Induction Duality: Primal-Dual Search for Invariants

ODED PADON, VMware Research, USA and Stanford University, USA
JAMES R. WILCOX, Certora, USA
JASON R. KOENIG, Stanford University, USA
KENNETH L. MCMILLAN, University of Texas at Austin, USA
ALEX AIKEN, Stanford University, USA

Many invariant inference techniques reason simultaneously about states and predicates, and it is well-known that these two kinds of reasoning are in some sense dual to each other. We present a new formal duality between states and predicates, and use it to derive a new primal-dual invariant inference algorithm. The new *induction duality* is based on a notion of provability by incremental induction that is formally dual to reachability, and the duality is surprisingly symmetric. The symmetry allows us to derive the dual of the well-known Houdini algorithm, and by combining Houdini with its dual image we obtain *primal-dual Houdini*, the first truly primal-dual invariant inference algorithm. An early prototype of primal-dual Houdini for the domain of distributed protocol verification can handle difficult benchmarks from the literature.

CCS Concepts: • **Theory of computation** → *Logic and verification*; **Invariants**; • **Software and its engineering** → **Formal methods**.

Additional Key Words and Phrases: invariant inference, induction duality, Houdini, primal-dual Houdini, IC3, property directed reachability, counterexample-guided abstraction refinement

ACM Reference Format:
Oded Padon, James R. Wilcox, Jason R. Koenig, Kenneth L. McMillan, and Alex Aiken. 2022. Induction Duality: Primal-Dual Search for Invariants. *Proc. ACM Program. Lang.* 6, POPL, Article 50 (January 2022), 29 pages. https://doi.org/10.1145/3498712

1 INTRODUCTION

In modern invariant inference methods, it is common for the search for a proof to be guided by the search for a refutation (i.e., a counterexample) and vice versa. It has been observed (e.g., in [Godefroid et al. 2010; McMillan 2014]) that the problems of proof and refutation are dual, in the sense that solutions in one space constrain solutions in the other. However, the treatment of proofs and refutations has always been asymmetric. As an example, many techniques make monotone progress on one side but not the other. For example, a CEGAR technique [Clarke et al. 2000] may accumulate predicates from one iteration to the next, while discarding counterexamples. On the other hand, ICE learning [Garg et al. 2014] accumulates counterexamples but discards conjectured invariants. IC3/PDR [Bradley 2011] accumulates both states and predicates but in an asymmetric manner, using backward reachable states and predicates proven to be invariant in a bounded sense.

We present the first invariant inference framework where proofs and refutations are fully symmetric. The framework is based on *induction duality*, a formal duality between a notion of proof

Authors' addresses: Oded Padon, VMware Research, Palo Alto, CA, USA and Stanford University, Stanford, CA, USA, padon@cs.stanford.edu; James R. Wilcox, Certora, USA, james@certora.com; Jason R. Koenig, Stanford University, Stanford, CA, USA, jrkoenig@stanford.edu; Kenneth L. McMillan, University of Texas at Austin, Austin, TX, USA, kenmcm@cs.utexas.edu; Alex Aiken, Stanford University, Stanford, CA, USA, aiken@cs.stanford.edu.

2475-1421/2022/1-ART50
https://doi.org/10.1145/3498712

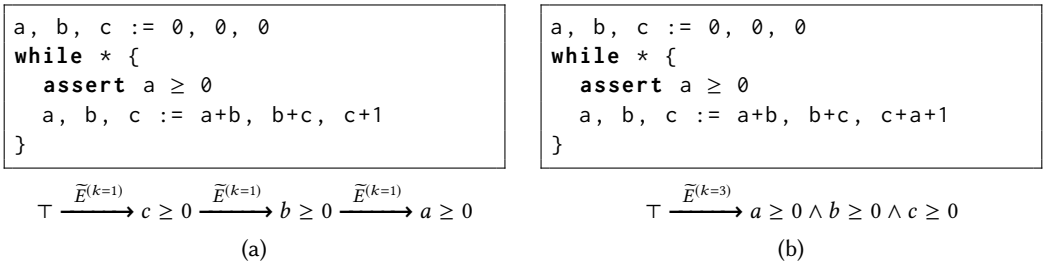

Fig. 1. Illustrating bounded-width incremental induction. The notation $P \xrightarrow{\widetilde{E}^{(k=n)}} Q$ means that there is an induction step of width n from P to Q (Section 3.2). Program (a) is provable with induction width 1. Program (b) requires induction width 3.

by incremental induction and a notion of refutation by forward reachability, which is surprisingly symmetric. Building on this symmetric duality, we derive *primal-dual Houdini*, a primal-dual invariant inference algorithm. Like a primal-dual linear programming solver, primal-dual Houdini alternates between the two sides of the duality making monotone progress on both sides: the proof side successively rules out more potential refutations, while the refutation side successively rules out more proofs. Moreover, the proof and refutation algorithms are symmetric at a high level, differing only in the underlying solver that produces steps in the proof or refutation.

Bounded-width incremental induction. A key ingredient of our approach is a notion of provability by incremental induction proofs that is formally dual to reachability in execution traces. Given some inductive predicate P, a step in an incremental proof strengthens P to another inductive predicate $P \wedge \Delta P$, and after enough strengthening steps the goal invariant is proven. This much is well-known [Bradley and Manna 2008; Manna and Pnueli 1995], and any such incremental proof can be contracted to a monolithic proof, i.e., a single induction step. We introduce *bounded-width incremental induction*, where in each step ΔP can only add bounded information to the invariant (e.g., bounded number of conjuncts), forcing proofs to proceed in an incremental fashion.

