $\mathcal{C}$ an assignment of those variables. The semantics of ψ over T is defined as follows:

- $D, T, \mathcal{C}_\Pi, \mathcal{C}, n \vDash \forall \pi \cdot \psi$ *iff for all* $t' \in T$ *we have* $D, T, \mathcal{C}_\Pi[\pi \mapsto t'], \mathcal{C}, n \vDash \psi$*;*
- $D, T, \mathcal{C}_\Pi, \mathcal{C}, n \vDash r_\pi(\boldsymbol{x})$ *iff* $\mathcal{C}_\Pi(\pi)$ *satisfies* $r(\boldsymbol{x})$ *with assignment* $\mathcal{C}$*;*
- $D, T, \mathcal{C}_\Pi, \mathcal{C}, n \vDash \forall x \cdot \psi$ *iff for all* $d \in D$ *we have* $D, T, \mathcal{C}_\Pi, \mathcal{C}[x \mapsto d], n \vDash \psi$*;*
- $\neg, \vee, \mathbf{X}, \mathbf{U}$ *connectives follows the inductive definition of LTL semantics.*
- $\mathbf{F}$ *and* $\mathbf{G}$ *connectives are defined and used as in LTL.*

We can now formally define what it means for a specification to satisfy an HyperFOLTL property.

Definition 4 (Satisfaction). *Let* **Spec** *be an FO transition system and ϕ an HyperFOLTL formula, then we say that* $\mathbf{Spec} \vDash \phi$ *if, for any domain D and for T, the set of traces of* **Spec** *with domain D, we have:* $D, T, [], [], 0 \vDash \phi$.

Remark 1 (Encoding of HyperFOLTL Satisfaction). The problem of checking if an FO transition system satisfy a HyperFOLTL formula can be reduced to the problem of satisfiability of an FOLTL formula [12]. Howewer, the obtained FOLTL formula does not describe an FO transition system. Consequently, invariant-based methods are insufficient for proving unsatisfiability.

4 Application to the leader election protocol

We'll now apply this formalism to our example, the leader election protocol.

4.1 Formal specification of the leader election protocol

Now, we can define the formal specification of the leader election protocol as an FO transition system (Σ, ι, τ). The signature Σ defines the components of the protocol, represented as sorts, relations and function symbols:

- Node is the sort of all nodes in the ring and Id is the sort of identifiers;
- $\mathrm{id} : \mathrm{Node} \to \mathrm{Id}$ is the function associating each node with its identifier;
- `succ` : $\mathrm{Node} \times \mathrm{Node}$ is the relation indicating that two nodes are successors in the ring;
- **msgs** : $\mathrm{Node} \times \mathrm{Id}$ is the relation indicating that a node has an identifier in its message list.

Then, ι is the formula specifying the initial states of the protocol and consists in two parts. First, we define general axioms denoted by the formula **Ring**, which specify that the system comprises a ring-shaped network whose nodes have unique identifiers, as defined in [23]. Then, we specify the rest of the initial states, i.e., that any **msgs** node list contains only the node's own identifier.

$$\mathbf{Init} := \forall x : \mathrm{Node}, i : \mathrm{Id} \cdot \mathbf{msgs}(x, i) \Leftrightarrow i = \mathrm{id}(x) \wedge \mathbf{Ring}$$

Finally, τ is the transition formula for the leader election protocol. In this protocol, the only possible operation is a node sending an identifier contained in its **msgs** list to its successor. The successor adds this identifier to its own **msgs** list if it is greater than or equal to its own identifier. Additionally, we need to specify what does not change using two formulas, $\mathsf{unchanged}_{\mathbf{msgs}}(s, r : \text{Node}, m : \text{Id})$ and $\mathsf{unchanged}_{\mathtt{succ}}$, referred to as frame conditions:

