Simplifying Safety Proofs with Forward-Backward Reasoning and Prophecy

EDEN FRENKEL, Tel Aviv University, Israel
KENNETH L. MCMILLAN, University of Texas at Austin, USA
ODED PADON, Weizmann Institute of Science, Israel
SHARON SHOHAM, Tel Aviv University, Israel

We propose an incremental approach for safety proofs that decomposes a proof with a complex inductive invariant into a sequence of simpler proof steps. Our proof system combines rules for (i) forward reasoning using inductive invariants, (ii) backward reasoning using inductive invariants of a time-reversed system, and (iii) prophecy steps that add witnesses for existentially quantified properties. We prove each rule sound and give a construction that recovers a single safe inductive invariant from an incremental proof. The construction of the invariant demonstrates the increased complexity of a single inductive invariant compared to the invariant formulas used in an incremental proof, which may have simpler Boolean structures and fewer quantifiers and quantifier alternations. Under natural restrictions on the available invariant formulas, each proof rule strictly increases proof power. That is, each rule allows to prove more safety problems with the same set of formulas. Thus, the incremental approach is able to reduce the search space of invariant formulas needed to prove safety of a given system. A case study on Paxos, several of its variants, and Raft demonstrates that forward-backward steps can remove complex Boolean structure while prophecy eliminates quantifiers and quantifier alternations.

CCS Concepts: • **Theory of computation** → **Logic and verification**; **Proof theory**; **Program verification**; *Automated reasoning*; • **Networks** → *Protocol testing and verification.*

Additional Key Words and Phrases: deductive verification, incremental proofs, forward analysis, backward analysis, prophecy, Paxos, Raft.

ACM Reference Format:
Eden Frenkel, Kenneth L. McMillan, Oded Padon, and Sharon Shoham. 2026. Simplifying Safety Proofs with Forward-Backward Reasoning and Prophecy. *Proc. ACM Program. Lang.* 10, PLDI, Article 263 (June 2026), 25 pages. https://doi.org/10.1145/3808341

1 Introduction

A prominent approach for verifying safety properties of complex systems is based on inductive invariants. Inductive invariants are properties that hold in the initial states and are preserved by every step of the system, and ergo they hold in all reachable states of the system. Therefore, a safety property can be proved by finding an inductive invariant that implies it (a *safe inductive invariant*). As systems and properties become more complex, the inductive invariants required to establish their safety may become extremely complex. In particular, if we use formulas in first-order logic to model systems, their safety properties, and their inductive invariants, the complexity of inductive invariants is exhibited in the Boolean structure of the corresponding formulas, the number of

Authors' Contact Information: Eden Frenkel, Tel Aviv University, Tel Aviv, Israel, edenfrenkel@mail.tau.ac.il; Kenneth L. McMillan, University of Texas at Austin, Austin, USA, kenmcm@cs.utexas.edu; Oded Padon, Weizmann Institute of Science, Rehovot, Israel, oded.padon@weizmann.ac.il; Sharon Shoham, Tel Aviv University, Tel Aviv, Israel, sharon.shoham@gmail.com.

This work is licensed under a Creative Commons Attribution 4.0 International License.
© 2026 Copyright held by the owner/author(s).
ACM 2475-1421/2026/6-ART263
https://doi.org/10.1145/3808341

quantifiers, and the structure of quantifier alternations. The complexity of inductive invariants poses a major challenge for verification. First, complex inductive invariants induce a huge search space, making it difficult to synthesize them automatically. Second, even validating that a formula is indeed an inductive invariant becomes challenging, both because of scale and because of the effect of quantifier structure on the complexity and decidability of validating inductiveness.

One common methodology for simplifying safety proofs relies on simplifying the components being verified. This is the general idea behind modular reasoning, which divides the verification task to sub-tasks that are simpler to reason about such that the safety proof of the overall system follows from the proofs of the sub-tasks. Examples of this methodology include assume-guarantee reasoning [Abadi and Lamport 1993, 1995; Henzinger et al. 1998; Jones 1981, 1983; Pnueli 1985] and Owicki-Gries proofs [Owicki and Gries 1976], where each sub-task verifies a component of the system. However, there are cases where even a single component requires complicated proofs. In these cases, simplifications that are tied to the modular structure of the system do not apply.

In this paper we investigate a complementary approach for simplifying safety proofs which, rather than dividing the system being verified, divides the proof itself into simpler sub-proofs. We do so by introducing *incremental proofs* that combine *forward* and *backward* reasoning and a suitable form of *prophecy* variables.

Our starting point is the well-known observation that safety can be established by *incremental induction*, where in each step an invariant is proved by induction while assuming previously proved invariants. Similarly to safe inductive invariants, all the invariants proven in such an incremental proof hold on all reachable states. Our first observation is that we can allow incremental proofs that combine forward and backward steps, where a backward step uses an invariant that holds in all states that are backward reachable from the bad states. Therefore, the combination of forward and backward steps allows the proof to use predicates that do not hold on either the forward reachable or the backward reachable states, but maintains the fact that the predicates hold on all paths from an initial state to a bad state. We observe that this combination can reduce the complexity (e.g., Boolean structure) of predicates used in proofs when compared to safe inductive invariants.

Next, we observe that we can sometimes simplify the kind of quantification needed in forward or backward invariants by introducing prophecy witnesses for some existentially quantified property, and using the witnesses in subsequent proof steps instead of quantified variables. Our notion of prophecy is similar to prophecy variables used in prior work, e.g. [Lamport and Merz 2022; Mann et al. 2021; Vick and McMillan 2023], but our development includes two key novelties. First, our notion of prophecy admits a sound and complete characterization in terms of an auxiliary safety problem: checking whether a certain prophecy witness can be added amounts to checking the safety of an auxiliary transition system. Second, our notion of prophecy combines particularly well with forward and backward proofs. Prophecy may not only decrease the number of quantifiers needed in proofs, but also the quantification depth and possibly eliminate quantifier alternations.

Based on these observations we define a proof system for incremental safety proofs that allows to combine forward-backward reasoning and prophecy in a synergetic way. This is enabled by our characterization of the soundness of adding prophecy witnesses as a safety problem. For example, it is possible to use forward-backward proof steps for identifying sound prophecy witnesses, and then use these witnesses in subsequent proof steps. This kind of interaction between the proof steps is key to simplifying the formulas used in proofs.

We demonstrate the ability of incremental proofs to simplify safety proofs both theoretically and via a case study. We prove that if we restrict the set of formulas that can be used as inductive invariants in proofs, then incremental proofs have an increased proof power compared to safe inductive invariants. That is, there are cases where no safe inductive invariant exists within the chosen set of formulas but an incremental proof does exist. Moreover, we show that each proof rule

strictly increases the overall proof power of the proof system. Reducing the complexity of formulas used in proofs may be beneficial for proof automation.

As a case study we consider the Paxos protocol for consensus [Lamport 1998, 2001], several of its variants, and Raft [Ongaro and Ousterhout 2014]. Safety verification of these protocols has been extensively investigated in recent years due to their importance in distributed systems and their intricate safety proofs that require a non-clausal Boolean structure and quantifier alternations [Frenkel et al. 2024; Hance et al. 2021; Hawblitzel et al. 2015; Koenig et al. 2022; Padon et al. 2017; Yao et al. 2022; Zhang et al. 2024, 2025a]. To expand our case study, for each variant of Paxos we consider several versions differing in how the protocol and its safety property are modeled. We show that in some cases we can significantly simplify the safety proofs of these protocols using our proof system, for example reducing a proof that required quantifier alternations and non-clausal predicates to clauses that are purely universally quantified. The case study also demonstrates that forward-backward reasoning can aid in finding useful prophecy witnesses that might be difficult to find using forward reasoning only. It therefore illustrates the significance of the synergetic combination of forward-backward and prophecy provided by our proof system. To the best of our knowledge, this kind of interaction between forward-backward reasoning and prophecy is novel.

In summary, this paper makes the following contributions:

- We introduce a proof system for incremental safety proofs that combines forward reasoning, backward reasoning and introduction of prophecy in a synergetic way.
- We provide a characterization of the soundness of our notion of prophecy witnesses using an auxiliary safety problem.
- We relate incremental safety proofs to the standard notion of safe inductive invariants via a translation of incremental proofs to inductive invariants.
- We investigate the proof power of incremental proofs and show that they may sometimes eliminate Boolean connectives as well as quantifiers, including nested and alternating quantifiers, that are required in safe inductive invariants. The interaction of forward-backward reasoning and prophecy is key to achieving such simplifications.
- We present a case study where we examine the Paxos consensus protocol, several of its variants, and Raft, and demonstrate how incremental proofs simplify their safety proofs.

The rest of the paper is organized as follows: Section 2 presents an overview of our approach for simplifying proofs, Section 3 includes preliminary definitions and notation, Section 4 describes our proof system for incremental forward-backward safety proofs, Section 5 extends the proof system with a proof rule for introducing prophecy witnesses, and Section 6 details the case study demonstrating proof simplification for Paxos variants and Raft. Finally, Section 7 discusses related work and concludes the paper.

2 Overview

In this section we illustrate and motivate the different proof rules comprising our proof system for incremental safety proofs. We do so by presenting several examples, where incremental proofs allow us to use invariants that are simpler than the safe inductive invariants required for establishing safety of the original problem. We demonstrate simplifications both in the Boolean structure of invariants and in their quantification.

2.1 Simplifying the Boolean Structure of Invariants

Our first motivating example considers a simple token-passing protocol between a *dealer* and two *players*, modeled with propositional variables: a, d, p_1, p_2. When a is true we say that the protocol is

active, and this property of the protocol never changes; d, p_1, and p_2 represent whether the dealer and the players each hold a token, respectively.

The protocol starts with the second player not holding a token, thus the initial condition is:

$$\iota = \neg p_2.$$

Then, in each step, one of three transitions can take place, each of which is captured by a formula that relates the values of the variables a, d, p_1, p_2 before the transition to their values after the transition, encoded by a', d', p_1', p_2'. If the protocol is active, a token can be passed from the dealer to the first player:

$$\tau_1 = a \wedge a' \wedge d \wedge \neg d' \wedge p_1' \wedge (p_2' \leftrightarrow p_2)$$

Moreover, the first player can always pass a token to the second:

$$\tau_2 = (a' \leftrightarrow a) \wedge (d' \leftrightarrow d) \wedge p_1 \wedge \neg p_1' \wedge p_2'$$

Lastly, if the protocol is inactive, the dealer can gain tokens arbitrarily:

$$\tau_3 = \neg a \wedge \neg a' \wedge d' \wedge (p_1' \leftrightarrow p_1) \wedge (p_2' \leftrightarrow p_2).$$

All in all, the transition formula for this protocol is given by $\tau = \tau_1 \vee \tau_2 \vee \tau_3$.

Say we would like to verify that it is impossible for the dealer and both players to all hold tokens simultaneously. That is, we would like to prove that the following bad states are unreachable:

$$\beta = d \wedge p_1 \wedge p_2.$$

Unfortunately, the property $\neg\beta \equiv \neg d \vee \neg p_1 \vee \neg p_2$ is not preserved by τ_3, and is therefore not inductive. In fact, there is no clause that can serve as a safe inductive invariant, since any clause that implies $\neg\beta$ must be a subset of the literals $\{\neg d, \neg p_1, \neg p_2\}$, and it is easy to check that none produce an inductive invariant. For a similar reason, there is no negation of a clause, or conjunction of literals, that is a safe inductive invariant for this safety problem (such a conjunction would have to be exactly $\neg p_2$ to be implied by ι, which is not inductive).

Thus, when looking for a safe inductive invariant for this simple protocol, either manually or automatically, we need to consider candidates containing both disjunctions and conjunctions. One such safe inductive invariant for this problem is as follows:

$$\varphi = (a \wedge \neg d) \vee \neg p_1 \vee \neg p_2,$$

which can be read as "either the protocol is active and the dealer does not hold a token, or one of the two players does not hold a token". This property holds on all initial states and is preserved by the transitions: τ_1 precisely results in a state where the protocol is active and the dealer has no token, τ_2 results in a state where the first player does not hold a token, and τ_3 considers an inactive protocol and does not change p_1 and p_2, which means that $\neg p_1 \vee \neg p_2$ must hold both before and after the transition. It also holds that $\varphi \Rightarrow \neg\beta$, which makes the inductive invariant safe.

When rewritten in conjunctive normal form, φ is equivalent to the conjunction of two clauses: $\varphi_1 = a \vee \neg p_1 \vee \neg p_2$ and $\varphi_2 = \neg d \vee \neg p_1 \vee \neg p_2$. We observe that it is sometimes possible to prove conjunctive invariants such as φ *incrementally*, by proving simpler auxiliary inductive invariants until those, when taken together, manage to prove safety. This can be done by iteratively verifying a sequence of inductive invariants $\varphi_1, \ldots, \varphi_n$, where for every $i = 1, ..., n$ we add $\bigwedge_{j<i} \varphi_j$ as additional axioms of the safety problem when verifying φ_i. Namely, we strengthen the initial states to $\iota \wedge \bigwedge_{j<i} \varphi_j$ and the transitions to $\bigwedge_{j<i} \varphi_j \wedge \tau \wedge \bigwedge_{j<i} \varphi_j'$. If, in addition, $\bigwedge_{i\le n} \varphi_i$ implies safety, then $\bigwedge_{i\le n} \varphi_i$ is a safe inductive invariant of the original safety problem.