Figure 1 gives illustrative examples showing how bounded-width incremental induction works. In Figure 1a, starting from the invariant $\top$, we can prove that $c \geq 0$ is an invariant of the loop. Then, in a second induction step, using the fact that $c \geq 0$ is an invariant we can prove that $b \geq 0$ is an invariant. Finally, in a third round, we can additionally prove that our goal, $a \geq 0$, is an invariant. This example requires only induction width 1, as each step adds only a single new conjunct. In contrast, the program in Figure 1b cannot be proven with induction width 1. It requires induction width 3, as the three conjuncts must be added simultaneously by induction.

Induction duality. With bounded width, proofs become more like execution traces, i.e., an unbounded iteration of *bounded* steps. As a result, induction can be used to *reason about proofs* in a way that is analogous to the standard use of induction to reason about reachable states. For example, to prove that the assertion in Figure 1b cannot be proven with induction width 1, we must prove a fact about unbounded sequences. As we show, such a fact can be proven by induction, where the induction hypothesis is a set of states. This set of states is analogous to a set of predicates that forms an inductive invariant, and we call it *dual-inductive*. The connection between induction over program execution and induction over proof sequences is formalized by our notion of *induction duality*. We note that induction duality is different from the standard duality between states and predicates given by a Galois connection [Cousot and Cousot 1979; Lawvere 1969]. Indeed, our

algorithmic developments in this paper utilize both the standard Galois-connection duality and the new induction duality.

Primal-dual Houdini. We leverage our induction duality to obtain a dual version of the well-known Houdini algorithm [Flanagan et al. 2001]. While Houdini finds the maximal inductive subset of a given set of predicates, the resulting *dual Houdini* procedure finds the maximal *dual-inductive* subset of a given set of states. Moreover, while Houdini finds counterexamples to induction (CTIs) as part of its computation which consist of states and transitions, dual Houdini finds *dual-CTIs* which consist of predicates and induction steps. We observe that dual-CTIs offer a new mechanism for discovering predicates, and can be computed by adapting existing techniques. By combining (primal) Houdini as a mechanism for learning states with dual Houdini as a mechanism for learning predicates, we obtain a primal-dual invariant inference algorithm, *primal-dual Houdini*. This new algorithm admits progress theorems that critically depend on the fact that the input to primal Houdini comes from the output of dual Houdini, and vice versa, which illustrates the primal-dual interaction between the two. Our prototype implementation of primal-dual Houdini in the domain of distributed protocol verification is already competitive with state-of-the-art techniques.

Contributions. This paper makes the following contributions: (i) formalizing *induction duality*, a symmetric connection between reachability and *bounded-width incremental induction*; (ii) deriving *dual Houdini* as the image under induction duality of the Houdini algorithm, including *dual-CTIs* which offer a new mechanism for predicate discovery; (iii) combining Houdini and dual Houdini to obtain *primal-dual Houdini*, a primal-dual invariant inference algorithm which admits interesting progress and termination theorems that highlight the primal-dual combination; and (iv) reporting experiments showing that an early prototype of primal-dual Houdini in the domain of distributed protocol verification is already competitive with state-of-the-art techniques.

Outline. Section 2 presents necessary preliminaries. Section 3 develops induction duality, first as a notion on graphs, and then in relation to bounded-width incremental induction. Section 4 develops primal-dual Houdini and presents several theorems utilizing its primal-dual nature. Section 5 reports on an early prototype of primal-dual Houdini for distributed protocol verification and compares it to five state-of-the-art tools. Finally, Section 6 discusses related work and Section 7 concludes.

2 PRELIMINARIES

In this section, we formalize well-known concepts of transition systems and inductive invariants. We begin by formalizing states and predicates with a well-known Galois connection between them. This Galois connection forms one standard duality between states/predicates and reachability/invariance, and later in the paper two dualities will be at play: the standard one and the new induction duality.

Notation. For a set $\mathcal{X}$, we use $\mathcal{P}(\mathcal{X})$ for the powerset of $\mathcal{X}$, $\mathcal{P}_\omega(\mathcal{X})$ for the set of *finite* subsets of $\mathcal{X}$, and $\mathcal{X}^*$ for the set of finite sequences of elements from $\mathcal{X}$. For $X \in \mathcal{P}_\omega(\mathcal{X})$, we use $|X|$ for the cardinality of X. For $X \in \mathcal{X}^*$, we use $|X|$ for the length of X and $X_i \in \mathcal{X}$ for the i-th element in the sequence, where $0 \leq i < |X|$. Whenever we have a binary relation $\square \subseteq \mathcal{X} \times \mathcal{Y}$ we write $x \square y$ for $(x, y) \in \square$ and $x \not\square y$ for $(x, y) \notin \square$, and lift $\square$ to combinations of sets and elements in the usual way (where $x \in \mathcal{X}$, $y \in \mathcal{Y}$, $X \subseteq \mathcal{X}$, and $Y \subseteq \mathcal{Y}$): $x \square Y$ is $\forall y \in Y.\ x \square y$, $X \square y$ is $\forall x \in X.\ x \square y$, and $X \square Y$ is $\forall x \in X, y \in Y.\ x \square y$; and $x \not\square Y$, $X \not\square y$, and $X \not\square Y$ are the negations thereof (e.g., $X \not\square Y$ means $\exists x \in X, y \in Y.\ x \not\square y$). We similarly lift binary relations to finite sequences: for $X \in \mathcal{X}^*$ and $Y \in \mathcal{Y}^*$, $X \square Y$ is $\bigwedge_{0 \leq i < |X|, 0 \leq j < |Y|} X_i \square Y_j$, $x \square Y$ is $\bigwedge_{0 \leq j < |Y|} x \square Y_j$, and $X \square y$ is $\bigwedge_{0 \leq i < |X|} X_i \square y$.