$$\mathsf{unchanged}_{\mathbf{msgs}}(s, r : \text{Node}, m : \text{Id}) := m < \text{id}(r) \Rightarrow (\mathbf{msgs}'(r, m) \Leftrightarrow \mathbf{msgs}(r, m))$$
$$\wedge\Big(\forall x : \text{Node}, i : \text{Id} \cdot (i \neq m \vee (x \neq s \wedge x \neq r)) \Rightarrow (\mathbf{msgs}(x, i) \Leftrightarrow \mathbf{msgs}'(x, i))\Big)$$
$$\mathsf{unchanged}_{\mathtt{succ}} := \forall x, y : \text{Node} \cdot \mathtt{succ}(x, y) \Leftrightarrow \mathtt{succ}'(x, y)$$

To simplify the specification of the desired hyperproperty, we introduce a formula: $\mathbf{SDef}(s) := \forall x : \text{Node} \cdot (\mathbf{Send}(x) \Leftrightarrow x = s)$ and a relation **Send** to label nodes performing the sending operation. Thus, the transition formula is:

$$\mathbf{Trans} := \exists s, r : \text{Node}, m : \text{id} \cdot \mathbf{SDef}(s) \wedge \mathtt{succ}(s, r) \wedge (m \geq \text{id}(r) \Rightarrow \mathbf{msgs}'(r, m))$$
$$\wedge \mathbf{msgs}(s, m) \wedge \neg \mathbf{msgs}'(s, m) \wedge \mathsf{unchanged}_{\mathbf{msgs}}(s, r, m) \wedge \mathsf{unchanged}_{\mathtt{succ}}$$

Our goal, in the context of the leader election protocol, is to determine whether an attacker can infer node identifiers based on observed message exchanges. Formally, we aim to verify that, for every possible identifier distribution and message-sending sequence, a trace exists that is consistent with both, which can be expressed as the following HyperFOLTL formula: [5]:

$$\forall \pi_1, \pi_2 \cdot \exists \pi_3 \cdot (\forall n : \text{Node} \cdot \text{id}_2(n) = \text{id}_3(n)) \wedge$$
$$\mathbf{G}(\forall n : \text{Node} \cdot \mathbf{Send}_1(n) \Leftrightarrow \mathbf{Send}_3(n))$$

This property asserts that for any two traces of the protocol, there exists a third trace that aligns with the first trace in terms of the order of sent operations and with the second trace in terms of identifiers. This characteristic falls under the category of non-interference properties, signifying that the identifiers distribution has no consequence on determining which node sends a message.

4.2 Detecting a violation of the property

The leader election protocol, as specified in the previous section, violates the intended non-interference policy. This occurs because a node discards messages if the received identifier is lower than its own. To grasp the issue, consider a two-node ring where the following sequence of messages is exchanged: Node 1 sends to Node 2; Node 2 sends to Node 1; Node 2 sends to Node 1. In this scenario, it can be deduced that at least one of the two messages sent by Node 2 contains the identifier of Node 1. Consequently, Node 1's identifier is greater than Node 2's, leading to Node 1 being elected as the leader by the end of the sequence.
Now that we have identified the violation of this property, our focus shifts to

[5] we recall that in such a formula, r_i is used to represent the relation r interpreted in the trace π_i.

detecting such violations in a general context. As observed, the property is infringed even in a network comprising only two nodes. Additionally, the model-checking process of HyperFOLTL on a bounded domain can be simplified by transforming it into the model-checking of HyperLTL. This transformation involves unfolding the first-order quantifiers within the specified bounded domain. Since non-interference properties for finite-state systems fall within the decidable $\mathbf{HyperLTL}_2$ fragment of HyperLTL [5], we can effectively assess violations of this property within the given bounded domain. For the leader election protocol, this domain comprises only 2 nodes. This method effectively identifies counter-examples, but there is no general way to compute a completeness threshold. In other words, the absence of a counter-example does not rule out its existence in a larger scope. For instance, in the leader election protocol, if no counter-examples are found with fewer than 3 nodes, they may still exist with 4 or more nodes.