For our simple example such a proof can be constructed from φ_1 and φ_2, since φ_1 is inductive by itself, and φ_2 is inductive under the assumption that φ_1 holds on every pre-state and post-state of a transitions, i.e., $\varphi_1 \wedge \varphi_2 \wedge \tau \wedge \varphi_1' \Rightarrow \varphi_2'$. Since $\varphi_1 \wedge \varphi_2$ implies $\neg\beta$ (and is in fact equivalent to φ),

this forms an incremental forward proof of safety, where in each step we added a clause of size 3, whereas no safe inductive invariant of the problem can be clausal in form.

However, we can construct a different kind of incremental proof that uses even simpler auxiliary invariants. This is done by observing that instead of a proof method that accumulates inductive invariants, which are properties of reachable states, we can accumulate properties of states that appear on error traces, i.e., on traces from an initial state to a bad state (even when such traces are hypothetical and do not exist). This allows us to prove safety "by contradiction": if the accumulated properties imply $\neg\beta$ or $\neg\iota$, we deduce that there are no error traces and the safety problem is safe.

To that end, in each step of the incremental proof we allow introducing either an inductive invariant, or a *backward-inductive invariant*, i.e., a formula φ such that $\beta \Rightarrow \varphi$ and $\varphi' \wedge \tau \Rightarrow \varphi$. Such a formula holds on all backward-reachable states from β, and is therefore an invariant of all error traces of the safety problem. Intuitively, this allows us to use simpler inductive invariants in incremental proofs, because in subsequent proof steps we can rely on auxiliary invariants that might not be true of reachable states, whereas in incremental forward proofs all auxiliary invariants were actual (forward) invariants of the problem.

For the simple dealer-player protocol we consider, we can use a backward-inductive invariant $\varphi_3 = \neg a \vee d$, since it holds on all bad states and is preserved by backward transitions (in bad states the dealer holds a token, and can lose a token in a backward transition only if the protocol is inactive). Then, using this assumption as an axiom for a forward proof step, the transition τ_1 is disabled (all its post-states violate φ_3), and therefore $\varphi_4 = \neg p_1 \vee \neg p_2$ is a (forward) inductive invariant. Thus, in this incremental forward-backward proof we can conclude that all states on traces from initial to bad states satisfy $\varphi_3 \wedge \varphi_4 = (\neg a \vee d) \wedge (\neg p_1 \vee \neg p_2)$, which implies $\neg\beta = \neg(d \wedge p_1 \wedge p_2)$. In this way we have shown the protocol to be safe using the auxiliary invariants $\neg a \vee d$ and $\neg p_1 \vee \neg p_2$, compared to $a \vee \neg p_1 \vee \neg p_2$ and $\neg d \vee \neg p_1 \vee \neg p_2$ in the incremental forward proof. This may seem like a modest simplification in the auxiliary invariants one needs to consider, but this stems from the very simple nature of the protocol. As we shall see in the sequel, the presence of quantification can significantly transform the power of forward-backward proofs compared to forward ones.

Note that $\varphi_3 \wedge \varphi_4$ is not an inductive invariant of the original protocol: it does not hold in the initial states and it is not preserved by τ_1. It is only guaranteed to hold in all the states along traces from an initial state to a bad state. However, the incremental proof does induce (another) safe inductive invariant for the protocol. In Section 4.2 we show that any incremental forward-backward proof can be converted into a safe inductive invariant of the problem which consists of a Boolean combination of the predicates used in the proof (Theorem 4.13). For the proof using the backward invariant $\varphi_3 = \neg a \vee d$ and the forward invariant $\varphi_4 = \neg p_1 \vee \neg p_2$, this results in $\neg\varphi_3 \vee \varphi_4 = \neg(\neg a \vee d) \vee (\neg p_1 \vee \neg p_2)$, which is equivalent to the safe inductive invariant $\varphi = (a \wedge \neg d) \vee (\neg p_1 \vee \neg p_2)$ presented before.

2.2 Forward-Backward Proofs and Quantification

We introduce quantification and first-order semantics into our motivating protocol by replacing the propositional variables with unary relations $a(\cdot), d(\cdot), p_1(\cdot), p_2(\cdot)$. Instead of a single dealer and two players, the protocol works with *teams*, each of which is composed of a dealer and two players, where each of $d(t)$, $p_1(t)$, and $p_2(t)$ is true if the dealer, first player, or second player of team t holds a token, respectively. Similarly, a team t is active if $a(t)$ holds. Initially, all second players hold no tokens:

$$\iota = \forall x. \neg p_2(x).$$

Similarly to before, the first transition allows a dealer from an active team to pass a token to a first player, but not necessarily from his own team. For readability, we use TLA+ style notation where

we specify transitions as formulas $[\varphi]_S$ where S denotes the modified set of values (i.e., anything outside S is not modified by the transition):

$$\tau_1 = \exists x, y.[a(x) \wedge d(x) \wedge \neg d'(x) \wedge p_1'(y)]_{\{d(x), p_1(y)\}}.$$

The second transition lets a first player of a team pass a token to the second player of the team:

$$\tau_2 = \exists x.[p_1(x) \wedge \neg p_1'(x) \wedge p_2'(x)]_{\{p_1(x), p_2(x)\}}.$$

And the last transition allows a dealer of an inactive team to gain a token:

$$\tau_3 = \exists x.[\neg a(x) \wedge d'(x)]_{\{d(x)\}}$$

The bad states of this updated protocol are the states where all dealers hold a token, and there is a team where both players hold a token as well:

$$\beta = \forall x.d(x) \wedge \exists y.p_1(y) \wedge p_2(y).$$

As in Section 2.1, $\neg\beta$ is not preserved by τ_3 and is therefore not inductive. One safe inductive invariant for the problem is

$$\varphi = (\exists x.a(x) \wedge \neg d(x)) \vee (\forall x.\neg p_1(x) \vee \neg p_2(x)).$$

Intuitively, this inductive invariant says that "Either there is an active team where the dealer has no token, or in every team there is a player that does not hold a token". Recall that in Section 2.1, we had no clausal safe inductive invariant, but could find an incremental forward proof that used only clausal auxiliary invariants, namely $a \vee \neg p_1 \vee \neg p_2$ and $\neg d \vee \neg p_1 \vee \neg p_2$. An analogous proof, however, seems impossible here, since the invariant cannot even be decomposed into conjuncts in a similar way. A similar forward-backward proof, however, is possible: we can first take $\varphi_1 = \forall x.\neg a(x) \vee d(x)$, which is backward-inductive. Using it as an axiom once again disables τ_1, which makes $\varphi_2 = \forall x.\neg p_1(x) \vee \neg p_2(x)$ inductive in the forward direction. Taken together, $\varphi_1 \wedge \varphi_2 \Rightarrow \neg\beta$, which concludes the incremental forward-backward safety proof.

Converting this forward-backward proof into a safe inductive invariant in the way described in Section 4.2 results in the invariant $\neg\varphi_1 \vee \varphi_2 = \neg(\forall x.\neg a(x) \vee d(x)) \vee (\forall x.\neg p_1(x) \vee \neg p_2(x))$, which is once again equivalent to φ. The incremental proof itself only needed to consider, and to perform inductiveness checks, over universally quantified clauses. On the other hand, this problem does not have any safe inductive invariant, or an incremental proof that is purely forward or purely backward, that consists of universally quantified clauses. This claim can be proved by the PDR$^\forall$ algorithm [Karbyshev et al. 2017] which is able to provide a proof that no universally quantified safe inductive invariant exists for this problem (and its time-reversal), let alone a clausal one. Indeed, we use the implementation of PDR$^\forall$ in `mypyvy` [Wilcox et al. 2024] to obtain such a proof.

2.3 Using Prophecy to Reduce Quantifier Nesting and Alternations

In the example of Section 2.2, we showed that an incremental forward-backward proof reduces the number of quantifiers needed in auxiliary invariants compared to those in a safe inductive invariant or a forward incremental proof. However, because a forward-backward proof can always be converted into a safe inductive invariant which is a Boolean combination of its auxiliary invariants (Theorem 4.13), forward-backward proofs on their own cannot reduce quantifier nesting or alternations. To that end, we propose using a notion of prophecy defined in Section 5 which replaces quantified variables with explicit witnesses.

For an example where quantifier alternations are needed in the inductive invariant, and are eliminated by the use of prophecy, we use a slightly simplified protocol compared to Sections 2.1 and 2.2, which nonetheless demonstrates how we gain quantifier nesting and alternations from using prophecy in incremental forward-backward proofs. This protocol still consists of an *active*

relation $a(\cdot)$, as well as a dealer relation, which is now binary, $d(\cdot,\cdot)$. As before, the *active* relation is unchanging and denotes whether a dealer of a certain team can pass tokens. Now, however, the dealer of each team can hold a token for every team, i.e., $d(t_1, t_2)$ indicates that the dealer of t_1 still holds the token associated with t_2. The players are no longer part of the modeled protocol.

Initially there is at least one inactive team, and dealers hold all tokens:

$$\iota = \exists x.\neg a(x) \wedge \forall x, y.d(x, y).$$

There is only one transition, where a dealer of an active team can lose any of its tokens (by passing it to the players of the appropriate team, which are not modeled):

$$\tau = \exists x, y.[a(x) \wedge \neg d'(x, y)]_{\{d(x,y)\}}.$$

Lastly, the bad states are those where all dealers have lost at least one token:

$$\beta = \forall x.\exists y.\neg d(x, y).$$

Intuitively, this state cannot be reached because all reachable states contain an inactive team whose dealer always holds all of their tokens. However, like before, $\neg\beta$ is not preserved by τ and is therefore not inductive, but we can write a safe inductive invariant for this problem that precisely captures the inactive team:

$$\varphi = \exists x.\neg a(x) \wedge \forall y.d(x, y).$$

This invariant contains quantifier nesting and alternations. Since a safe inductive invariant for this problem would sensibly require quantifier nesting, we cannot have a forward-backward proof using auxiliary invariants with a single quantifier as before, since such a proof can be converted into nesting-free safe inductive invariant (Theorem 4.13). Additionally, once again we obtain a proof that this problem has no universal forward proof or a universal backward proof via the PDR$^\forall$ algorithm [Karbyshev et al. 2017] implemented in `mypyvy` [Wilcox et al. 2024], clausal or otherwise.

To allow incremental proofs that avoid quantifier nesting and alternations, as well as simplify the Boolean structure of the required predicates, we propose using *prophecy*. The idea is to allow a proof step that adds a fresh symbol w to the safety problem, along with an axiom φ that prophesies some property of w along every trace. This is a generalization of the previous proof steps, which added axioms that did not mention any new symbols. Moreover, similarly to how in a forward-backward proof step we required φ to be an invariant of error traces (to ensure soundness), with the prophecy proof rule we require that adding φ as an axiom is guaranteed to preserve all error traces of the problem. That is, a prophecy step φ is *sound* for a safety problem $\Pi = (\iota, \tau, \beta)$ if every error trace of Π can be transformed (by assigning a certain value to w) into an error trace of the problem with the additional axiom, i.e., the safety problem with initial states $\iota \wedge \varphi(w)$, transitions $\varphi(w) \wedge \tau \wedge (w' = w) \wedge (\varphi(w))'$, and bad states $\beta \wedge \varphi(w)$.

In the above example, any forward trace starting from ι must have some team x such that $\neg a(x)$, since this holds in all initial states and $a(\cdot)$ remains unchanged. Formally, this can be verified by checking that $\iota \Rightarrow \exists x.\neg a(x)$, and that $\neg a(x)$, which is a formula with a free variable x, is forward-preserved, i.e., $\neg a(x) \wedge \tau \Rightarrow \neg a'(x)$. Since these verification conditions are valid, on forward traces from ι it is sound to assume that there exists some witness w which satisfies $\neg a(x)$. Next, we can add w as an unchanging witness to the safety problem along with the axiom $\varphi_1 = \neg a(w)$, resulting in an equivalently safe safety problem.

Under this prophecy assumption, the transition τ becomes disabled for $x = w$, and so the formula $\varphi_2 = \exists y.\neg d(w, y)$ is backward inductive. Taken together $\varphi_1 \wedge \varphi_2 = \neg a(w) \wedge \exists y.\neg d(w, y)$ implies $\neg\iota$, which means that the problem is safe. Note that this proof used only existentially quantified auxiliary invariants, with a single quantified variable and no Boolean connectives. However, we proved a property whose safe inductive invariant contained quantifier nesting and alternations.

In Section 5 we generalize this proof method by precisely defining the notion of a *sound prophecy* used in this paper and showing that the soundness of a given prophecy can itself be stated as a safety problem to be proven, which fits neatly into our approach of incrementally proving various safety problems until we are able to prove the original one in question. Moreover, simiarly to before, we describe how a forward-backward incremental proof with prophecy can be converted into a safe inductive invariant for the problem (Theorem 5.6). Like with forward-backward proofs, this conversion can create complex Boolean combinations of the auxiliary invariants used in the proof, but may also introduce quantifiers to eliminate the use of the witness symbols used in prophecy. For example, converting the above proof results in the safe inductive invariant $\varphi = \exists x.\neg a(x) \wedge \forall y.d(x,y)$, where in this simple example, the witness has been effectively existentially quantified.