States and predicates. Let us fix a set $\mathbb{S}$ of *states*, a set $\mathbb{P}$ of *predicates*, and $\models \subseteq \mathbb{S} \times \mathbb{P}$ a *satisfaction relation* between states and predicates. The set of states $\mathbb{S}$ is typically countably infinite and

represents all possible system configurations (e.g., mappings from program variables to integers, finite first-order structures). The set of predicates $\mathbb{P}$ is typically countably infinite and represents possible assertions used in proofs by induction about reachable system configurations (e.g., linear inequalities, universally quantified formulas). In the rest of this paper, finite subsets of $\mathbb{P}$ represent conjunction; $\mathbb{P}$ itself is not closed under conjunction, which is especially crucial for *bounded-width incremental induction* (defined in Section 3.2). In the sequel we use s, t to range over states; p, q to range over predicates; S, R to range over sets of states; and P, Q to range over sets of predicates.

Galois connection. The satisfaction relation gives rise to a well-known Galois connection between the powerset lattices of states and predicates [Cousot and Cousot 1977, 1979; Lawvere 1969; Smith 2010]. If $\langle L, \leq \rangle$ and $\langle M, \sqsubseteq \rangle$ are posets and $\alpha : L \to M$ and $\gamma : M \to L$ satisfy $\forall x \in L,\ y \in M.\ \alpha(x) \sqsubseteq y \leftrightarrow x \leq \gamma(y)$, then the pair (α, γ) is a *Galois connection*, denoted by $\langle L, \leq \rangle \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} \langle M, \sqsubseteq \rangle$. The following functions form such a Galois connection $\langle \mathcal{P}(\mathbb{S}), \subseteq \rangle \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} \langle \mathcal{P}(\mathbb{P}), \supseteq \rangle$ between the powerset lattice of states (ordered by $\subseteq$) and the powerset lattice of predicates (ordered by $\supseteq$):

$$\alpha : \mathcal{P}(\mathbb{S}) \to \mathcal{P}(\mathbb{P}) = \lambda S.\ \{p \in \mathbb{P} \mid S \models p\}, \tag{1}$$

$$\gamma : \mathcal{P}(\mathbb{P}) \to \mathcal{P}(\mathbb{S}) = \lambda P.\ \{s \in \mathbb{S} \mid s \models P\}. \tag{2}$$

Transition systems. A *transition system* is given by a set of *initial states* $\iota \subset \mathbb{S}$ and a *transition relation* $\tau \subset \mathbb{S} \times \mathbb{S}$. An *execution trace* of a transition system is a sequence of states $s_0, \ldots, s_n$ such that $s_0 \in \iota$ and $(s_i, s_{i+1}) \in \tau$ for $0 \leq i < n$. A state is *reachable* if it is part of any execution trace. Otherwise, the state is *unreachable*. Equivalently, the set of reachable states can be defined as $\mathrm{lfp}\, \lambda S.\ \iota \cup \{s' \mid \exists s \in S.\ (s, s') \in \tau\}$, where we use $\mathrm{lfp}\, f$ to denote the least fixed point of f. We say that a finite set of states $R \in \mathcal{P}_\omega(\mathbb{S})$ is *evidently reachable* if all of its states are reachable via execution traces that only contain states from R. Equivalently, R is evidently reachable iff:

$$R = \mathrm{lfp}\, \lambda S.\ (\iota \cap R) \cup \{s' \mid \exists s \in S.\ (s, s') \in \tau \cap (R \times R)\}\,. \tag{3}$$

Given a finite R, checking if R is evidently reachable therefore amounts to a straightforward finite least fixed point computation. Note that to be evidently reachable, a set has to include all the states needed to justify its reachability. Thus, a set of reachable states that is not evidently reachable may be made evidently reachable by including additional states that witness missing transitions.

Invariants and Inductiveness. A predicate is *invariant* if it is satisfied by all reachable states. Otherwise, the predicate is *noninvariant*. A set of predicates is invariant if all its predicates are invariant. A finite set of predicates $Q \in \mathcal{P}_\omega(\mathbb{P})$ is an *inductive invariant* or *inductive* if:

$$\forall s \in \iota.\ s \models Q, \text{ and} \tag{4}$$

$$\forall (s, s') \in \tau.\ s \models Q \to s' \models Q. \tag{5}$$

That is, a set of predicates is an inductive invariant if it is satisfied by the initial states and preserved by transitions of τ. Any inductive invariant is invariant (by induction on execution traces).

3 INDUCTION DUALITY

We now develop *induction duality*, a formal symmetric duality between reachability and provability by bounded-width incremental induction. As we shall see in Section 4, under the formalism developed here, dual Houdini is obtained as the image of classical (primal) Houdini. We start with a duality on graphs (Section 3.1). We then apply the graph duality to obtain a formal definition of bounded-width incremental induction that is dual to reachability (Section 3.2).

The graph notion of induction duality is a connection between two graphs. As we shall see, in one graph edges will correspond to transitions and paths to execution traces, and in the other graph

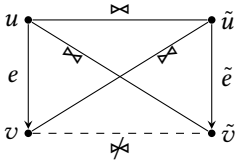

Fig. 2. Illustration of $\|$ and eq. (6). For $e = (u, v)$ and $\tilde{e} = (\tilde{u}, \tilde{v})$, out of the 16 possible ways $\bowtie$ can relate u, v, $\tilde{u}$, and $\tilde{v}$, only this one has $e \nparallel \tilde{e}$.

edges will correspond to incremental induction steps and paths to incremental induction proofs. The connection is symmetric, laying the foundation for primal-dual Houdini.