4.3 Proving the satisfaction of the property

Having identified a violation of the non-interference policy within the leader election protocol, a straightforward correction can be proposed to align with the non-interference policy. This correction is simple: allowing any node to send a message containing its own identifier at any given time. With this adjustment, all nodes can transmit messages continuously, ensuring that no discernible information can be deduced from these transmissions. Thus, the new transition formula is:

$$\begin{aligned}&\mathbf{Trans} := \exists s, r : \mathrm{Node}, m : \mathrm{id} \cdot (\mathbf{Sending}(s, r, m) \vee \mathbf{OwnSending}(s, r, m))\\ &\wedge \mathsf{unchanged}_{\mathbf{msgs}}(s, r, m) \wedge \mathsf{unchanged}_{\mathtt{succ}}\end{aligned}$$

Where $\mathbf{Sending}(s, r, m)$ corresponds to the initial sending operation and $\mathbf{OwnSending}(s, r, m)$ corresponds to the sending of its own identifier:

$$\begin{aligned}&\mathbf{Sending}(s, r, m) := \mathbf{SDef}(s) \wedge \mathtt{succ}(s, r) \wedge \mathbf{msgs}(s, m)\\ &\wedge \neg \mathbf{msgs}'(s, m) \wedge (m \geq \mathrm{id}(r) \Rightarrow \mathbf{msgs}'(r, m))\\ &\mathbf{OwnSending}(s, r, m) := \mathbf{SDef}(s) \wedge \mathtt{succ}(s, r) \wedge m = \mathrm{id}(s)\\ &\wedge (\mathbf{msgs}(s, m) \Leftrightarrow \mathbf{msgs}'(s, m)) \wedge (m \geq \mathrm{id}(r) \Rightarrow \mathbf{msgs}'(r, m))\end{aligned}$$

Remark 2. For trace properties, correcting a protocol or a system is done by preventing some error traces to occur. However, the previous modification corrects the leader election protocol by enlarging the set of possible trace, demonstrating that non-interference is not a trace property.

While it might be apparent that the property is now satisfied, a question arises: how can this be rigorously proven in this context? The proof can be established through the utilization of a novel method relying on an FO formula that we call *linking invariant* which establishes relationships between the states and transitions of multiple traces in a system. Section 5 provides a formal definition of the concept of a linking invariant.

In the following, we will illustrate the application of this concept to three traces to satisfy the HyperFOLTL formula in the context of the leader election protocol. A linking invariant is used to construct states of the third trace π_3 from the states of π_1 and π_2. This linking invariant is presented as a first-order formula on relations describing the states of all three traces. In the case of the modified leader election, the linking invariant is the following:

$$\begin{aligned}&\mathrm{L} := \forall n : \mathrm{Node} \cdot \mathrm{id}_2(n) = \mathrm{id}_3(n) \wedge (\mathbf{Send}_1(n) \Leftrightarrow \mathbf{Send}_3(n))\\ &\wedge \forall n_1, n_2 : \mathrm{Node} \cdot (\mathtt{succ}_1(n_1, n_2) \Leftrightarrow \mathtt{succ}_2(n_1, n_2) \Leftrightarrow \mathtt{succ}_3(n_1, n_2))\end{aligned}$$

In the following, Σ_1, Σ_2, and Σ_3 represent the symbols' signatures for respectively the first, second, and third traces referred by the HyperFOLTL formula. To proceed, the linking invariant L must satisfy the following conditions: (1) There exists an initial linking condition, L_i, s. t. $L_i \vDash L$ and for any possible valuation of the signature corresponding to universally quantified traces, there exists a valuation of the remaining relation that satisfies the formula; In simpler terms, the initial condition L_i must be robust enough to guarantee L under all possible initial settings for the system's traces. (2) $L_i \vDash \phi$ where ϕ is the FO formula obtained after removing the path quantifiers and the **G** operator; (3) For any couple of models $(\mathcal{M}, \mathcal{M}')$ satisfying L and the transition formula, there is a model $\mathcal{M}_L$ of L with same valuation than $\mathcal{M}'$ outside of Σ_3. This ensures that the linking invariant L can be maintained consistently across different system states and transitions, with $\mathcal{M}_L$ aligning with $\mathcal{M}'$ for all variables not in Σ_3.