3 Preliminaries

In this section we provide the background necessary for our developments. We use first-order logic to model systems and their properties. System states are encoded using a first-order vocabulary Σ which consists of constant symbols, function symbols and relation symbols. Given a first-order vocabulary Σ, a *transition system* is a pair (ι, τ), where ι is a closed formula over Σ representing initial states of the system and τ is a closed formula over $\Sigma \uplus \Sigma'$ that represents the transitions of the system. $\Sigma' = \{a' \mid a \in \Sigma\}$ is a copy of the vocabulary used to represent the post-state of a transition. For a formula φ over Σ we denote by φ' the formula obtained by substituting each symbol in Σ with its counterpart in Σ'. We use τ^{-1} to denote the inverse of τ obtained by replacing each symbol from Σ by its counterpart in Σ' and vice versa.

A *safety problem* is a triple (ι, τ, β), where (ι, τ) is a transition system and β is a closed formula over the same vocabulary Σ representing bad states. The safety problem (ι, τ, β) is safe if there is no sequence of τ-transitions (a trace of (ι, τ)) from an initial state to a bad state.

A closed formula φ over Σ is an *inductive invariant* for a transition system (ι, τ) if $\iota \Rightarrow \varphi$ (initiation) and $\varphi \wedge \tau \Rightarrow \varphi'$ (consecution), which means that φ holds in all the initial states and it is preserved by the transitions of the system. The formula φ is a *safe inductive invariant* for the safety problem (ι, τ, β) if, in addition, $\varphi \Rightarrow \neg\beta$ (safety). An inductive invariant over-approximates the set of states that can be reached from the initial states of the system. Therefore, if it is safe, it implies that the reachable states are disjoint from the bad states, thus the safety problem is safe.

4 Incremental Proofs

In this paper we present a family of proof systems for constructing incremental safety proofs. The soundness of each proof system follows from the soundness of its individual inference rules (see explicit proofs in the extended version of the paper [Frenkel et al. 2026b]). In this section we start with incremental proofs that combine forward and backward reasoning.

4.1 Incremental Forward Proofs

The statements of the proof systems we present are safety problems. Each proof system is defined by a set of inference rules whose premises and conclusions are safety problems, possibly with additional side conditions. Provable statements correspond to safety problems that are provably safe using the inference rules of the system. A proof of a safety problem Π is then a tree where each node is a safety problem, the root is Π and each node is obtained by applying an inference rule.

Since we are interested in analyzing the proof power of each system, we sometimes parameterize a proof system by the class of predicates $\mathbb{P}$ used as inductive invariants in proof rules. We compare how expressive that class has to be to prove the same safety problems in different proof systems.

The most basic proof component of the incremental proofs we consider is proving an invariant by induction, i.e., by showing that all initial states satisfy it, and that all transitions preserve it.

Since not all invariants of a transition system are inductive, to make such a system (relatively) complete, we also include an inference rule that allows strengthening the bad states (i.e., weakening the proven invariant), which preserves safety.

Definition 4.1. The *forward proof system* **F** consists of the following proof rules:

$$\text{(Ind)}\ \frac{}{(\iota, \tau, \neg\varphi)}\ \iota \Rightarrow \varphi,\ \varphi \wedge \tau \Rightarrow \varphi' \qquad \text{(Cons)}\ \frac{(\iota, \tau, \neg\varphi)}{(\iota, \tau, \beta)}\ \varphi \Rightarrow \neg\beta$$

For a class of predicates $\mathbb{P}$, we denote by $\mathbf{F}^{\mathbb{P}}$ the system **F** where applications of (Ind) are restricted to $\varphi \in \mathbb{P}$.

Theorem 4.2 (Soundness). *Every safety problem that has an* **F***-proof is safe.*

Note that all **F**-proofs can be collapsed into the following form, by collapsing consecutive consequence rules:

$$\begin{array}{r} \text{(Ind)}\ \dfrac{}{(\iota, \tau, \neg\varphi)}\ \iota \Rightarrow \varphi,\ \varphi \wedge \tau \Rightarrow \varphi' \\ \text{(Cons)}\ \dfrac{(\iota, \tau, \neg\varphi)}{(\iota, \tau, \beta)}\ \varphi \Rightarrow \neg\beta \end{array}$$

Thus the **F** proof system has no meaningful way to simplify proofs by dividing them into multiple applications of the induction rule. To allow such simplifications we introduce incremental proofs. Incremental proofs may use an additional proof rule that allows adding previously proved invariants as axioms of the safety problem. That is, in order to prove a safety problem $\Pi = (\iota, \tau, \beta)$ safe, the incremental proof rule requires proving that (1) a predicate φ is an invariant with respect to ι and τ, and (2) the problem Π, when restricted to states that satisfy φ, is safe.

Definition 4.3. The *incremental forward proof system* **FI** consists of the proof rules of **F** (Definition 4.1), and the additional incremental proof rule:

$$\text{(Inc)}\ \frac{(\iota, \tau, \neg\varphi) \quad (\iota \wedge \varphi, \tau \wedge \varphi \wedge \varphi', \beta \wedge \varphi)}{(\iota, \tau, \beta)}$$

For a class of predicates $\mathbb{P}$, we denote by $\mathbf{FI}^{\mathbb{P}}$ the system **FI** where applications of (Ind) are restricted to $\varphi \in \mathbb{P}$, and by $\mathbf{FI}^{\mathbb{P}}_n$ the system where proofs use at most n applications of (Ind).

Theorem 4.4 (Soundness). *Every safety problem that has an* **FI***-proof is safe.*

Like the rest of the proof systems presented in this paper, an incremental **FI**-proof of a safety problem can always be converted into a safe inductive invariant for the same problem. This is another way to prove soundness theorems like Theorems 4.2 and 4.4, but can also be used to reason about how expressive inductive invariants need to be to match the proof power of a given proof system. In the case of incremental forward proofs $\mathbf{FI}^{\mathbb{P}}$, each proof can be converted into a safe inductive invariant in the form of a conjunction over $\mathbb{P}$, where the number of conjuncts precisely corresponds to the number of (Ind) rules used in the proof. This conversion is given below.

Definition 4.5. For every proof $P \in \mathbf{FI}$ define $\mathrm{Inv}^{\rightarrow}(P)$ by induction on proof structure:

- if the root of P is a conclusion of (Ind), denoted $(\iota, \tau, \neg\varphi)$, then $\mathrm{Inv}^{\rightarrow}(P) = \varphi$;
- if the root of P is a conclusion of (Cons) with a premise whose proof is $\tilde{P}$, then
$$\mathrm{Inv}^{\rightarrow}(P) = \mathrm{Inv}^{\rightarrow}(\tilde{P});$$
- if the root of P is a conclusion of (Inc) with premises whose proofs are P_1 and P_2, then
$$\mathrm{Inv}^{\rightarrow}(P) = \mathrm{Inv}^{\rightarrow}(P_1) \wedge \mathrm{Inv}^{\rightarrow}(P_2).$$

Theorem 4.6. *Let $P \in \mathbf{FI}^{\mathbb{P}}$ be an incremental forward proof of a safety problem Π. Then $\mathrm{Inv}^{\rightarrow}(P)$ is a safe inductive invariant of Π, and if P contains $n \in \mathbb{N}$ instances of the (Ind) rule, then $\mathrm{Inv}^{\rightarrow}(P)$ is a conjunction of n predicates from $\mathbb{P}$.*

Theorem 4.6, whose proof is provided in the extended version of the paper [Frenkel et al. 2026b], ensures that any proof in $\mathbf{FI}^{\mathbb{P}}_n$ can be converted into a safe inductive invariant in $\wedge_n\mathbb{P}$ for the same safety problem, where $\wedge_n\mathbb{P}$ denotes the class of predicates that includes conjunctions of at most n predicates from $\mathbb{P}$. This hints that incremental proofs may avoid conjunctions that are needed in a safe inductive invariant. In fact, there exist safety problems whose safe inductive invariant requires n conjuncts that can be proven incrementally. These problems are provable in $\mathbf{FI}^{\mathbb{P}}_n$ but not in $\mathbf{FI}^{\mathbb{P}}_{n-1}$ (which subsumes $\mathbf{F}^{\mathbb{P}}$). These results about the proof power of **FI** are summarized in Theorem 4.7.

Theorem 4.7. *For every $\mathbb{P}$, all safety problems that are provable in $\mathbf{FI}^{\mathbb{P}}_n$ are also provable in $\mathbf{F}^{\wedge_n\mathbb{P}}$. However, for every $n > 0$ there exist a set of predicates $\mathbb{P}$ and a safety problem Π such that Π is provable in $\mathbf{FI}^{\mathbb{P}}_n$ but not provable in $\mathbf{FI}^{\mathbb{P}}_{n-1}$.*

Corollary 4.8. *If $\mathbb{P}$ is closed under conjunction, then any $\mathbf{FI}^{\mathbb{P}}$-provable problem is $\mathbf{F}^{\mathbb{P}}$-provable.*

4.2 Incremental Forward-Backward Proofs

The next development in our proof system comes from the observation that a safety problem is safe if and only if its time-reversed problem is safe. That is, to prove the safety of $\Pi = (\iota, \tau, \beta)$, we can equivalently prove the safety of $\Pi^{-1} = (\beta, \tau^{-1}, \iota)$. Intuitively, time-reversal allows us to prove safety problems whose safe inductive invariants require negations, since φ is a safe inductive invariant of Π^{-1} if and only if $\neg\varphi$ is a safe inductive invariant of Π. Although this observation seems simple, the key insight is that when combined with incrementality, a time-reversing inference step, which allows to switch back and forth between forward reasoning and backward reasoning, results in a significant increase of proof power. Namely, since incremental proofs allow us to mimic conjunctive invariants, the addition of negation through time-reversal induces a rich Boolean structure for inductive invariants that contain both negations and conjunctions (which form a functionally complete set of connectives). This means that by using incremental proofs that allow time-reversal steps we may sometimes simplify the Boolean structure of auxiliary inductive invariants used in the proof in cases where a safe inductive invariant of the safety problem requires a complex Boolean structure (beyond conjunctions).

Definition 4.9. The *incremental forward-backward proof system* **FBI** consists of the proof rules of **FI** (Definition 4.1), and the additional time-reversal proof rule:

$$\text{(Rev)}\ \frac{(\beta, \tau^{-1}, \iota)}{(\iota, \tau, \beta)}$$

For a class of predicates $\mathbb{P}$, we denote by $\mathbf{FBI}^{\mathbb{P}}$ the system **FBI** where applications of (Ind) are restricted to $\varphi \in \mathbb{P}$, and by $\mathbf{FBI}^{\mathbb{P}}_n$ the system where proofs use at most n applications of (Ind).

Theorem 4.10 (Soundness). *Every safety problem that has an* **FBI***-proof is safe.*

Although (Rev) is the only necessary addition to the proof system for harnessing backward induction, it is often more concise to write proofs using derived backward variants of (Ind) and (Inc), which directly allow backward induction and incrementality.

Theorem 4.11. *The following are derived rules of* **FBI***:*

$$\text{(B-Ind)}\ \frac{}{(\neg\varphi, \tau, \beta)}\ \beta \Rightarrow \varphi,\ \varphi' \wedge \tau \Rightarrow \varphi \qquad \text{(B-Cons)}\ \frac{(\neg\varphi, \tau, \beta)}{(\iota, \tau, \beta)}\ \varphi \Rightarrow \neg\iota$$

$$\text{(B-Inc)}\ \frac{(\neg\varphi, \tau, \beta) \quad (\iota \wedge \varphi, \tau \wedge \varphi \wedge \varphi', \beta \wedge \varphi)}{(\iota, \tau, \beta)}$$

Proof. Each of the rules can be derived by applying (Rev) to each of the premises, applying the respective **FI** rule, and applying (Rev) to the resulting conclusion. □

Similarly to Definition 4.5 for **FI**, every **FBI**-proof can be converted into a safe inductive invariant for the safety problem in its root. As mentioned before, the only change is the handling of (Rev), which adds a negation to the computed invariant due to the time-reversal. When combined with the conjunctions introduced by the (Inc) rule in Definition 4.5, the computed invariant can now be equivalent to any arbitrary Boolean combination predicates from the considered class.

Definition 4.12. For every proof $P \in \mathbf{FBI}$ define $\mathrm{Inv}^{\rightleftharpoons}(P)$ by induction on proof structure:

- if the root of P is not a conclusion of (Rev), define $\mathrm{Inv}^{\rightleftharpoons}(P)$ like $\mathrm{Inv}^{\rightarrow}(P)$ in Definition 4.5;
- if the root of P is a conclusion of (Rev) with a premise whose proof is $\tilde{P}$, then

$$\mathrm{Inv}^{\rightleftharpoons}(P) = \neg\, \mathrm{Inv}^{\rightleftharpoons}(\tilde{P}).$$

Theorem 4.13. *Let $P \in \mathbf{FBI}^{\mathbb{P}}$ be an incremental forward-backward proof of a safety problem Π. Then $\mathrm{Inv}^{\rightleftharpoons}(P)$ is a safe inductive invariant of Π, and if P contains $n \in \mathbb{N}$ instances of the (Ind) rule, then $\mathrm{Inv}^{\rightleftharpoons}(P)$ is a Boolean combination of n predicates from $\mathbb{P}$.*

The proof of the theorem is deferred to the extended version of the paper [Frenkel et al. 2026b]. The overall results about the proof power of **FBI** are summarized as follows.