3.1 Induction-Dual Graphs

A (directed, possibly infinite) *graph* G is a pair $G = (V, E)$, where V is a set of vertices and $E \subseteq V \times V$ is a set of edges. We use E^* for the reflexive-transitive closure of E. A *path* in G is a finite sequence of vertices $\pi = \pi_0, \ldots, \pi_n$ such that $(\pi_i, \pi_{i+1}) \in E$ for every $0 \le i < n$; we write $|\pi|$ for n the length of π, and $\pi \in E^*$ to mean that π is a path of E-edges.

We consider two typically infinite graphs, $G = (V, E)$ and $\widetilde{G} = (\widetilde{V}, \widetilde{E})$, and a binary relation $\bowtie \subseteq V \times \widetilde{V}$ on their vertices. As we shall see in Section 3.2, G represents states and transitions, $\widetilde{G}$ represents predicates and induction steps, and $\bowtie$ represents the satisfaction relation. We use u, v, w to range over vertices, e to range over edges, and π to range over paths in G. We use $\tilde{u}$, $\tilde{v}$, $\tilde{w}$, $\tilde{e}$, and $\tilde{\pi}$ to range over corresponding objects of $\widetilde{G}$.

Definition 3.1 (Induction-Dual Graphs). For pairs $e = (u, v) \in V \times V$ and $\tilde{e} = (\tilde{u}, \tilde{v}) \in \widetilde{V} \times \widetilde{V}$, let $e \parallel \tilde{e}$ denote:

$$u \bowtie \tilde{u} \wedge u \bowtie \tilde{v} \wedge v \bowtie \tilde{u} \rightarrow v \bowtie \tilde{v}. \tag{6}$$

$\widetilde{G}$ is an *induction dual* to G if $E \parallel \widetilde{E}$, i.e., $\forall e \in E,\ \tilde{e} \in \widetilde{E}.\ e \parallel \tilde{e}$.

The $\|$ relation is illustrated in Figure 2. As can be seen from the horizontal symmetry of the figure, eq. (6) and Definition 3.1 are symmetric between G and $\widetilde{G}$. That is, $\widetilde{G}$ is induction dual to G by relation $\bowtie$ iff G is induction dual to $\widetilde{G}$ by relation $\bowtie^T$ ($\bowtie$ transposed). We thus simply say that G and $\widetilde{G}$ are induction duals. Note that induction duality is antitone (anti-monotone) w.r.t. E and $\widetilde{E}$. That is, removing edges from induction-dual graphs results in induction-dual graphs (and a pair of graphs with no edges are always induction dual), but adding edges may break induction duality.

Induction duality imposes a *pairwise* constraint on *edges* of G and $\widetilde{G}$, and the next theorem shows it also implies a constraint on *paths* of G and $\widetilde{G}$.

Theorem 3.2 (Paths in Induction-Dual Graphs). *Let graphs $G = (V, E)$ and $\widetilde{G} = (\widetilde{V}, \widetilde{E})$ be induction dual by relation $\bowtie$, then $\forall \pi \in E^*,\ \tilde{\pi} \in \widetilde{E}^*.\ \pi_0 \bowtie \tilde{\pi} \wedge \pi \bowtie \tilde{\pi}_0 \rightarrow \pi \bowtie \tilde{\pi}$.*

Proof. Suppose to the contrary that $\pi_0 \bowtie \tilde{\pi}$ and $\pi \bowtie \tilde{\pi}_0$ but $\pi \not\bowtie \tilde{\pi}$. Let $i = \min\{i' \mid 0 \le i' \le |\pi| \wedge \pi_{i'} \not\bowtie \tilde{\pi}\}$ and $j = \min\{j' \mid 0 \le j' \le |\tilde{\pi}| \wedge \pi_i \not\bowtie \tilde{\pi}_{j'}\}$. Since $\pi_0 \bowtie \tilde{\pi}$, $\pi \bowtie \tilde{\pi}_0$, we have $0 < i, j$. Consider the edges $(\pi_{i-1}, \pi_i) \in E$ and $(\tilde{\pi}_{j-1}, \tilde{\pi}_j) \in \widetilde{E}$, and observe that we must have $(\pi_{i-1}, \pi_i) \nparallel (\tilde{\pi}_{j-1}, \tilde{\pi}_j)$, contradicting $E \parallel \widetilde{E}$. □

The definition of induction duality only constrains single edges in the graphs, but as Theorem 3.2 shows, this translates by induction to a similar relation on paths of the two graphs. The proof of Theorem 3.2 critically uses induction on both paths simultaneously, that is, taking the minimum indices in both π and $\tilde{\pi}$. Another perspective is provided by the following corollary.