The challenge lies in verifying the previous conditions. Condition (1) ensures that for any initial state corresponding to universally quantified traces, we can find initial states for the existentially quantified traces such that the Linking Invariant is satisfied at the start. In the leader election protocol, the initial linking condition states that we copy the initial state of trace 2 except for the **Send** relation that is taken from trace 1. Since the relation copied from trace 1 and trace 2 are distinct, the satisfaction of condition (1) is trivial. Then, condition (2) constitutes a standard first-order logic problem, solvable through a decidable fragment of FO or a sound reasoning method. Condition (3) ensures that from any combination of states satisfying the linking invariant, whatever the universally quantified traces do, there is a way to satisfy the Linking Invariant by choosing well the next states for existentially quantified traces. By induction, conditions (1) and (3) allow to prove the HyperFOLTL formula: Q_1 pi_1, ..., Q_n pi_n. G(L). Our proposed approach for verifying condition (3) assumes that any transition is limited to modifying the values of relations within a bounded set of elements in the domain, denoted as $y_1 \dots y_k$. This requirement is typically satisfied by distributed systems [22]. Given this assumption, verifying this condition can be simplified to a Quantified Boolean Formula (QBF) problem by examining the unfolding of the formula on the elements $y_1 \dots y_k$ that undergo modification.

For the leader election protocol, the modified relations are **Send** on s and **msgs** on (s, i) and (r, i). If, for any potential modification to these values for π_1 and π_2, it remains feasible to define values for π_3 while maintaining the linking invariant, it can be deduced that condition (3) is met. Let us define that

$\mathbf{Sending}_i$ (or $\mathbf{OwnSending}_i$) represents **Sending** (or **OwnSending**) with all instances of any relation symbol r replaced by r_i, and $\mathbf{InvPreservation}(s, r)$ represents the fact that the linking invariant is satisfied on s and r before and after a transition. Hence, our proposed method gives the following formula:

$$\Big(\big(\mathbf{Sending}_1(s,r,m) \vee \mathbf{OwnSending}_1(s,r,m)\big)$$
$$\wedge \big(\mathbf{Sending}_2(s,r,m) \vee \mathbf{OwnSending}_2(s,r,m)\big)\Big)$$
$$\Rightarrow \Big(\big(\mathbf{Sending}_3(s,r,m) \vee \mathbf{OwnSending}_3(s,r,m)\big) \wedge \mathbf{InvPreservation}(s,r)\Big)$$
$$\mathbf{InvPreservation}(s,r) := \Big(\big(\bigwedge_{n\in\{s,r\}} \mathrm{id}_2(n) = \mathrm{id}_3(n) \wedge (\mathbf{Send}_1(n) \Leftrightarrow \mathbf{Send}_3(n))\big)$$
$$\wedge \big(\bigwedge_{n_1,n_2\in\{s,r\}} (\mathtt{succ}_1(n_1,n_2) \Leftrightarrow \mathtt{succ}_2(n_1,n_2) \Leftrightarrow \mathtt{succ}_3(n_1,n_2))\big)\Big)$$
$$\Rightarrow \Big(\big(\bigwedge_{n\in\{s,r\}} \mathrm{id}'_2(n) = \mathrm{id}'_3(n) \wedge (\mathbf{Send}'_1(n) \Leftrightarrow \mathbf{Send}'_3(n))\big)$$
$$\wedge \big(\bigwedge_{n_1,n_2\in\{s,r\}} (\mathtt{succ}'_1(n_1,n_2) \Leftrightarrow \mathtt{succ}'_2(n_1,n_2) \Leftrightarrow \mathtt{succ}'_3(n_1,n_2))\big)\Big)$$

To confirm condition (3) for the leader election protocol, one can solve the QBF problem using the preceding formula. This involves universal quantification of relations associated to traces π_1 and π_2, and for relations on trace π_3, existential quantification if primed and universal quantification otherwise.