Theorem 4.14. *For every $\mathbb{P}$, all safety problems that are provable in $\mathbf{FBI}^{\mathbb{P}}_n$ are also provable in $\mathbf{F}^{\mathbb{P}_n}$, where $\mathbb{P}_n$ consists of all Boolean combinations of at most n predicates from $\mathbb{P}$. However, for every $n > 0$ there exist a set of predicates $\mathbb{P}$ and a safety problem Π such that Π is provable in $\mathbf{FBI}^{\mathbb{P}}_n$ but not provable in either $\mathbf{FBI}^{\mathbb{P}}_{n-1}$, $\mathbf{FI}^{\mathbb{P}}$, or $\mathbf{F}^{\mathbb{P}}$.*

Proof. The first part of the theorem is a direct corollary of Theorem 4.13. For the second part, define the vocabulary with n nullary relations $\Sigma = \{p_1, \ldots, p_n\}$ and the set of predicates $\mathbb{P} = \{p_1, \ldots, p_n\}$. Denote $\varphi = p_1 \wedge (\neg p_2 \vee (p_3 \wedge (\neg p_4 \vee \cdots)))$ and consider the safety problem $\Pi = (\varphi, \bot, \neg\varphi)$. This safety problem is provable in $\mathbf{FBI}^{\mathbb{P}}_n$, but not provable in either $\mathbf{FBI}^{\mathbb{P}}_{n-1}$, $\mathbf{FI}^{\mathbb{P}}$, or $\mathbf{F}^{\mathbb{P}}$, since any safe inductive invariant of Π must be equivalent to φ, and there exists no such formula consisting of fewer than n predicates, or purely of conjunctions over $\mathbb{P}$. □

5 Incremental Forward-Backward Proofs with Prophecy

The proof systems presented in Section 4 let us prove safety properties using inductive invariants of a simpler Boolean structure than that of any safe inductive invariant of the original safety problem. However, the simplification afforded by **FI** and **FBI** *only* allows simplifying the Boolean structure of inductive invariants; that is, if a safe inductive invariant cannot be expressed as a Boolean combination of several simpler predicates, e.g., due to quantification, then there is no hope for these proof systems to simplify the kind of predicates involved in a safety proof.

Our goal in this section is to augment the forward-backward proof system **FBI** with an additional rule that is able to eliminate quantifier nesting and alternation, and thus simplify proofs of problems that require quantified invariants that cannot otherwise be broken down into a Boolean combination of simpler predicates. We first introduce a general prophecy proof rule and then discuss specific heuristics for adding prophecy.

5.1 Prophecy in Forward-Backward Proofs

We introduce a *prophecy proof rule* that prophesies that there exists an element x that always satisfies some formula $\varphi(x)$ and uses it to simplify the proof by extending the vocabulary with a fresh constant symbol, w, that serves as a *witness* for the existential quantifier. The rule then adds $\varphi(w)$ as an axiom. To ensure soundness, it is required that the added axiom does not affect the error traces of the system. Then, by reasoning about the safety problem with the prophecy transformation, we are able to prove properties of the witness which translate to quantified properties of the original

system. Since this proof system is an extension of **FBI**, it is able to eliminate both quantification as well as logical connectives such as conjunctions and disjunctions.

We begin by describing the safety problem resulting from adding the prophecy witness constant. For the sake of simplicity, we consider adding a single witness which remains fixed along any given trace. The approach naturally generalizes to multiple prophecy symbols, and to a prophecy that is updated in each transition according to a given update rule.

Definition 5.1. Let $\Pi = (\iota, \tau, \beta)$ be a safety problem over vocabulary Σ, let w be a fresh constant symbol, and let $\varphi(x)$ be a formula over Σ. Define the safety problem Π^w_φ over $\Sigma \cup \{w\}$ as

$$\Pi^w_\varphi = (\iota \wedge \varphi(w), \varphi(w) \wedge \tau \wedge w' = w \wedge (\varphi(w))', \beta \wedge \varphi(w)).$$

Then $\varphi(w)$ is a *sound prophecy* for Π if for every trace of Π from ι to β, there exists an interpretation of w that results in a trace of Π^w_φ from $\iota \wedge \varphi(w)$ to $\beta \wedge \varphi(w)$.

The soundness condition states that there is a suitable witness for every trace from an initial state to a bad state, where the witness satisfies φ all along the trace. This means that, when a bad state is reached, there exists an x (the witness) such that $\varphi(x)$ has been always true in the past. Therefore, we can formulate the soundness condition in a complete way as a temporal safety problem whose bad states are bad states of the original system where the aforementioned condition does not hold. Namely, for every x, either $\varphi(x)$ does not hold at the moment or it has not been always true in the past. To track whether $\varphi(x)$ has always been true in the past, we follow a standard tableau construction for the "always in the past" temporal operator, and augment the state of the system with a fresh predicate $m(\cdot)$ that remembers for each x whether $\varphi(x)$ has been always true in the past. Initially, every x that satisfies $\varphi(x)$ is added to m, and in every transition, if x continues to satisfy $\varphi(x)$, it remains in m. This ensures that if an element is not in m then it has not always been true in the past. That is, we can use $\neg m(x)$ to identify elements x that have not always been true in the past. The resulting safety problem is defined in the following theorem.

Theorem 5.2. *Let $\Pi = (\iota, \tau, \beta)$ be a safety problem over vocabulary Σ, let w be a fresh constant symbol, and let $\varphi(x)$ be a formula over Σ. Then $\varphi(w)$ is a sound prophecy for Π if and only if the following safety problem, over vocabulary $\Sigma \cup \{m(\cdot)\}$, where $m(\cdot)$ is a fresh relation symbol, is safe:*

$$\Pi^{\text{sound}}_\varphi = (\iota \wedge \forall x.\varphi(x) \rightarrow m(x), \tau \wedge \forall x.(m(x) \wedge \varphi(x) \wedge \varphi'(x)) \rightarrow m'(x), \beta \wedge \forall x.\varphi(x) \rightarrow \neg m(x)).$$

We call a safe inductive invariant of $\Pi^{\text{sound}}_\varphi$ a soundness invariant *for $\varphi(w)$.*

Next, we present the additional inference rule that allows proving a safety problem by first proving that a given prophecy $\varphi(w)$ is sound, and also proving that the safety problem under this prophecy assumption, and with a witness symbol, is safe. The soundness of this inference rule is a direct consequence of Definition 5.1, which implies that for a sound prophecy $\varphi(w)$, a safety problem Π is safe if and only if Π^w_φ is safe; and Theorem 5.2, which guarantees that the premise $\Pi^{\text{sound}}_\varphi$ ensures that the prophecy used in the inference rule is sound.

Definition 5.3. The *incremental forward-backward proof system with prophecy* **FBPI** consists of the proof rules of **FBI** (Definition 4.9), and the additional prophecy proof rule below, where Π is a safety problem over vocabulary Σ, w is a fresh constant symbol, and $\varphi(x)$ is a formula over Σ:

$$\text{(Proph)}\ \frac{\Pi^{\text{sound}}_\varphi \quad \Pi^w_\varphi}{\Pi}$$

For a class of predicates $\mathbb{P}$, we denote by $\mathbf{FBPI}^{\mathbb{P}}$ the system **FBPI** where applications of (Ind) are restricted to $\varphi \in \mathbb{P}$, and by $\mathbf{FBPI}^{\mathbb{P}}_n$ the system where proofs use at most n applications of (Ind).

Theorem 5.4 (Soundness). *Every safety problem that has an* **FBPI***-proof is safe.*

Similarly to how we derived the rules (B-Ind), (B-Cons), and (B-Inc) of **FBI** in Theorem 4.11, it is possible to derive a backward (Proph) rule by time-reversing each of the premises before (Proph) and then time-reversing its conclusion. However, the resulting rule is isomorphic to the original (Proph) rule (with the polarity of m flipped), and is therefore redundant.

Next, we describe how to convert an incremental proof with prophecy into a safe inductive invariant for the same safety problem. As before, the (Ind) rule determines the predicates used in the proof, whereas (Inc) creates conjunctions, and (Rev) creates negations. The crucial change is that when handling the (Proph) proof rule, we take the safe inductive invariant of $\Pi_\varphi^{\text{sound}}$, denoted ξ, and substitute every occurrence of $m(t)$ for some term t with $\psi[t/w]$, where ψ is the safe inductive invariant of Π_φ^w. We denote the result of this operation by $\xi[\psi/m]$, and assume w.l.o.g. that ξ and ψ use distinct variables. Intuitively, this results in an inductive invariant for Π because, since ψ is an inductive invariant for witness elements that always satisfy $\varphi(x)$, $\psi[t/w]$ can be used as a valid interpretation of $m(t)$ to substitute into the soundness invariant ξ while maintaining inductiveness. Moreover, $\xi[\psi/m]$ implies the safety of Π. This is true because the soundness invariant ξ implies that any bad state we reach must have an element that satisfies both $\varphi(x)$ and $m(\cdot)$. Therefore, the resulting invariant $\xi[\psi/m]$ implies that any bad state we reach must have a witness w that satisfies $\varphi(w)$ and ψ. Moreover, ψ implies the safety property of Π_φ^w, which states that if we have a witness w satisfying $\varphi(w)$, then we are not in a bad state. This shows we cannot reach a bad state at all, since if we have a witness w satisfying both $\varphi(w)$ and ψ, we are not in a bad state.

Typically, the soundness invariant ξ would include existential quantifiers since it has to imply the safety property of $\Pi_\varphi^{\text{sound}}$, which is $\beta \to \exists x.\varphi(x) \wedge m(x)$. Moreover, occurrences of $m(\cdot)$ in ξ are likely to be applied to terms $t(x)$ that include these existentially quantified variables. Therefore, when the occurrences of $m(t(x))$ in ξ are substituted by $\psi[t(x)/w]$, the quantifiers of ψ become nested under the quantifiers of ξ. Since the constant w in ψ is substituted by $t(x)$, this is often a real nesting that cannot be broken into a Boolean combination of un-nested quantifiers. The reason is that ψ may include constraints that mix the quantified variables of ψ and w, in which case $\psi[t(x)/w]$ includes constraints that mix the quantified variables of ψ and x. Thus, the use of the (Proph) proof rule may allow us to reduce the number of quantifiers, and even quantifier nesting, that is required in a safe inductive invariant.

Definition 5.5. For every proof $P \in$ **FBPI** define $\text{Inv}^{\rightleftharpoons}_{\text{pro}}(P)$ by induction on proof structure:

- if the root of P is not a conclusion of (Proph), define $\text{Inv}^{\rightleftharpoons}_{\text{pro}}(P)$ like $\text{Inv}^{\rightleftharpoons}(P)$ in Definition 4.12;
- if the root of P is a conclusion of (Proph) with premises $\Pi_\varphi^{\text{sound}}$ and Π_φ^w, whose proofs are P_{sound} and P_w, respectively, let $\xi = \text{Inv}^{\rightleftharpoons}_{\text{pro}}(P_{\text{sound}})$ and $\psi = \text{Inv}^{\rightleftharpoons}_{\text{pro}}(P_w)$. Then, $\text{Inv}^{\rightleftharpoons}_{\text{pro}}(P) = \xi[\psi/m]$, where the notation indicates that we substitute every application $m(t)$ in ξ, for any term t, with the formula $\psi[t/w]$.

Theorem 5.6. *Let $P \in$ **FBPI**$^{\mathbb{P}}$ be an incremental forward-backward proof of a safety problem Π. Then $\text{Inv}^{\rightleftharpoons}_{\text{pro}}(P)$ is a safe inductive invariant of Π, and if P contains $n \in \mathbb{N}$ instances of the (Ind) rule, and the quantification depth in predicates of $\mathbb{P}$ is at most d, then $\text{Inv}^{\rightleftharpoons}_{\text{pro}}(P)$ is a formula with quantification depth at most $n \cdot d$.*

The crux of the proof of Theorem 5.6 is formalized and proved in the extended version [Frenkel et al. 2026b]. Next, consider how the additional quantification depth in computed invariants may arise. To compute the invariant, in each application of the (Proph) inference rule we take an invariant ξ of the premise $\Pi_\varphi^{\text{sound}}$ and an invariant ψ of the premise Π_φ^w, and replace all occurrences of the relation $m(\cdot)$ applied to a term t in ξ with $\psi[t/w]$. Thus, if ξ has quantification depth k, and

ψ has quantification depth ℓ, the resulting formula $\xi[\psi/m]$ has quantification depth of at most $k+\ell$. Meanwhile, the quantification depth of the invariant computed for the (Inc) rule is the maximum of the quantification depths associated with the invariants of its two premises, and the rest of the inference rules have no effect on the quantification depth in the computed invariant at all. Therefore, if n predicates are involved in the computation of the invariant, and each has quantification depth of at most d, then the resulting invariant cannot exceed quantification depth $n \cdot d$.