Corollary 3.3. *Let $\mu(\pi, \tilde{u})$ denote the minimum index of a vertex in π that is* not *related to $\tilde{u}$ by $\bowtie$, or ∞ if $\pi \bowtie \tilde{u}$. That is: $\mu(\pi, \tilde{u}) = \min\{i \mid (0 \le i \le |\pi| \wedge \pi_i \not\bowtie \tilde{u}) \vee i = \infty\}$. Similarly, let $\tilde{\mu}(u, \tilde{\pi}) = \min\{i \mid (0 \le i \le |\tilde{\pi}| \wedge u \not\bowtie \tilde{\pi}_i) \vee i = \infty\}$. If $\pi \in E^*$, $\tilde{\pi} \in \widetilde{E}^*$, then $\mu(\pi, \tilde{\pi}_0) \le \mu(\pi, \tilde{\pi}_j)$ for $0 \le j \le |\tilde{\pi}|$; dually $\tilde{\mu}(\pi_0, \tilde{\pi}) \le \tilde{\mu}(\pi_i, \tilde{\pi})$ for $0 \le i \le |\pi|$.*

Corollary 3.3 means that if there is a path from $\tilde{u}$ to $\tilde{v}$ in $\widetilde{G}$, then any path π in G must break $\bowtie$ connection with $\tilde{u}$ *before, or at the same time,* as it breaks the connection with $\tilde{v}$ (and dually when v is reachable from u in G, for paths in $\widetilde{G}$).

Intuitively, Theorem 3.2 and Corollary 3.3 suggest that for induction-dual graphs we can view paths in one graph as incremental induction proofs on paths of the other graph (and vice versa). To see this, suppose there are special vertices $u_0 \in V$ and $\tilde{u}_0 \in \widetilde{V}$ such that $u_0 \bowtie \widetilde{V}$ and $V \bowtie \tilde{u}_0$. Then, we can view a path from $\tilde{u}_0$ to any $\tilde{v}$ as an incremental induction proof that for any v reachable from u_0, $v \bowtie \tilde{v}$. Similarly, we can view a path from u_0 to any v as a proof that for $\tilde{v}$ reachable from $\tilde{u}_0$, $v \bowtie \tilde{v}$.

3.2 Bounded-Width Incremental Induction Proofs

We now relate the graph duality to transition systems and incremental induction proofs. We use the $\|$ relation and eq. (6) to obtain a notion of *bounded-width incremental induction* that is symmetric to reachability. This symmetry is a property of the *induction duality structure*, defined below, which formalizes bounded-width incremental induction proofs for a transition system. The structure includes four sets of edges related by $\|$, which are explained in detail following the definition.

Definition 3.4 (Induction Duality Structure). Given a set of states $\mathbb{S}$, a set of predicates $\mathbb{P}$, a set of initial states $\iota \subseteq \mathbb{S}$, a transition relation $\tau \subseteq \mathbb{S} \times \mathbb{S}$, and an *induction width* $k \in \mathbb{N}$, the *induction duality structure* is $\left\langle V, \widetilde{V}, \bowtie, E, E_\omega, \widetilde{E}_\omega, \widetilde{E} \right\rangle$, where $V = \mathcal{P}_\omega(\mathbb{S})$, $\widetilde{V} = \mathcal{P}_\omega(\mathbb{P})$, $\bowtie \subseteq V \times \widetilde{V} = \{(S, P) \mid S \models P\}$, and:

$$E \subseteq V \times V = \{(S, S \cup \{s'\}) \mid s' \in \iota \vee \exists s \in S.\ (s, s') \in \tau\}, \tag{7}$$

$$\widetilde{E}_\omega \subseteq \widetilde{V} \times \widetilde{V} = \{(P, P \cup Q) \mid (P, P \cup Q) \parallel E\}, \tag{8}$$

$$\widetilde{E} \subseteq \widetilde{V} \times \widetilde{V} = \{(P, P \cup Q) \mid (P, P \cup Q) \parallel E \wedge |Q| \le k\}, \tag{9}$$

$$E_\omega \subseteq V \times V = \left\{(S, S \cup R) \mid (S, S \cup R) \parallel \widetilde{E}\right\}. \tag{10}$$

Vertices in V are finite sets of states, vertices in $\widetilde{V}$ are finite sets of predicates, and $\bowtie$ is the satisfaction relation (where a set of predicates is interpreted conjunctively). The four sets of edges, E, $\widetilde{E}_\omega$, $\widetilde{E}$ and E_ω, are all increasing, that is they are of the form $(X, X \cup Y)$. E is determined by ι and τ, $\widetilde{E}_\omega$ is the set of increasing edges parallel ($\|$) to E, $\widetilde{E}$ the subset of those where at most k predicates are added, and E_ω the set of all increasing edges parallel to $\widetilde{E}$.[1] We now explain the meaning of these four sets of edges, and use them to define bounded-width incremental induction proofs.

E: Transitions and reachability. E is directly determined by ι and τ, such that (V, E) captures the transition system. Every trace of the transition system $s_0, \ldots, s_n$ corresponds to a path in (V, E): $\emptyset, \{s_0\}, \{s_0, s_1\}, \ldots, \{s_0, \ldots, s_n\}$. The following proposition relates E and evident reachability.

Proposition 3.5. *For any set of states $R \in \mathcal{P}_\omega(\mathbb{S})$: $(\emptyset, R) \in E^*$ iff R is evidently reachable.*

Proof. $\Rightarrow$: by induction on the path in E^*. $\Leftarrow$: by induction on the least fixed point of eq. (3). □

[1] In this paper, we restrict $\widetilde{E}$ by number of predicates added. This restriction is both simple and suitable for the examples we consider in our evaluation. However, the theory we develop does not strictly depend on this specific restriction, and is valid as long as $\widetilde{E}$ is *some* subset of $\widetilde{E}_\omega$ that can be effectively searched. We consider such generalizations, leading to other notions of bounded incremental induction, an opportunity for future research.