5 Formalizing Linking Invariant

This section provides a formal definition of the linking invariant concept and presents a sketch of a proof for proving condition 3 to satisfy this latter.

Definition 5 (Linking invariant). *Let L and P be FO formulas on Σ^*[6] and $\mathrm{TS} = (\Sigma, \iota, \tau)$ be an FO transition system, L is said to be a linking invariant proving $\forall\pi_1, \ldots \pi_n, \exists\pi_{n+1}, \ldots \pi_m\, \mathbf{G}(P)$ for* TS *if:*

- *There exists L_i an FO formula on Σ^* such that:*
  - *$\forall$ s_1,s_2, …, s_n satisfying ι, $\exists$ s_{n+1}, …, s_m satisfying ι such that s_1,s_2, …, $s_m \vDash L_i$[7];*
  - *$L_i \vDash L$;*
- *$L \vDash P$.*
- *For all $s_1, s_2, \ldots, s_m, s'_1, \ldots, s'_n$ s.t. $s_1, \ldots, s_m \vDash L$ and $\forall 1 \leq i \leq n, s_i, s'_i \vDash \tau$, there id $s'_{n+1}, \ldots, s'_m$ s.t. $s'_1, \ldots, s'_m \vDash L$ and $\forall n+1 \leq i \leq m, s_i, s'_i \vDash \tau$.*

[6] Σ^* represents the signature obtained by duplicating relations in Σ for all trace quantifiers appearing in the HyperFOLTL formula we are trying to prove.

[7] The notation s_1,s_2, …, $s_m \vDash L_i$ is used to compose the FO structures for evaluating the formula so s_1 is used for symbols from Σ_1, s_2 for Σ_2, etc.

The above definition states that (1) an initial condition ensuring L holds at the start, (2) L guarantees the desired property P, and (3) L is preserved across transitions. Then, the following result can be proved from a simple induction.

Theorem 1 (Linking invariant). *If* TS *admits a linking invariant proving* $\forall\pi_1,\dots\pi_n,\exists\pi_{n+1},\dots\pi_m\,\mathbf{G}(P)$ *then:* $\text{TS}\vDash\forall\pi_1,\dots\pi_k,\exists\pi_{k+1},\dots\pi_n\,\mathbf{G}(P)$

Theorem 2 (Criteria for condition 3). *Let's assume that* τ *restricts changes of values to a finite set of terms*[8] *and that* L *is a universally quantified formula, then checking condition (3) can be reduced to a QBF problem.*

Proof. First, we define some notations. Let ϕ be an FO formula, then ϕ_i is the formula where all relation r of ϕ are replaced by r_i, denoting the relation r interpreted in the i-th trace. Analogously, ϕ' denotes ϕ where all its relation symbols r are replaced by r' representing the value of r at the next instant in time. Then, we say that $m_1,\dots,m_i,m'_1,\dots m'_j\vDash\phi$ if ϕ is true when interpreting all relation of the form r_k (resp. r'_k) in structure m_k (resp. m'_k). Then we say that $m_1,\dots,m_i,m'_1,\dots m'_j$ satisfy ϕ.

Finally, if E is a set of boolean variables, $Q_BE(Q\in\{\forall,\exists\})$ denotes the boolean quantifiers applied on all variables in E.

Let $m_1,\dots,m_n,m'_1,\dots,m'_k$ be a list of structures such that there is no structures $m'_{k+1},\dots m'_n$ satisfying $L\wedge\tau_1\wedge\ldots\wedge\tau_m\wedge L'$. So either :

1. $L\wedge\tau_1\wedge\ldots\tau_k$ is not satisfied by $m_1,\dots,m_n,m'_1,\dots,m'_k$;
2. for any $m'_{k+1},\dots m'_n$, $m_1,\dots,m_n,m'_1,\dots,m'_n$ do not satisfy $\tau_{k+1}\wedge\ldots\tau_n\wedge L'$.