Example 5.7. Recall the problem $\Pi = (\iota, \tau, \beta)$ with vocabulary $\Sigma = \{a(\cdot), d(\cdot,\cdot)\}$ from Section 2.3:

$$\iota = \exists x.\neg a(x) \wedge \forall x, y.d(x,y) \qquad \beta = \forall x.\exists y.\neg d(x,y)$$
$$\tau = (\forall x.a(x) \leftrightarrow a'(x)) \wedge \exists x_0, y_0.a(x_0) \wedge \forall x, y.d'(x,y) \leftrightarrow (d(x,y) \wedge (x \neq x_0 \vee y \neq y_0))$$

Initial states of this system all contain some *inactive* element, i.e., an element not satisfying $a(\cdot)$, and the interpretation of $a(\cdot)$ remains fixed along transitions. Moreover, in initial states all pairs of elements satisfy $d(\cdot,\cdot)$, but in each transition a pair (x_0, y_0) is removed from the relation, assuming $a(x_0)$ holds. The bad states of the system are those where all elements x have some y that is not in the relation with them. Intuitively, this cannot happen, since for any trace there exists some x where $\neg a(x)$ holds throughout, and for this x no pairs can be removed from the relation, which initially contains (x, y) for every y. This can be proven using the following inductive invariant with quantification depth 2 (with a real nesting of quantifiers): $\exists x.\neg a(x) \wedge \forall y.d(x,y)$.

However, using the proof system **FBPI**, the safety of this system can be proven using predicates with a single existential quantifier only, by introducing a prophecy witness for $\neg a(x)$, where we let $\Pi^{\text{sound}}_{\neg a(x)} = (\iota_m, \tau_m, \beta_m)$ and $\Pi^{w}_{\neg a(x)} = (\iota_w, \tau_w, \beta_w)$:

$$\dfrac{\text{(Cons)}\ \dfrac{\text{(Ind)}\ \dfrac{}{(\iota_m, \tau_m, \neg\exists x.\neg a(x) \wedge m(x))}}{\Pi^{\text{sound}}_{\neg a(x)}} \quad \text{(B-Cons)}\ \dfrac{\text{(B-Ind)}\ \dfrac{}{(\neg\exists y.\neg d(w,y), \tau_w, \beta_w)}}{\Pi^{w}_{\neg a(x)}}}{\Pi}\ \text{(Proph)}$$

We prove the soundness problem $\Pi^{\text{sound}}_{\neg a(x)}$ using an inductive invariant that says there is always an element satisfying both $\neg a(\cdot)$ and $m(\cdot)$, which means that adding a witness with axiom $\neg a(w)$ is sound. We then prove the problem $\Pi^{w}_{\neg a(x)}$ with the added witness via a backward inductive invariant that shows there is some element y that is not in the relation $d(\cdot,\cdot)$ with w. This is indeed a backward inductive invariant since it is implied by β, and since backward transitions cannot change $\neg a(w)$ or $r(x,y)$ for any x with $\neg a(x)$, backward consecution also holds. Lastly, if we denote this proof by P, we get $\text{Inv}^{\rightleftharpoons}_{\text{pro}}(P) = \exists x.\neg a(x) \wedge \neg\exists y.\neg d(x,y)$, which is equivalent to the safe inductive invariant $\exists x.\neg a(x) \wedge \forall y.d(x,y)$ we considered before.

Finally, the overall results about the proof power of **FBPI** are summarized as follows.

Theorem 5.8. *For every $\mathbb{P}$ whose predicates have quantification depth of at most $d \in \mathbb{N}$, all safety problems that are provable in $\mathbf{FBPI}^{\mathbb{P}}_n$ are also provable in $\mathbf{F}^{\mathbb{Q}_{n\cdot d}}$, where $\mathbb{Q}_{n\cdot d}$ consists of all predicates with quantification depth at most $n \cdot d$. However, for every $n > 0$ there exist $\mathbb{P}$ and a safety problem Π such that Π is provable in $\mathbf{FBPI}^{\mathbb{P}}_n$ but not provable in either $\mathbf{FBPI}^{\mathbb{P}}_{n-1}$, $\mathbf{FBI}^{\mathbb{P}}$, $\mathbf{FI}^{\mathbb{P}}$, or $\mathbf{F}^{\mathbb{P}}$.*

Proof. The first part of the theorem is a direct corollary of Theorem 5.6. For the second part, define the vocabulary with one unary relation and $n-1$ binary relations $\Sigma = \{p_1(\cdot), r_2(\cdot,\cdot), \ldots, r_n(\cdot,\cdot)\}$, and the set of predicates $\mathbb{P} = \{\exists x_1.p(x_1) \wedge m(x_1)\} \cup \{\exists x_i.r_i(w_i, x_i) \wedge m(x_i)\}_{i=2}^{n}$. Denote

$$\varphi = \exists x_1.p_1(x_1) \wedge \forall x_2.\neg r_2(x_1,x_2) \vee \exists x_3.r_3(x_2,x_3) \wedge \forall x_4.\neg r_4(x_3,x_4) \vee \cdots$$

and consider the safety problem $\Pi = (\varphi, \bot, \neg\varphi)$. This safety problem is provable in $\mathbf{FBPI}^{\mathbb{P}}_n$, by first adding a witness w_2 with axiom $p_1(w_2)$ in the forward direction, then a witness w_3 with axiom

$r_2(w_2, w_3)$ in the backward direction, then a witness w_4 with axiom $r_3(w_3, w_4)$ in the forward direction, and so on. If we prove the soundness of the prophecies using the appropriate soundness invariants, the safe inductive invariant computed for this proof via $\mathrm{Inv}^{\rightleftharpoons}_{\mathrm{pro}}(\cdot)$ is exactly equivalent to φ. However, Π is not provable in $\mathbf{FBPI}^{\mathbb{P}}_{n-1}$, $\mathbf{FBI}^{\mathbb{P}}$, $\mathbf{FI}^{\mathbb{P}}$, or $\mathbf{F}^{\mathbb{P}}$, since any safe inductive invariant must be equivalent to φ, and there exists no such formula with quantification depth of less than n. □

5.2 Prophecy Heuristics

When using the forward-backward proof system without prophecy (Definition 4.9), one can approach the task of constructing a proof, either manually or automatically, by iteratively attempting to find meaningful forward- or backward-inductive invariants and adding them as axioms to the safety problem, then proceeding in the same fashion while exploring different proof depths. Proving the soundness of each step is straightforward since it involves a simple induction query. However, when the proof system also allows prophecy (Definition 5.3), the soundness premise of the (Proph) rule might require its own inductive invariant or a more complicated proof structure to prove. Therefore, coming up with an appropriate prophecy and proving its associated soundness problem introduces a new challenge in terms of proof search.

In this section we present a heuristic approach for narrowing the search space for prophecies and their soundness invariants. We propose to do so by recognizing patterns of certain sound prophecies, and providing explicit verification conditions for each pattern, which can be checked without explicitly constructing the soundness problem for the prophecy and attempting a direct proof. The idea is that, while our proof system is designed to allow general safety proofs, in practice we might want to restrict the kind of prophecies we use to simplify verifying their soundness.

For instance, consider the prophecy used in Example 5.7. We can generalize this kind of prophecy to any formula $\varphi(x)$ which for every initial state holds on some element, i.e., $\iota \Rightarrow \exists x.\varphi(x)$, and is always forward-preserved, i.e., $\varphi(x) \wedge \tau \Rightarrow \varphi'(x)$. If these conditions hold, $\varphi(w)$ is obviously sound, and we would like to add it without exploring proof possibilities for its associated soundness problem. In other words, we would like to use the following inference rule directly:

$$\text{(Proph-Fwd)}\ \frac{(\iota, \tau, \beta)^w_\varphi}{(\iota, \tau, \beta)}\ \iota \Rightarrow \exists x.\varphi(x),\ \varphi(x) \wedge \tau \Rightarrow \varphi'(x)$$

And indeed, we can use this rule directly by deriving (Proph-Fwd) once for all prophecies of this kind, relying on the verification conditions above. We derive (Proph-Fwd) for a problem $\Pi = (\iota, \tau, \beta)$, whose premise is Π^w_φ, by denoting $\Pi^{\mathrm{sound}}_\varphi = (\iota_m, \tau_m, \beta_m)$ and deriving:

$$\dfrac{\dfrac{\text{(Ind)}\ \dfrac{}{(\iota_m, \tau_m, \neg\exists x.\varphi(x) \wedge m(x))}}{\text{(Cons)}}\ \Pi^{\mathrm{sound}}_\varphi}{\text{(Proph)}}\ \Pi^w_\varphi}{\Pi}$$

Using this approach, different classes of prophecy can be described and their explicit verification conditions written down and verified. These can be used in proofs as prophecy short-cuts via the appropriate derived inference rules, e.g., (Proph-Fwd) for forward-preserved prophecy formulas. When searching for a proof, these can be used to constrain the kinds of prophecy considered, and to supply explicit verification conditions for each.

We present one more prophecy pattern which turns out to be central to our evaluation in Section 6.2. This prophecy considers a predicate θ that is forward-preserved, and thus partitions any trace of the program into an initial part where all states satisfy $\neg\theta$, and a tail where all elements satisfy θ. This must be the case because once θ holds at some point in the trace it must hold thereafter, since it is forward-preserved. The added prophecy $\varphi(x)$ has the property that it holds

for all elements before θ holds; that it is forward preserved for any individual element once θ holds; and that $\exists x.\varphi(x)$ is itself an inductive invariant. Intuitively, we can think of $\varphi(x)$ as a property that initially holds for all elements of a state until θ is activated, upon which (at least) one element is "selected" and continues to satisfy $\varphi(x)$ the entire length of the trace. The verification conditions detailed above translate to the following inference rule:

$$\text{(Proph-Select)}\ \frac{(\iota,\tau,\beta)^w_\varphi}{(\iota,\tau,\beta)} \quad \begin{array}{c} \iota \Rightarrow \exists x.\varphi(x),\ \exists x.\varphi(x) \wedge \tau \Rightarrow \exists x.\varphi'(x) \\ \neg\theta \Rightarrow \forall x.\varphi(x),\ \theta \wedge \tau \Rightarrow \theta', \\ \varphi(x) \wedge \theta \wedge \tau \Rightarrow \varphi'(x) \end{array}$$

which can be derived in **FBPI**, using similar notation to before:

$$\begin{array}{l} \text{(Ind)} \\ \text{(Cons)} \end{array} \frac{}{(\iota_m,\tau_m,\neg((\neg\theta \wedge \forall x.m(x)) \vee (\theta \wedge \exists x.\varphi(x) \wedge m(x))))} \qquad \text{(Proph)}\ \frac{\Pi^{\text{sound}}_\varphi \qquad \Pi^w_\varphi}{\Pi}$$

Lastly, note that for each of the two prophecy rules, we do not only get explicit verification conditions, but also an explicit way to construct safe inductive invariants from safety proofs that use them. For example, if we add a prophecy $\varphi(x)$ using the (Proph-Fwd) rule and get an invariant ψ for the problem with the prophecy Π^w_φ, then a safe inductive invariant for Π is $\exists x.\varphi(x) \wedge \psi[x/w]$. Similarly, if we use the (Proph-Select) rule for adding prophecy, the resulting safe inductive invariant shall be $(\neg\theta \wedge \forall x.\psi[x/w]) \vee (\theta \wedge \exists x.\varphi(x) \wedge \psi[x/w])$.

6 Case Study: Proofs of Paxos Variants and Raft

We now demonstrate that the proof techniques developed in Sections 4 and 5 lead to simpler proofs of the Paxos consensus protocol [Lamport 1998, 2001], several of its variants, and Raft [Ongaro and Ousterhout 2014]. Both Paxos and Raft are central distributed consensus protocols that have drawn the attention of the verification community. Our incremental proofs use simpler predicates compared to the standard proofs that use a single (forward) inductive invariant. This simplification of the proofs could be advantageous for automated proof search, as it narrows the space of potential invariant formulas. While our case study shows promising results for several complex consensus protocols, we acknowledge that there is no guarantee that similar simplifications will occur in other protocols. We hope that the evidence presented here will motivate the community to apply our framework to complex protocols beyond those considered here. We begin with a detailed illustration using single-decree Paxos, and follow with results from more complex variants. All proofs reported here are available in the paper's artifact [Frenkel et al. 2026a] and included as examples in the `mypyvy` tool [Wilcox et al. 2024].