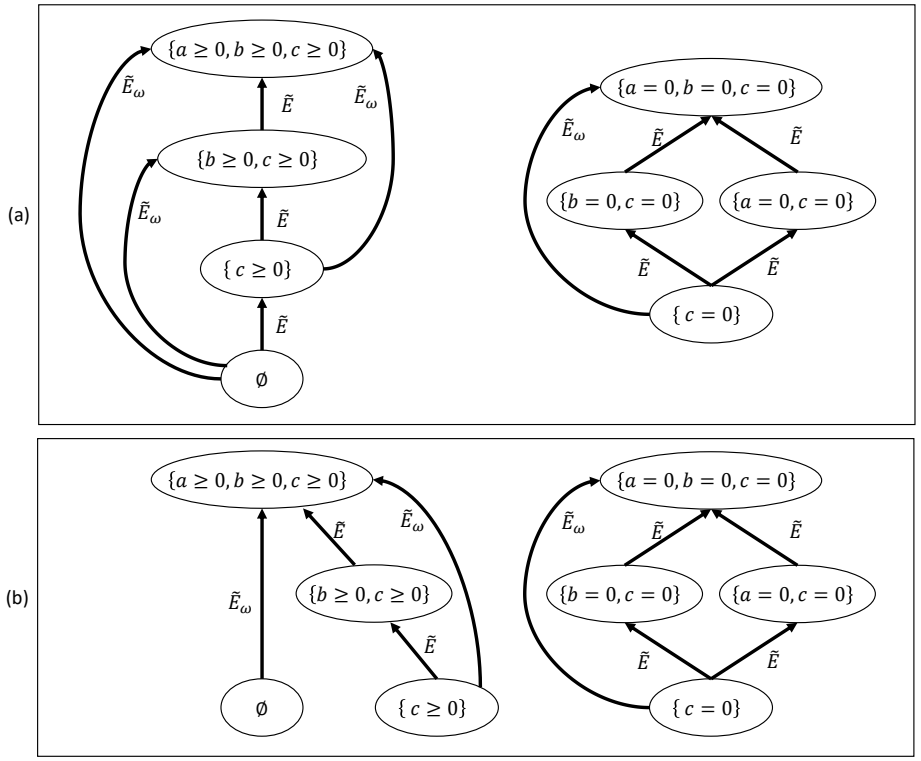

Fig. 3. Illustration of $\widetilde{E}$ and $\widetilde{E}_\omega$ (Definition 3.4) for the programs of Figure 1 and induction width $k = 1$. (a) corresponds to the program listed in Figure 1a, and (b) to the program listed in Figure 1b. The set of predicates considered is $\mathbb{P} = \{a \geq 0, b \geq 0, c \geq 0, a = 0, b = 0, c = 0\}$. Only a selected subgraph is presented (i.e., not all subsets of $\mathbb{P}$ are included). Self loops are omitted, and every edge labeled $\widetilde{E}$ also implies an $\widetilde{E}_\omega$ edge (as $\widetilde{E} \subseteq \widetilde{E}_\omega$). Note that since $\widetilde{E}$ and $\widetilde{E}_\omega$ edges are always from P to $P \cup Q$, there is no edge from, e.g., $\{c = 0\}$ to $\{c \geq 0\}$. (There is an $\widetilde{E}$ edge from $\{c = 0\}$ to $\{c = 0, c \geq 0\}$ which is not depicted since the latter set is not included in the presented subgraph.)

$\widetilde{E}_\omega$: Monolithic induction and incremental induction. $\widetilde{E}_\omega$ is defined to be the set of increasing edges parallel ($\|$) to E. While $\widetilde{E}_\omega$ is defined in terms of $\|$ and E, the following proposition characterizes it directly in terms of ι and τ.

Proposition 3.6. *For any sets of predicates $P, Q \in \mathcal{P}_\omega(\mathbb{P})$, $(P, P \cup Q) \in \widetilde{E}_\omega$ iff:*

$$\forall s \in \iota.\ s \models P \rightarrow s \models Q, \text{ and} \tag{11}$$

$$\forall (s, s') \in \tau.\ s \models P \wedge s \models Q \wedge s' \models P \rightarrow s' \models Q. \tag{12}$$

Proof. By combining eqs. (6) to (8). Note that eq. (12) is very similar in form to eq. (6). □

By observing that for $P = \emptyset$, eqs. (11) and (12) reduce to eqs. (4) and (5), we get the following corollary relating $\widetilde{E}_\omega$ and inductive invariants.

Corollary 3.7. *For any set of predicates $Q \in \mathcal{P}_\omega(\mathbb{P})$, $(\emptyset, Q) \in \widetilde{E}_\omega$ iff Q is inductive.*

For any edge $(P, Q) \in \widetilde{E}_\omega$, by Theorem 3.2 if P is invariant then Q is also invariant. Therefore, if there is a path in $\widetilde{E}_\omega$ from $\emptyset$ to Q, i.e. $(\emptyset, Q) \in \widetilde{E}_\omega^*$, then Q is invariant. We therefore call $(P, Q) \in \widetilde{E}_\omega$

an *induction transition* from P to Q, and define an *incremental induction proof* to be a path in $\widetilde{E}_\omega$ that starts at $\emptyset$. We expect that every incremental induction proof can be reduced to a single (monolithic) inductive invariant. With our definitions, this expectation translates to: $(\emptyset, Q) \in \widetilde{E}^*_\omega \Rightarrow (\emptyset, Q) \in \widetilde{E}_\omega$. Indeed, as Theorem 3.8 later shows, we actually have $\widetilde{E}^*_\omega = \widetilde{E}_\omega$.