If condition 1 is verified then we are considering structures that either does not satisfy the linking invariant or structures that are not valid successor considering our FO transition system. So, we can assume that condition 2 is verified. We define $Y'_i=\{r'_i(\boldsymbol{y})\in\mathcal{R}\mid\boldsymbol{y}\in\{y_1,\dots y_n\}^\star\}$, $Y'=\bigcup_{i>k}Y'_i$ and A the set of atoms (relation applied to terms) in $L\wedge\tau_1\wedge\ldots\wedge\tau_m\wedge L'$.

ϕ^B the boolean formula obtained by removing all FO quantifiers from ϕ. ϕ_D is the boolean formula obtained by unfolding all FO quantifiers on a finite domain D. In the following, D will denotes all constants and variables appearing in L and τ. If condition 2 is not satisfied, we have a counterexample to the QBF problem[9]: $P_{QBF}=\forall_BA\setminus Y'\exists_BY'\cdot L^B\wedge(\bigwedge_{1\le i\le k}\tau_i^B)\Rightarrow(\bigwedge_{k+1\le j\le n}\tau_j^B)\wedge L'_D$

Indeed, values for universally quantified variables can be taken from the structures $m_1,\dots,m_n,m'_1,\dots,m'_k$. Then, if the QBF problem could be solved by completing the remaining boolean variables, then we would be able to define the new values of the finite set of relations whose values change during the transition. Moreover, L'_D is verified and because it unfolds the universal quantifiers of L' on

[8] τ is of the form $\exists y_1,\dots,y_n\cdot\forall x_1,\dots,x_m\cdot x_1,\dots,x_m\notin F\Rightarrow NoChange(x_1,\dots,x_m)\wedge\Phi_F(y_1,\dots,y_n)$. Where $NoChange$ means that all relation on these variables keeps the same value after the transition.

[9] This construction can be improved to simplify the resulting QBF problem, but the proof of correctness becomes harder.

all combination for which values of relations have changed we can conclude that those new values satisfy L'. Then, it is possible to define $m'_{k+1}, \dots m'_n$ such that $m_1, \dots, m_n, m'_1, \dots, m'_n$ satisfy $\tau_{k+1} \wedge \dots \tau_n \wedge L'$, contradicting our hypothesis. We conclude that if P_{QBF} is true, condition (3) is satisfied.

6 Related work

Formal verification of security policy is an active area of research. Some works, such as [26], introduce type systems to ensure secure information flow in programming languages and systems. However, they face challenges in considering all possible executions, leading to undecidable problems and the need for approximations, reducing accuracy. Other works, like [15], focus on non-interference enforcement in real-time systems using timed automata, verify non-interference property based on different behavior notions: trace equivalence, reachability equivalence, etc. However, it is limited to finite-state systems. Approaches discussed in [1] and [14] also employ methodologies based on timed automata along with timed non-interference to capture interference-free systems, particularly those with high-frequency actions. Infinite-state system verification, as discussed in [3], relies on finite-state abstraction, with no guarantee of finding suitable abstractions. Specific methods, as outlined in [12], address security policy verification in multi-agent workflows, a specific subset of distributed systems, by encoding HyperFOLTL formulas into decidable fragments of FOLTL, for which satisfiability can be reduced to finite-state model-checking. Nevertheless, strict system requirements must be met to ensure that the problem encoding falls within a decidable fragment of FOLTL. Another approach, presented in [21], reduces the problem of verifying security policies in multi-agent workflows to the inference and verification of an inductive invariant but also demands strict system requirements. Meanwhile, methods introduced in [11] focus on addressing hyperproperties in infinite-state systems without relying on finite-state abstraction, primarily by disproving hyperproperties by identifying counter-examples rather than formally proving their satisfaction. Other works, such as [9] and [8], focuses on deductive verification using generalized version of Hoare logic for proving hyperproperties on programs. The notion of linking invariant generalizes some ideas used in those proof techniques to distributed systems.