6.1 Proof Simplification for Single-Decree Paxos

We start our investigation with the first-order logic model of single-decree Paxos listed in Figure 1. This model is a slightly simplified version of the model from [Padon et al. 2017].[1] The key safety property we wish to prove is that only a single value can be decided. Paxos ensures this by operating in *rounds*, where each node remembers its current round, and does not participate in lower rounds. A decision requires a majority (quorum) of votes on a proposed value, and proposing a value in a new round requires a quorum of nodes that join that round. When a node joins a round, it reports its latest vote so far (i.e., that in the highest round), and the proposed value must correspond to the

[1]We abstract away the "1a" or "start round" messages since they are not important for the proof. We also state the safety property in terms of fresh constants rather than universal quantification (a form of Herbrandization) and focus on the case of consistency between two different rounds, assuming $r_1 < r_2$. Interestingly, the Herbrandization leads to a proof in EPR of the first-order model of Paxos without requiring the rewrites developed in [Padon et al. 2017].

```
1  sort node, quorum, round, value
2
3  relation ≤ : round, round
4  axiom Γ_{total order}[≤]
5
6  relation member : node, quorum
7  axiom ∀q1, q2 : quorum. ∃n : node.
8                member(n, q1) ∧ member(n, q2)
9
10 function current_round : node → round
11 relation join_msg : node, round, round, value
12 relation join_msg_no_vote : node, round
13 relation propose_msg : round, value
14 relation vote_msg : node, round, value
15 relation decision : round, value
16
17 init ∀n, r1, r2, v. ¬join_msg(n, r1, r2, v)
18 init ∀n, r. ¬join_msg_no_vote(n, r)
19 init ∀r, v. ¬propose_msg(r, v)
20 init ∀n, r, v. ¬vote_msg(n, r, v)
21 init ∀n, r, v. ¬decision(n, r, v)
22
23 individual r1, r2 : round
24 individual v1, v2 : value
25 axiom r1 < r2 ∧ v1 ≠ v2
26 safety ¬(decision(r1, v1) ∧ decision(r2, v2))
```

```
27 action JOIN_ROUND(n : node, r : round) {
28   require current_round(n) < r
29   if ∀r, v. ¬vote_msg(n, r, v) then {
30     join_msg_no_vote(n, r) := true
31   } else {
32     var maxr, v := max{(r', v') | vote_msg(n, r', v')}
33     join_msg(n, r, maxr, v) := true
34   }
35   current_round(n) := r
36 }
37 action PROPOSE(r : round, q : quorum) {
38   require ∀v. ¬propose_msg(r, v)∧
39             ∀n. member(n, q) →
40               join_msg_no_vote(n, r) ∨ ∃r', v'. join_msg(n, r, r', v')
41   var maxr, v :=                    # v is arbitrary if the set is empty
42         max{(r', v') | ∃n. member(n, q) ∧ join_msg(n, r, r', v')}
43   propose_msg(r, v) := true
44 }
45 action VOTE(n : node, r : round, v : value) {
46   require propose_msg(r, v) ∧ r = current_round(n)
47   vote_msg(n, r, v) := true
48 }
49 action LEARN(r:round, v:value, q:quorum) {
50   require ∀n. member(n, q) → vote_msg(n, r, v)
51   decision(r, v) := true
52 }
```

Fig. 1. Model of Paxos consensus algorithm as a transition system in many-sorted first-order logic.

latest vote reported by a quorum. This ensures that a proposal at a higher round cannot conflict with a decision at a lower round, including decisions that happen after the proposal was made (using a different quorum). See [Lamport 2001; Padon et al. 2017] for more details.

We now show how a forward-backward proof, whose proof tree is given in Figure 2a, can simplify the predicates needed in the proof of Paxos. The key source of simplification is the invariant that states that if a value is proposed in a round, then no other value can be *choosable* at a lower round. Choosable here means that it is possible that the value will become decided in the future. That is, that there is a quorum of nodes all of which have either voted for that value or can still vote for it, i.e., they have not yet joined a higher round. In the context of the model listed in Figure 1, where we assume $r_1 < r_2$ and $v_1 \neq v_2$, this invariant is formalized as follows:

$$\begin{aligned}&\forall r : \text{round}.\ r_1 < r \wedge \mathit{propose_msg}(r, v_2) \rightarrow \\ &\quad \forall q : \text{quorum} \exists n : \text{node}.\ \mathit{member}(n, q) \wedge \neg \mathit{vote_msg}(n, r_1, v_1) \wedge \mathit{current_round}(n) > r_1.\end{aligned} \tag{1}$$

That is, the invariant states that if v_2 is proposed at any round greater than r_1, then v_1 is no longer choosable at r_1 (i.e., in every quorum there is a node that is at a round greater than r_1 and did not vote for v_1 at r_1). Of all of the invariants in the proof of Paxos, this one has the most complex Boolean structure. Note that it is not a clause because of the alternation of $\vee$ and $\wedge$. Interestingly, a forward-backward proof can replace this invariant with simpler predicates. The key point is that in a forward-only proof, we must represent the disjunction between v_2 being proposed (at any round greater than r_1) and v_1 being choosable (at r_1). However, in a forward-backward proof we can eliminate the disjunction, since we know that in a bad state there must be a decision on v_1 at r_1, so it must be choosable at every point along the path from init to bad.

$$\dfrac{\dfrac{}{(\iota, \tau, \neg\varphi_1)}\,(\text{Ind})\ \dfrac{\dfrac{}{(\neg\varphi_2, \tau \wedge \varphi_1 \wedge \varphi_1', \beta \wedge \varphi_1)}\,(\text{B-Ind}) \qquad \dfrac{\dfrac{}{(\iota \wedge \varphi_1 \wedge \varphi_2, \tau \wedge \varphi_1 \wedge \varphi_1' \wedge \varphi_2 \wedge \varphi_2', \neg\varphi_3)}\,(\text{Ind})}{(\iota \wedge \varphi_1 \wedge \varphi_2, \tau \wedge \varphi_1 \wedge \varphi_1' \wedge \varphi_2 \wedge \varphi_2', \beta \wedge \varphi_1 \wedge \varphi_2)}\,(\text{Cons})}{(\iota \wedge \varphi_1, \tau \wedge \varphi_1 \wedge \varphi_1', \beta \wedge \varphi_1)}\,(\text{B-Inc})}{(\iota, \tau, \beta)}\,(\text{Inc})$$

(a) Proof tree of forward-backward proof.

$$\dfrac{\dfrac{\dfrac{}{(\neg\varphi_2^q, \tau \wedge \varphi_1^q \wedge \varphi_1^{q\prime}, \beta \wedge \varphi_1^q)}\,(\text{B-Ind}) \qquad \dfrac{\dfrac{}{(\iota \wedge \varphi_1^q \wedge \varphi_2^q, \tau \wedge \varphi_1^q \wedge \varphi_1^{q\prime} \wedge \varphi_2^q \wedge \varphi_2^{q\prime}, \neg\varphi_3)}\,(\text{Ind})}{(\iota \wedge \varphi_1^q \wedge \varphi_2^q, \tau \wedge \varphi_1^q \wedge \varphi_1^{q\prime} \wedge \varphi_2^q \wedge \varphi_2^{q\prime}, \beta \wedge \varphi_1^q \wedge \varphi_2^q)}\,(\text{Cons})}{(\iota \wedge \varphi_1^q, \tau \wedge \varphi_1^q \wedge \varphi_1^{q\prime}, \beta \wedge \varphi_1^q)}\,(\text{B-Inc})}{(\iota, \tau, \beta)}\,(\text{Proph-Select})\ [\theta = \mathit{decision}(r_1, v_1)]$$

(b) Proof tree of forward-backward proof with prophecy.

Fig. 2. Proof trees for incremental safety proofs of the Paxos protocol in Figure 1, with side conditions omitted for simplicity. The triple (ι, τ, β) denotes the safety problem defined by the protocol. Formulas φ_1 and φ_2 are eqs. (2) and (3), respectively, and φ_3 consists of eq. (4) and a few additional invariants which we omit to simplify the presentation. Formulas φ_1^q and φ_2^q are identical to eqs. (2) and (3), respectively, except q is not quantified and is instead introduced as a fresh prophecy witness by (Proph-Select), which uses $\theta = \mathit{decision}(r_1, v_1)$.

Formally, we first argue by a forward proof step that a decision is supported by a quorum of votes (as is done in the forward-only proof), focusing on r_1 and v_1:

$$\mathit{decision}(r_1, v_1) \rightarrow \exists q : \text{quorum}. \forall n : \text{node}.\ \mathit{member}(n, q) \rightarrow \mathit{vote_msg}(n, r_1, v_1). \tag{2}$$

This step corresponds to the first application of (Inc) in Figure 2a, starting from the root, where φ_1 is eq. (2) above. This proof rule results in a left-hand premise proved by forward induction, and a right-hand premise where eq. (2) is assumed as an axiom (see Definition 4.3).

Next, to prove the right-hand premise, we argue with a backward proof step that on any path to the bad states v_1 must always be choosable at r_1:

$$\exists q : \text{quorum} \forall n : \text{node}.\ \mathit{member}(n, q) \wedge \mathit{current_round}(n) > r_1 \rightarrow \mathit{vote_msg}(n, r_1, v_1). \tag{3}$$

This step corresponds to the application of (B-Inc) in Figure 2a, where φ_2 is eq. (3) above, and the left-hand premise is proved via backward induction.

Finally, using a forward step we can show that in the transition system resulting from assuming eqs. (2) and (3), v_2 cannot be proposed at any round greater than r_1:

$$\forall r : \text{round}.\ r_1 < r \rightarrow \neg \mathit{propose_msg}(r, v_2). \tag{4}$$

This easily leads to the ultimate safety proof. Formally, this last step corresponds to proving the forward inductive invariant φ_3 in Figure 2a, which consists of eq. (4) and a few supporting invariants.

All in all, we replaced the forward proof with a forward-backward proof that uses less complex auxiliary invariants. The Boolean structure of eqs. (2) to (4) is simpler than that of eq. (1). This might be helpful for proof automation, as the search space for clausal invariants is smaller.

Using prophecy, the invariants can be simplified even further, eliminating the existential quantifier in eq. (3) and replacing it with a witness obtained for the property expressed in eq. (2). That is, using prophecy, the first forward step in the proof can be replaced by the introduction of a witness to the quorum that ultimately leads to the decision on v_1. Then, in the backward step, we can state that v_1 must remain choosable using the same quorum, which is expressed using a purely universally quantified formula. That is, the forward invariant gives us a witness that makes the

backward invariant simpler. We note that while the use of prophecy variables in general is not a new proof technique, the synergy between forward-backward reasoning and prophecy is new.

Formally, the quorum that comes from eq. (2) is introduced as a prophecy witness using the rule (Proph-Select) (Section 5.2) with the prophecy $\varphi_1^q = \forall n : \text{node}.\ \mathit{decision}(r_1, v_1) \wedge \mathit{member}(n, q) \rightarrow \mathit{vote_msg}(n, r_1, v_1)$ and $\theta = \mathit{decision}(r_1, v_1)$. The proof tree using prophecy is given in Figure 2b.

6.2 Application to Paxos Variants and Raft

Similarly to the simplification from Section 6.1, we use the incremental proof systems developed in Sections 4 and 5 to prove the safety of various Paxos variants and an abstract model of Raft. We compare the relative complexity of user-specified predicates used in each proof method in terms of Boolean structure, the number of quantifiers and quantifier alternations required, as well as the number of prophecy symbols used by each proof.

We consider several Paxos variants modeled in [Padon et al. 2017]: Paxos, Flexible Paxos, Multi-Paxos, and Fast Paxos, all modeled in the effectively propositional (EPR) fragment of first-order logic; as well as the Paxos FOL variant presented (in simplified form) in Section 6. These Paxos variants have complex inductive invariants that are highly challenging to find automatically. This challenge drove the development of several recent invariant inference techniques, e.g., [Frenkel et al. 2024; Hance et al. 2021; Koenig et al. 2022; Yao et al. 2022]. Some of these techniques are able to find the required invariants automatically, but in some cases the search involves long runtimes (on the order of hours), due to the complexity of the invariants and the resulting search space.

In addition to the standard variants of the original Paxos examples from [Padon et al. 2017], we extend our example set with versions of these which differ in the arity of the *decision* relation of the protocol. In most of the original examples, this relation is modeled to indicate both which value was decided upon, and in which round. We introduce additional versions where the *decision* relation indicates only the decided value, as well as versions where it indicates the decided value, in which round the decision took place, and which quorum of nodes was responsible for the decision. These alternative versions, which differ slightly in the level of abstraction and the complexity of required invariants, allow us to demonstrate the different proof power of the proof systems considered.

As for Raft, we consider the abstract model from [Taube et al. 2018a]. The invariant for this EPR modeling of the protocol is significantly more complex than the ones for Paxos, and it has not been found automatically by the above-mentioned tools. Other works considered variants or specific parts of Raft and automated some aspects of the proof [Schultz et al. 2022; Zhang et al. 2025b].

As a preprocessing step, we rewrite the safety property of all examples using fresh immutable constants rather than universal quantification, similarly to the example in Section 6.1.

The complexity of user-specified predicates in the proof of each example for different proof systems is detailed in Table 1. For Paxos examples, the suffix of the name of each example indicates the arity of the *decision* relation. In all proofs we consider predicates that are conjunctions of formulas with a certain number of quantifiers (Q), quantifier alternations (A), and Boolean connectives (B). We indicate whether the formulas are purely clausal (contain only disjunctions). Since the class of predicates we consider is closed under conjunction, all forward proofs require no incrementality. State-of-the-art invariant inference tools, e.g., P-FOL-IC3 [Koenig et al. 2022] and DuoAI [Yao et al. 2022], produce invariants of similar complexity to the described forward proofs for their benchmarks. For forward-backward proofs, we consider a sequential proof structure which incrementally adds a predicate in each step which is either forward-inductive or backward-inductive (relative to the predicates added so far), similarly to the proof in Figure 2a. For forward-backward proofs with prophecy, proofs follow a similar pattern, but each step may also be a prophecy step where a sound prophecy is added instead of an inductive invariant using the (Proph-Select) heuristic presented in Section 5.2, similarly to the proof in Figure 2b. We indicate in parentheses how many prophecy

Table 1. Parameters of user-specified predicates involved in the proof of various Paxos variants and Raft. Q = number of quantifiers, A = number of quantifier alternations, B = number of Boolean connectives, p = number of prophecy symbols. The class of predicates used consist of closing these under conjunction, so forward proofs are always of length 1. We indicate whether the predicates in the proof are clausal, i.e., include only disjunctions. For forward-backward proofs, we write the parameters of each iteration in a separate line, denoting forward and backward steps by **F** and **B**, respectively, or **FP** and **BP**, respectively, when prophecy is used. A left arrow (⟵) means there is no additional simplification so the proof is identical to the one on the left. The reported verification time for each proof technique is computed as the median runtime of 15 runs.