Figure 3 illustrates $\widetilde{E}$ and $\widetilde{E}_\omega$ for the two programs listed in Figure 1 with induction width $k = 1$. For the illustration, we set $\mathbb{P} = \{a \geq 0, b \geq 0, c \geq 0, a = 0, b = 0, c = 0\}$, which leads to $|\widetilde{V}| = 2^6 = 64$. The figure only shows the subgraphs obtained for a selected set of 8 vertices in $\widetilde{V}$. (Note that $\mathbb{P}$ and $\widetilde{V}$ are typically infinite, and we use this finite example for illustration purposes.) In both Figure 3a and Figure 3b, there is an $\widetilde{E}_\omega$ edge from $\emptyset$ to $\{a \geq 0, b \geq 0, c \geq 0\}$, representing the inductive invariant for both programs. For both programs, $\widetilde{E}_\omega$ also includes an edge from $\{c = 0\}$ to $\{a = 0, b = 0, c = 0\}$. Clearly, this edge does not represent invariants of the programs. However, according to Corollary 3.3, any execution trace of either program that violates $a = 0$ or $b = 0$ must also violate $c = 0$, which represents valid knowledge about the programs that may be useful in analyzing them. Thus, as intuitive way to interpret an $\widetilde{E}_\omega$ edge from P to Q is: any execution trace that satisfies P also satisfies Q.

Our notion of induction transition differs from the notion of relative inductiveness used in IC3/PDR [Bradley 2011, 2012], which is also inspired by incremental induction. Focusing on τ, IC3's notion of Q being inductive relative to P is $\forall (s, s') \in \tau.\ s \models P \wedge s \models Q \rightarrow s' \models Q$ whereas our induction transition from P to Q is given by eq. (12), which is very similar but also includes $s' \models P$ in the antecedent. In terms of Figure 2 and eq. (6), IC3's relative inductiveness omits the edge $v \bowtie \tilde{u}$ (P corresponds to $\tilde{u}$ and Q to $\tilde{v}$), which would not result in a symmetric duality if we took it as our definition of induction transitions. Note however, that if P is inductive, then relative induction coincides with eq. (12). In an incremental induction proof the source of every induction transition is itself inductive, but as we shall see, primal-dual Houdini uses induction transitions where P is not inductive as part of the search process.

Finally, we note that if ι, τ, P and Q are represented symbolically using formulas in the standard way (where τ is represented using primed symbols for the post-state), then checking if there is an induction transition from P to $P \cup Q$ can be reduced to checking the validity of the following implications matching eqs. (11) and (12), which in many cases can be automated using an SMT solver: $\iota \wedge P \rightarrow Q$ and $P \wedge Q \wedge \tau \wedge P' \rightarrow Q'$.

$\widetilde{E}$: Bounded-width incremental induction. While $\widetilde{E}_\omega$ edges represent incremental induction proofs, they pose no restriction on the amount of information each induction transition can add at once, as manifested by the fact that any path in $\widetilde{E}_\omega$ can be compacted to a single induction transition. This property also makes the task of searching for an induction transition very difficult, which motivates our definition of $\widetilde{E}$, a restriction of $\widetilde{E}_\omega$ to induction transitions where at most k predicates are added. We call $(P, Q) \in \widetilde{E}$ a *k-width induction transition* from P to Q, and define a *k-width incremental induction proof* to be a path in $\widetilde{E}$ that starts at $\emptyset$. We say that a predicate p is *k-provable* if there is some Q s.t. $(\emptyset, Q) \in \widetilde{E}^*$ with $p \in Q$. That is, a predicate is k-provable if it can be proven to be invariant using a k-width incremental induction proof. If a predicate is not k-provable, we say it is *k-unprovable*. We say that a set of predicates Q is *evidently k-provable* if a k-width proof of every predicate in Q can be constructed using only predicates from Q.

For example, as Figure 3 shows, for the program of Figure 1a there is an $\widetilde{E}$ path from $\emptyset$ to $\{a{\geq}0, b{\geq}0, c{\geq}0\}$, which is not the case for the program of Figure 1b. Therefore, for Figure 1a, $a{\geq}0$ is 1-provable and $\{a{\geq}0, b{\geq}0, c{\geq}0\}$ is evidently 1-provable. In contrast, for Figure 1b, $a{\geq}0$ is 3-provable but not 1- or 2-provable. While in both examples a monolithic proof uses the same three predicates, the narrowest incremental proofs for each example are of different widths, due to

the different dependency structure between the predicates. In general, given a set of predicates Q that is inductive, the minimal k for which Q is evidently k-provable matches the largest strongly connected component in a suitable dependency graph.

We say a state is *k-abstractly-reachable* if it satisfies all k-provable predicates. That is, a state is k-abstractly-reachable if it cannot be proven to be unreachable using k-width proofs. All reachable states are k-abstractly-reachable, but an unreachable state may also be k-abstractly-reachable (i.e., when proving its unreachability requires induction width $> k$ or predicates beyond $\mathbb{P}$). For example, by the discussion above, any bad state (with $a < 0$) is *not* 1-abstractly-reachable for the example of Figure 1a, but is 1- and 2-abstractly-reachable (but not 3-abstractly-reachable) for Figure 1b.