7 Conclusion

In this paper, we demonstrate the applicability of FO transition systems and HyperFOLTL formulas to express non-interference policies in distributed systems, using the leader election protocol as an example. We explore a potential approach for formally verifying such properties. Future work will involve formalizing the conditions proposed in section 4.3 as a general deduction rule, implementing the automatic verification of the proposed methodolgy and evaluating its effectivness. We also intend to extend our approach to cover other hyperproperties.

References

1. Barbuti, R., Tesei, L.: A decidable notion of timed non-interference. Fundamenta Informaticae **54**(2-3), 137–150 (2003)
2. Bell, D., Padula, L.: Secure Computer Systems: Mathematical Foundations and Model. Mitre Corporation (1973), https://books.google.fr/books?id=y_SNPAAACAAJ
3. Beutner, R., Finkbeiner, B.: Software verification of hyperproperties beyond k-safety. In: Shoham, S., Vizel, Y. (eds.) Computer Aided Verification. pp. 341–362. Springer International Publishing, Cham (2022)
4. Chang, E., Roberts, R.: An improved algorithm for decentralized extrema-finding in circular configurations of processes. Commun. ACM **22**(5), 281–283 (May 1979). https://doi.org/10.1145/359104.359108, http://doi.acm.org/10.1145/359104.359108
5. Clarkson, M.R., Finkbeiner, B., Koleini, M., Micinski, K.K., Rabe, M.N., Sánchez, C.: Temporal logics for hyperproperties. CoRR **abs/1401.4492** (2014), http://arxiv.org/abs/1401.4492
6. Clarkson, M.R., Schneider, F.B.: Hyperproperties. In: 2008 21st IEEE Computer Security Foundations Symposium. pp. 51–65 (2008). https://doi.org/10.1109/CSF.2008.7
7. Conchon, S., Goel, A., Krstic, S., Mebsout, A., Zaïdi, F.: Cubicle: A parallel smt-based model checker for parameterized systems - tool paper. In: CAV. pp. 718–724 (2012)
8. Dardinier, T., Müller, P.: Hyper hoare logic: (dis-)proving program hyperproperties. Proceedings of the ACM on Programming Languages **8**(PLDI), 1485–1509 (Jun 2024). https://doi.org/10.1145/3656437, http://dx.doi.org/10.1145/3656437
9. Dickerson, R., Ye, Q., Zhang, M.K., Delaware, B.: Rhle: Modular deductive verification of relational ∀∃ properties. In: Programming Languages and Systems: 20th Asian Symposium, APLAS 2022, Auckland, New Zealand, December 5, 2022, Proceedings. p. 67–87. Springer-Verlag, Berlin, Heidelberg (2022). https://doi.org/10.1007/978-3-031-21037-2_4, https://doi.org/10.1007/978-3-031-21037-2_4
10. Finkbeiner, B.: Temporal hyperproperties. Bulletin of the EATCS **123** (2017)
11. Finkbeiner, B., Frenkel, H., Hofmann, J., Lohse, J.: Automata-based software model checking of hyperproperties (2023)
12. Finkbeiner, B., Müller, C., Seidl, H., Zălinescu, E.: Verifying security policies in multi-agent workflows with loops. In: Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security. p. 633–645. CCS '17, Association for Computing Machinery, New York, NY, USA (2017). https://doi.org/10.1145/3133956.3134080, https://doi.org/10.1145/3133956.3134080
13. Finkbeiner, B., Zimmermann, M.: The first-order logic of hyperproperties. CoRR **abs/1610.04388** (2016), http://arxiv.org/abs/1610.04388
14. Focardi, R., Gorrieri, R., Martinelli, F.: Real-time information flow analysis. IEEE Journal on Selected Areas in Communications **21**(1), 20–35 (2003). https://doi.org/10.1109/JSAC.2002.806122
15. Gardey, G., Mullins, J., Roux, O.H.: Non-interference control synthesis for security timed automata. Electronic Notes in Theoretical Computer Science **180**(1), 35–53 (2007). https://doi.org/https://doi.org/10.1016/j.entcs.2005.05.046, https://www.sciencedirect.com/science/article/pii/S1571066107003143,