Example	**Forward**			**Forward-Backward**			**Forward-Backward + Prophecy**		
	Predicates	Clausal	Time	Predicates	Clausal	Time	Predicates	Clausal	Time
paxos-epr-v	Q=4, A=1, B=5	×	0.59 s	⟵			**FP:** Q=1, A=0, B=2, p=2 **FP:** Q=1, A=0, B=2, p=2 **B:** Q=1, A=0, B=2 **F:** Q=2, A=0, B=2	✓	0.43 s
paxos-epr-rv	Q=3, A=1, B=5	×	0.46 s	**F:** Q=2, A=1, B=2 **B:** Q=2, A=1, B=2 **F:** Q=2, A=0, B=2	✓	0.45 s	**FP:** Q=1, A=0, B=2, p=1 **FP:** Q=1, A=0, B=2, p=1 **B:** Q=1, A=0, B=2 **F:** Q=2, A=0, B=2	✓	0.45 s
paxos-epr-rvq	Q=2, A=0, B=5	×	0.38 s	**F:** Q=1, A=0, B=2 **B:** Q=1, A=0, B=2 **F:** Q=2, A=0, B=2	✓	0.4 s	⟵		
flexible-paxos-epr-v	Q=4, A=1, B=5	×	0.54 s	⟵			**FP:** Q=1, A=0, B=2, p=2 **FP:** Q=1, A=0, B=2, p=2 **B:** Q=1, A=0, B=2 **F:** Q=2, A=0, B=2	✓	0.43 s
flexible-paxos-epr-rv	Q=3, A=1, B=5	×	0.45 s	**F:** Q=2, A=1, B=2 **B:** Q=2, A=1, B=2 **F:** Q=2, A=0, B=2	✓	0.45 s	**FP:** Q=1, A=0, B=2, p=1 **FP:** Q=1, A=0, B=2, p=1 **B:** Q=1, A=0, B=2 **F:** Q=2, A=0, B=2	✓	0.44 s
flexible-paxos-epr-rvq	Q=2, A=0, B=5	×	0.37 s	**F:** Q=1, A=0, B=2 **B:** Q=1, A=0, B=2 **F:** Q=2, A=0, B=2	✓	0.4 s	⟵		
multi-paxos-epr-nirv	Q=3, A=1, B=5	×	0.56 s	**F:** Q=2, A=1, B=2 **B:** Q=2, A=1, B=2 **F:** Q=3, A=0, B=2	✓	0.54 s	**FP:** Q=1, A=0, B=2, p=1 **FP:** Q=1, A=0, B=2, p=1 **B:** Q=1, A=0, B=2 **F:** Q=3, A=0, B=2	✓	0.5 s
fast-paxos-epr-v	Q=5, A=1, B=7	×	1.5 s	⟵			**FP:** Q=1, A=0, B=3, p=3 **FP:** Q=1, A=0, B=3, p=3 **B:** Q=1, A=0, B=3 **F:** Q=2, A=0, B=3	✓	0.8 s
fast-paxos-epr-rv	Q=3, A=1, B=7	×	1.2 s	**F:** Q=2, A=1, B=3 **B:** Q=2, A=1, B=3 **F:** Q=2, A=0, B=3	✓	1 s	**FP:** Q=1, A=0, B=3, p=2 **FP:** Q=1, A=0, B=3, p=2 **B:** Q=1, A=0, B=3 **F:** Q=2, A=0, B=3	✓	0.81 s
paxos-fol-v	Q=5, A=1, B=5	×	1.6 s	⟵			**FP:** Q=1, A=0, B=2, p=2 **FP:** Q=1, A=0, B=2, p=2 **B:** Q=1, A=0, B=2 **F:** Q=4, A=0, B=3	✓	0.62 s
paxos-fol-rv	Q=4, A=1, B=5	×	0.65 s	**F:** Q=2, A=1, B=2 **B:** Q=2, A=1, B=2 **F:** Q=4, A=0, B=3	✓	0.65 s	**FP:** Q=1, A=0, B=2, p=1 **FP:** Q=1, A=0, B=2, p=1 **B:** Q=1, A=0, B=2 **F:** Q=4, A=0, B=3	✓	0.6 s
paxos-fol-rvq	Q=4, A=0, B=5	×	0.52 s	**F:** Q=1, A=0, B=2 **B:** Q=1, A=0, B=2 **F:** Q=4, A=0, B=3	✓	0.54 s	⟵		
raft-epr	Q=7, A=2, B=9	×	3.7 s	⟵			**F:** Q=7, A=0, B=8 **FP:** Q=1, A=0, B=2, p=3 **FP:** Q=1, A=0, B=2, p=3 **B:** Q=1, A=0, B=2 **F:** Q=3, A=0, B=5 **F:** Q=3, A=0, B=5	✓	2.8 s

symbols were added in each step (p). We use a straightforward generalization of the prophecy proof rule presented in Section 5.1 which allows adding multiple prophecy symbols in each step.

All proofs were manually written and mechanically checked using `mypyvy` [Wilcox et al. 2024], which discharges logical queries to the Z3 [De Moura and Bjørner 2008] SMT solver. To check the proofs, `mypyvy` was extended with syntax and a proof checker for forward-backward proofs with prophecy of the sequential nature described above. This extension has been incorporated into the mypyvy tool, and is also available in the paper's artifact [Frenkel et al. 2026a]. Table 1

presents the verification time for each considered proof, computed as the median of 15 runs. All experiments were carried out on a `z1d.large` AWS machine, which has 2 cores and 16GB of RAM. The timing results show that for this case study, writing proofs in our proposed proof system noticeably reduces verification times for the more difficult examples, and matches verification times for the examples that are verified relatively quickly (in less than a second).

As can be seen in Table 1, for most examples, a forward-backward proof greatly simplifies the Boolean structure of predicates considered, by enabling proofs to use only clausal formulas (rather than ones containing both conjunctions and disjunctions). As expected, there is no reduction in the quantifier alternations required in forward-backward proofs compared to forward proofs. Interestingly, Raft and the Paxos variants that contain only values in the *decision* relation do not seem to have forward-backward proofs that simplify forward-only proofs, due to quantification that prevents decomposing non-clausal invariants. However, using prophecy steps, we can avoid this quantification and again prove safety using only conjunctions of universally quantified clauses.

On the other hand, in the round-value Paxos variants, forward-backward proofs simplify the Boolean structure of invariants; and adding prophecy allows the proof to avoid all existential quantification in user-specified predicates. In fact, in all forward-backward proofs in Table 1, we only ever need to use prophecy in the forward direction, and only in the beginning of each proof. Note that since we use the (Proph-Select) heuristic, only θ is specified by the user, which is quantifier-free in our examples, leading to proofs that only involve universal, clausal predicates. However, the soundness invariant for the prophecy, which is automatically constructed from θ, contains existential quantification and quantifier alternations.

Lastly, we observe that including the deciding quorum in the decision relation of these protocols simplifies even the forward invariants to require no alternation. However, these still include existential quantification, and the predicates in these proofs are non-clausal. Forward-backward proofs simplify these to purely universally quantified clauses, and consequently, these examples cannot be simplified further using prophecy.

7 Related Work and Concluding Remarks

The idea of using a backward analysis to strengthen a forward analysis was first proposed theoretically by Patrick Cousot.[2] It has since been applied in a variety of ways. For example, in symbolic model checking, a representation of the reachable states of a system, computed by a forward analysis, is typically used as a constraint on the backward fixed point iterations when evaluating future-time temporal operators [McMillan 1992]. (This results in a simplification of the representation of the fixed point approximations, though it does not increase precision, since neither the forward nor the backward analyses are approximate.) In [Cousot et al. 2012] iterated forward and backward analysis is used to strengthen contracts inferred for methods of sequential programs created by refactoring. In [Vizel et al. 2013] forward and backward interpolants are used for reachability analysis. In [Bakhirkin and Monniaux 2017] a forward (top-down) and backward (bottom-up) collecting semantics for Horn clauses are alternated to verify sequential programs that cannot be verified using either analysis alone, and it is found that multiple alternations are sometimes needed. A more recent method [Fang et al. 2025] uses an approximate backward analysis to strengthen the forward construction of an abstract reachability tree using Craig interpolants as in [McMillan 2006]. None of these works address the problem of quantification, however.

By contrast, in this paper we consider the possible advantages of forward-backward analysis in the particular case of quantified invariants of parameterized protocols. Forward-backward analysis has not been considered in the protocol verification literature. It has been observed [Koenig et al.

[2] *cf.* [Cousot 1978] pp. (3)-41,42.

2022] that the lack of a conjunctive normal form greatly increases the search space for invariant formulas when existential quantifiers are needed. Forward-backward invariant generation can address this problem in two ways. First, in the alternation-free case, we find that a forward-backward approach can in practice reduce the space of invariants to only CNF formulas, despite the presence of existential quantifiers. Second, we find that the approach combines well with the addition of prophecy variables, since *(a)* this transformation can eliminate quantifier alternations, and *(b)* we can infer useful prophecy variables for forward analysis from backward analysis, and *vice versa*.

We can compare this approach to other methods of constraining the invariant search for protocol invariants. One approach is to bias the search toward certain forms that are hypothesized to occur commonly in practice. For example, in [Koenig et al. 2022], formulas of the form $\forall X.\ (C \rightarrow \exists Y.D)$, where X, Y are variable sets and C, D are cubes. This technique is effective for consensus protocols, since it captures the form of an important invariant, but as a heuristic it is possibly over-fitted to such protocols. By using forward-backward analysis, we can achieve a similar effect in a way that is more principled and possibly more general. Many other techniques that can synthesize mixed universal/existential invariants are also based on ad-hoc syntactic biases. These include [Schultz et al. 2024; Yao et al. 2022] which are based on reachable state sampling and [Hance et al. 2021] which is based on brute-force enumeration of a template space with SMT-based checking. In principle, such methods could benefit from the reduction of formula complexity obtainable with a forward-backward approach. In other words, the forward-backward approach may allow a stronger syntactic bias. For the sampling methods, it would be necessary to sample backward reachable states, which might present computational challenges. For property-driven invariant generation methods, such as [Koenig et al. 2020, 2022], there is a different challenge, since these methods cannot generate an invariant in the absence of a property to be proved. It is not clear what property should be proved in any forward or backward pass except the last. Still, there would be no difficulty in using a property-driven method in the last pass and a syntax-guided method in earlier passes. In any event, the benefit of forward-backward analysis in this application is largely orthogonal to the method of analysis and has the potential to improve various techniques that apply syntactic biases to invariant generation.

History and prophecy variables have very wide application in program verification. History variables were introduced by Clint [Clint 1973] to solve a problem in using Hoare-style reasoning for coroutines. Prophecy variables were introduced by Abadi and Lamport [Abadi and Lamport 1988] to resolve non-determinism in proofs of refinement, and are still widely used in that context [Lamport and Merz 2022]. The idea that history and prophecy variables can be used to simplify the quantifier structure of invariant formulas may be considered folklore. Uncovering the first use of the idea is difficult but, for example, history and prophecy variables are used in [McMillan 2001] to eliminate quantifiers from the invariant of a cache coherence protocol. History and prophecy are explicitly introduced as a method to eliminate quantifiers in [McMillan 2018]. History variables are also explicitly used for this purpose in [Taube et al. 2018b] in the form of abstract modules. More recent work automatically introduces first-order prophecy variables to eliminate universal quantifiers [Mann et al. 2021; Vick and McMillan 2023].

What is primarily novel here, relative to the above-mentioned works is that we consider the interaction of history/prophecy with a forward-backward proof heuristic (note that history variables might be considered prophecy variables of the time-reversed system). We also formalize the side condition for adding prophecy as an auxiliary safety problem. We observe that existential invariants in one direction can be used as a guide to introduce prophecy variables in the other direction. Moreover, this transformation reduces the search space for forward or backward invariants by eliminating quantifier alternations. Thus, the two methods can be seen as mutually reinforcing.

Data Availability

The data and code supporting this paper are available in the paper's artifact [Frenkel et al. 2026a] and as part of the mypyvy tool [Wilcox et al. 2024].

Acknowledgments

We thank James R. Wilcox for developing the mypyvy tool and for his assistance in integrating our proof checker into it. We thank Raz Lotan and the anonymous reviewers for their helpful and insightful comments. The research leading to these results has received funding from the European Research Council under the European Union's Horizon 2020 research and innovation programme (grant agreement No [759102-SVIS]). This research was partially supported by Israel Science Foundation (ISF) grant No. 2117/23. This research was partially supported by Israel Science Foundation research grant (ISF's No. 4136/25) and the Maimonides Fund's Future Scientists Center, a research grant from the Center for New Scientists at the Weizmann Institute of Science, and a grant from the Azrieli Foundation.

References

Martin Abadi and Leslie Lamport. 1988. The Existence of Refinement Mappings. In *Proceedings of the 3rd Annual Symposium on Logic in Computer Science* (proceedings of the 3rd annual symposium on logic in computer science ed.). 165–175. https://www.microsoft.com/en-us/research/publication/the-existence-of-refinement-mappings/ LICS 1988 Test of Time Award.