proceedings of the International Workshop on Security and Concurrency (SecCo 2005)
16. Ghosal, S., Shyamasundar, R.K.: A generalized notion of non-interference for flow security of sequential and concurrent programs. In: 2020 27th Asia-Pacific Software Engineering Conference (APSEC). pp. 51–60 (2020). https://doi.org/10.1109/APSEC51365.2020.00013
17. Goguen, J.A., Meseguer, J.: Security policies and security models. In: 1982 IEEE Symposium on Security and Privacy. pp. 11–20 (1982). https://doi.org/10.1109/SP.1982.10014
18. Lamport, L.: A simple approach to specifying concurrent systems. Commun. ACM **32**(1), 32–45 (jan 1989). https://doi.org/10.1145/63238.63240, https://doi.org/10.1145/63238.63240
19. Lamport, L.: Specifying Systems: The TLA+ Language and Tools for Hardware and Software Engineers. Addison-Wesley Longman Publishing Co., Inc., USA (2002)
20. Macedo, N., Brunel, J., Chemouil, D., Cunha, A., Kuperberg, D.: Lightweight specification and analysis of dynamic systems with rich configurations. In: Proceedings of the 2016 24th ACM SIGSOFT International Symposium on Foundations of Software Engineering. p. 373–383. FSE 2016, Association for Computing Machinery, New York, NY, USA (2016). https://doi.org/10.1145/2950290.2950318, https://doi.org/10.1145/2950290.2950318
21. Müller, C., Seidl, H., Zălinescu, E.: Inductive invariants for noninterference in multi-agent workflows. In: 2018 IEEE 31st Computer Security Foundations Symposium (CSF). pp. 247–261 (2018). https://doi.org/10.1109/CSF.2018.00025
22. Padon, O., Hoenicke, J., Losa, G., Podelski, A., Sagiv, M., Shoham, S.: Reducing liveness to safety in first-order logic. Proc. ACM Program. Lang. **2**(POPL) (dec 2017). https://doi.org/10.1145/3158114, https://doi.org/10.1145/3158114
23. Padon, O., McMillan, K.L., Panda, A., Sagiv, M., Shoham, S.: Ivy: Safety verification by interactive generalization. SIGPLAN Not. **51**(6), 614–630 (jun 2016). https://doi.org/10.1145/2980983.2908118, https://doi.org/10.1145/2980983.2908118
24. Peyras, Q., Bodeveix, J.P., Brunel, J., Chemouil, D.: Sound verification procedures for temporal properties of infinite-state systems. In: Silva, A., Leino, K.R.M. (eds.) Computer Aided Verification. pp. 337–360. Springer International Publishing, Cham (2021)
25. Pnueli, A.: The temporal logic of programs. In: 18th Annual Symposium on Foundations of Computer Science (sfcs 1977). pp. 46–57 (1977). https://doi.org/10.1109/SFCS.1977.32
26. Sabelfeld, A., Myers, A.: Language-based information-flow security. IEEE Journal on Selected Areas in Communications **21**(1), 5–19 (2003). https://doi.org/10.1109/JSAC.2002.806121
27. Zimmermann, J., Mohay, G.: Distributed intrusion detection in clusters based on non-interference. In: Proceedings of the 2006 Australasian Workshops on Grid Computing and E-Research - Volume 54. p. 89–95. ACSW Frontiers '06, Australian Computer Society, Inc., AUS (2006)