Martín Abadi and Leslie Lamport. 1993. Composing Specifications. *ACM Trans. Program. Lang. Syst.* 15, 1 (1993), 73–132. doi:10.1145/151646.151649

Martín Abadi and Leslie Lamport. 1995. Conjoining Specifications. *ACM Trans. Program. Lang. Syst.* 17, 3 (1995), 507–534. doi:10.1145/203095.201069

Alexey Bakhirkin and David Monniaux. 2017. Combining Forward and Backward Abstract Interpretation of Horn Clauses. In *Static Analysis - 24th International Symposium, SAS 2017, New York, NY, USA, August 30 - September 1, 2017, Proceedings (Lecture Notes in Computer Science, Vol. 10422)*, Francesco Ranzato (Ed.). Springer, 23–45. doi:10.1007/978-3-319-66706-5_2

M. Clint. 1973. Program proving: Coroutines. *Acta Informatica* 2, 1 (1973), 50–63. doi:10.1007/BF00571463

Patrick Cousot. 1978. *Méthodes itératives de construction et d'approximation de points fixes d'opérateurs monotones sur un treillis, analyse sémantique des programmes.* Accreditation to supervise research. Institut National Polytechnique de Grenoble - INPG ; Université Joseph-Fourier - Grenoble I. https://theses.hal.science/tel-00288657 Universités : Université scientifique et médicale de Grenoble et Institut national polytechnique de Grenoble.

Patrick Cousot, Radhia Cousot, Francesco Logozzo, and Michael Barnett. 2012. An abstract interpretation framework for refactoring with application to extract methods with contracts. In *Proceedings of the 27th Annual ACM SIGPLAN Conference on Object-Oriented Programming, Systems, Languages, and Applications, OOPSLA 2012, part of SPLASH 2012, Tucson, AZ, USA, October 21-25, 2012*, Gary T. Leavens and Matthew B. Dwyer (Eds.). ACM, 213–232. doi:10.1145/2384616.2384633

Leonardo De Moura and Nikolaj Bjørner. 2008. Z3: An efficient SMT solver. In *International conference on Tools and Algorithms for the Construction and Analysis of Systems*. Springer, 337–340.

Ruijie Fang, Zachary Kincaid, and Thomas Reps. 2025. Software Model Checking via Summary-Guided Search (Extended Version). *CoRR* abs/2508.15137 (2025). arXiv:2508.15137 doi:10.48550/ARXIV.2508.15137

Eden Frenkel, Tej Chajed, Oded Padon, and Sharon Shoham. 2024. Efficient Implementation of an Abstract Domain of Quantified First-Order Formulas. In *International Conference on Computer Aided Verification*. Springer, 86–108.

Eden Frenkel, Kenneth McMillan, Oded Padon, and Sharon Shoham. 2026a. *Simplifying Safety Proofs with Forward-Backward Reasoning and Prophecy (Artifact).* doi:10.5281/zenodo.19071811

Eden Frenkel, Kenneth L. McMillan, Oded Padon, and Sharon Shoham. 2026b. Simplifying Safety Proofs with Forward-Backward Reasoning and Prophecy. arXiv:2604.15266 [cs.LO] https://doi.org/10.48550/arXiv.2604.15266

Travis Hance, Marijn Heule, Ruben Martins, and Bryan Parno. 2021. Finding Invariants of Distributed Systems: It's a Small (Enough) World After All. In *18th USENIX Symposium on Networked Systems Design and Implementation, NSDI 2021, April 12-14, 2021*, James Mickens and Renata Teixeira (Eds.). USENIX Association, 115–131. https://www.usenix.org/conference/nsdi21/presentation/hance

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

Thomas A. Henzinger, Shaz Qadeer, and Sriram K. Rajamani. 1998. You Assume, We Guarantee: Methodology and Case Studies. In *Computer Aided Verification, 10th International Conference, CAV '98, Vancouver, BC, Canada, June 28 - July 2, 1998, Proceedings (Lecture Notes in Computer Science, Vol. 1427)*, Alan J. Hu and Moshe Y. Vardi (Eds.). Springer, 440–451. doi:10.1007/BFB0028765

Cliff B. Jones. 1981. *Developing methods for computer programs including a notion of interference.* Ph. D. Dissertation. University of Oxford, UK.

Cliff B. Jones. 1983. Specification and Design of (Parallel) Programs. In *Information Processing 83, Proceedings of the IFIP 9th World Computer Congress, Paris, France, September 19-23, 1983*, R. E. A. Mason (Ed.). North-Holland/IFIP, 321–332.

Aleksandr Karbyshev, Nikolaj Bjørner, Shachar Itzhaky, Noam Rinetzky, and Sharon Shoham. 2017. Property-directed inference of universal invariants or proving their absence. *Journal of the ACM (JACM)* 64, 1 (2017), 1–33.

Jason R. Koenig, Oded Padon, Neil Immerman, and Alex Aiken. 2020. First-order quantified separators. In *Proceedings of the 41st ACM SIGPLAN International Conference on Programming Language Design and Implementation, PLDI 2020, London, UK, June 15-20, 2020*, Alastair F. Donaldson and Emina Torlak (Eds.). ACM, 703–717. doi:10.1145/3385412.3386018

Jason R. Koenig, Oded Padon, Sharon Shoham, and Alex Aiken. 2022. Inferring Invariants with Quantifier Alternations: Taming the Search Space Explosion. In *Tools and Algorithms for the Construction and Analysis of Systems*, Dana Fisman and Grigore Rosu (Eds.). Springer International Publishing, Cham, 338–356.

Leslie Lamport. 1998. The Part-Time Parliament. *ACM Trans. Comput. Syst.* 16, 2 (1998), 133–169. doi:10.1145/279227.279229

Leslie Lamport. 2001. Paxos made simple. *ACM SIGACT News* 32, 4 (2001), 18–25.

Leslie Lamport and Stephan Merz. 2022. Prophecy made simple. *ACM Transactions on Programming Languages and Systems (TOPLAS)* 44, 2 (2022), 1–27.

Makai Mann, Ahmed Irfan, Alberto Griggio, Oded Padon, and Clark W. Barrett. 2021. Counterexample-Guided Prophecy for Model Checking Modulo the Theory of Arrays. In *Tools and Algorithms for the Construction and Analysis of Systems - 27th International Conference, TACAS 2021, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2021, Luxembourg City, Luxembourg, March 27 - April 1, 2021, Proceedings, Part I (Lecture Notes in Computer Science, Vol. 12651)*, Jan Friso Groote and Kim Guldstrand Larsen (Eds.). Springer, 113–132. doi:10.1007/978-3-030-72016-2_7

Kenneth L. McMillan. 1992. *Symbolic Model Checking: an approach to the state explosion problem.* Ph. D. Dissertation. Carnegie Mellon. CMU Tech Rpt. CMU-CS-92-131.

Kenneth L. McMillan. 2001. Parameterized Verification of the FLASH Cache Coherence Protocol by Compositional Model Checking. In *Correct Hardware Design and Verification Methods, 11th IFIP WG 10.5 Advanced Research Working Conference, CHARME 2001, Livingston, Scotland, UK, September 4-7, 2001, Proceedings (Lecture Notes in Computer Science, Vol. 2144)*, Tiziana Margaria and Thomas F. Melham (Eds.). Springer, 179–195. doi:10.1007/3-540-44798-9_17

Kenneth L. McMillan. 2006. Lazy Abstraction with Interpolants. In *Computer Aided Verification, 18th International Conference, CAV 2006, Seattle, WA, USA, August 17-20, 2006, Proceedings*. 123–136. doi:10.1007/11817963_14

Kenneth L. McMillan. 2018. Eager Abstraction for Symbolic Model Checking. In *Computer Aided Verification - 30th International Conference, CAV 2018, Held as Part of the Federated Logic Conference, FloC 2018, Oxford, UK, July 14-17, 2018, Proceedings, Part I (Lecture Notes in Computer Science, Vol. 10981)*, Hana Chockler and Georg Weissenbacher (Eds.). Springer, 191–208. doi:10.1007/978-3-319-96145-3_11

Diego Ongaro and John Ousterhout. 2014. In search of an understandable consensus algorithm. In *Proceedings of the 2014 USENIX Conference on USENIX Annual Technical Conference* (Philadelphia, PA) *(USENIX ATC'14)*. USENIX Association, USA, 305–320.

Susan S. Owicki and David Gries. 1976. An Axiomatic Proof Technique for Parallel Programs I. *Acta Informatica* 6 (1976), 319–340. doi:10.1007/BF00268134

Oded Padon, Giuliano Losa, Mooly Sagiv, and Sharon Shoham. 2017. Paxos made EPR: Decidable reasoning about distributed protocols. *Proc. ACM Program. Lang.* 1, OOPSLA (2017), 108:1–108:31. doi:10.1145/3140568

Amir Pnueli. 1985. In Transition From Global to Modular Temporal Reasoning about Programs. In *Logics and Models of Concurrent Systems*, Krzysztof R. Apt (Ed.). Springer Berlin Heidelberg, Berlin, Heidelberg, 123–144.

William Schultz, Edward Ashton, Heidi Howard, and Stavros Tripakis. 2024. Scalable, Interpretable Distributed Protocol Verification by Inductive Proof Slicing. *CoRR* abs/2404.18048 (2024). arXiv:2404.18048 doi:10.48550/ARXIV.2404.18048

William Schultz, Ian Dardik, and Stavros Tripakis. 2022. Plain and Simple Inductive Invariant Inference for Distributed Protocols in TLA+. In *2022 Formal Methods in Computer-Aided Design (FMCAD)*. 273–283. doi:10.34727/2022/isbn.978-3-85448-053-2_34

Marcelo Taube, Giuliano Losa, Kenneth L. McMillan, Oded Padon, Mooly Sagiv, Sharon Shoham, James R. Wilcox, and Doug Woos. 2018a. Modularity for decidability of deductive verification with applications to distributed systems. *SIGPLAN Not.* 53, 4 (June 2018), 662–677. doi:10.1145/3296979.3192414

Marcelo Taube, Giuliano Losa, Kenneth L. McMillan, Oded Padon, Mooly Sagiv, Sharon Shoham, James R. Wilcox, and Doug Woos. 2018b. Modularity for decidability of deductive verification with applications to distributed systems. In *Proceedings of the 39th ACM SIGPLAN Conference on Programming Language Design and Implementation, PLDI 2018, Philadelphia, PA,*

USA, June 18-22, 2018, Jeffrey S. Foster and Dan Grossman (Eds.). ACM, 662–677. doi:10.1145/3192366.3192414

Cole Vick and Kenneth L. McMillan. 2023. Synthesizing History and Prophecy Variables for Symbolic Model Checking. In *Verification, Model Checking, and Abstract Interpretation - 24th International Conference, VMCAI 2023, Boston, MA, USA, January 16-17, 2023, Proceedings (Lecture Notes in Computer Science, Vol. 13881)*, Cezara Dragoi, Michael Emmi, and Jingbo Wang (Eds.). Springer, 320–340. doi:10.1007/978-3-031-24950-1_15

Yakir Vizel, Orna Grumberg, and Sharon Shoham. 2013. Intertwined Forward-Backward Reachability Analysis Using Interpolants. In *Tools and Algorithms for the Construction and Analysis of Systems - 19th International Conference, TACAS 2013, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2013, Rome, Italy, March 16-24, 2013. Proceedings (Lecture Notes in Computer Science, Vol. 7795)*. Springer, 308–323. doi:10.1007/978-3-642-36742-7_22

James R Wilcox, Yotam MY Feldman, Oded Padon, and Sharon Shoham. 2024. mypyvy: A research platform for verification of transition systems in first-order logic. In *International Conference on Computer Aided Verification*. Springer, 71–85.

Jianan Yao, Runzhou Tao, Ronghui Gu, and Jason Nieh. 2022. DuoAI: Fast, Automated Inference of Inductive Invariants for Verifying Distributed Protocols. In *16th USENIX Symposium on Operating Systems Design and Implementation, OSDI 2022, Carlsbad, CA, USA, July 11-13, 2022*, Marcos K. Aguilera and Hakim Weatherspoon (Eds.). USENIX Association, 485–501. https://www.usenix.org/conference/osdi22/presentation/yao

Tony Nuda Zhang, Travis Hance, Manos Kapritsos, Tej Chajed, and Bryan Parno. 2024. Inductive Invariants That Spark Joy: Using Invariant Taxonomies to Streamline Distributed Protocol Proofs. In *18th USENIX Symposium on Operating Systems Design and Implementation, OSDI 2024, Santa Clara, CA, USA, July 10-12, 2024*, Ada Gavrilovska and Douglas B. Terry (Eds.). USENIX Association, 837–853. https://www.usenix.org/conference/osdi24/presentation/zhang-nuda

Tony Nuda Zhang, Keshav Singh, Tej Chajed, Manos Kapritsos, and Bryan Parno. 2025a. Basilisk: Using Provenance Invariants to Automate Proofs of Undecidable Protocols. In *19th USENIX Symposium on Operating Systems Design and Implementation, OSDI 2025, Boston, MA, USA, July 7-9, 2025*, Lidong Zhou and Yuanyuan Zhou (Eds.). USENIX Association, 1–17. https://www.usenix.org/conference/osdi25/presentation/zhang-tony

Tony Nuda Zhang, Keshav Singh, Tej Chajed, Manos Kapritsos, and Bryan Parno. 2025b. Basilisk: Using provenance invariants to automate proofs of undecidable protocols. In *19th USENIX Symposium on Operating Systems Design and Implementation (OSDI 25)*. 1–17.

Received 2025-11-14; accepted 2026-04-03