E_ω: Monolithic induction proofs of k-unprovability. In order to prove that some predicate is k-unprovable, or that some state is k-abstractly-reachable, we must use induction proofs to put an upper bound on $\widetilde{E}^*$. The induction duality structure provides such proofs via E_ω, defined to be all edges in V that are parallel ($\|$) to $\widetilde{E}$. That is, E_ω edges connect sets of states, and E_ω is to $\widetilde{E}$ precisely what $\widetilde{E}_\omega$ is to E. In proofs of k-unprovability, the induction hypothesis is thus a set of states. Note that by Theorem 3.2, if $(\emptyset, R) \in E_\omega$ then all the states in R satisfy every k-provable predicate, that is, they are k-abstractly-reachable. We therefore say R is *evidently k-abstractly-reachable*, or *dual-inductive*, if $(\emptyset, R) \in E_\omega$. We expect that every evidently reachable set of states is also evidently k-abstractly-reachable. With our definition and Proposition 3.5 this expectation translates to $(\emptyset, R) \in E^* \implies (\emptyset, R) \in E_\omega$. Indeed, as Theorem 3.8 later shows, we have $E^* \subseteq E_\omega$.

Checking if a set of states R is evidently k-abstractly-reachable amounts to finding a k-width induction transition $(P, Q) \in \widetilde{E}$ such that $(\emptyset, R) \not\parallel (P, Q)$, which we call a *dual-CTI* (dual counterexample to induction), or concluding that no such dual-CTI exists. Indeed, one of the motivations for restricting induction width is to make such a dual-inductiveness check practical, as the restriction to k-width restricts the search space for dual-CTIs.[2]

For the program of Figure 1b and the set of predicates used in Figure 3, the following set is evidently 1-abstractly-reachable and also evidently 2-abstractly-reachable:

$$R = \{(0,0,-1)\,,\ (0,-1,0)\,,\ (-1,-1,1)\,,\ (-2,0,1)\,,\ (-2,1,0)\,,\ (-1,1,-1)\}\,,$$

where (x, y, z) denotes the program state where $a = x$, $b = y$, and $c = z$. To see that the above R is evidently 1-abstractly-reachable, one needs to see that every $(P, Q) \in \widetilde{E}$ is parallel to $(\emptyset, R)$, which amounts to: if $R \models P$ then $R \models Q$. It is easy to see that this holds for the $\mathbb{P}$ used in Figure 3, since the only $P \subseteq \mathbb{P}$ that is satisfied by R is $\emptyset$, which has no outgoing $\widetilde{E}$ edges (for either $k = 1$ or $k = 2$). However, the reader can also convince themselves that the above R is evidently 1-abstractly-reachable even when $\mathbb{P}$ includes all predicates of the form $v \square x$ where v is a program variable, $x \in \mathbb{Z}$, and $\square \in \{=, \leq, \geq, <, >\}$.

Recap. We have seen that the four sets of edges in the induction duality structure represent:

E: transitions of the transition system, defining the set of reachable states via E^*;
$\widetilde{E}$: steps of bounded-width incremental induction, defining the set of k-provable predicates via $\widetilde{E}^*$;
$\widetilde{E}_\omega$: unbounded monolithic induction providing an upper bound on E^*, i.e., proving that some states are unreachable or some predicates are invariant; and
E_ω: unbounded monolithic induction providing an upper bound on $\widetilde{E}^*$, i.e., proving that some predicates are k-unprovable or some states are k-abstractly-reachable.

[2]Checking dual-inductiveness (i.e., searching for a dual-CTI) is a central task resulting from the theory developed in this paper. However, since the solution is domain-specific, we delay discussing its details to Section 5.

Properties of the induction duality structure. We now prove several properties of the induction duality structure. These properties have been used above to show that our definitions match the intuitive expectations, and they are also used in the development of primal-dual Houdini. Intuitively, items 1 and 2 in the following theorem explore the transitive closure of the four types of edges, and items 3 and 4 restate the maximality of $\widetilde{E}_\omega$ and E_ω in a way that is useful for the sequel.

Theorem 3.8. *If* $\left\langle V, \widetilde{V}, \bowtie, E, E_\omega, \widetilde{E}, \widetilde{E}_\omega \right\rangle$ *is an induction duality structure, then:*

(1) $E^* \subseteq E_\omega = E_\omega^*$ *and* $\widetilde{E}^* \subseteq \widetilde{E}_\omega = \widetilde{E}_\omega^*$ *;*
(2) $E^* \parallel \widetilde{E}_\omega$ *and* $E_\omega \parallel \widetilde{E}^*$ *;*
(3) for $P, Q \in \widetilde{V}$*, either* $(P, P \cup Q) \in \widetilde{E}_\omega$ *or* $\exists e \in E.\ e \not\parallel (P, P \cup Q)$ *; and*
(4) for $S, R \in V$*, either* $(S, S \cup R) \in E_\omega$ *or* $\exists \tilde{e} \in \widetilde{E}.\ (S, S \cup R) \not\parallel \tilde{e}$*.*

Proof. We first observe that for $e \in V \times V$ and $P_1, P_2, P_3 \in \widetilde{V}$, if $e \parallel (P_1, P_2)$, $e \parallel (P_2, P_3)$ and $P_2 \subseteq P_3$, then $e \parallel (P_1, P_3)$. Similarly, for $\tilde{e} \in \widetilde{V} \times \widetilde{V}$ and $S_1, S_2, S_3 \in V$, if $(S_1, S_2) \parallel \tilde{e}$, $(S_2, S_3) \parallel \tilde{e}$ and $S_2 \subseteq S_3$, then $(S_1, S_3) \parallel \tilde{e}$. Therefore, if $\mathcal{E} \in V \times V$ and $\widetilde{\mathcal{E}} \in \widetilde{V} \times \widetilde{V}$ are sets of increasing edges with $\mathcal{E} \parallel \widetilde{\mathcal{E}}$, we get $\mathcal{E} \parallel \widetilde{\mathcal{E}}^*$ and $\mathcal{E}^* \parallel \widetilde{\mathcal{E}}$. By applying this result to E, $\widetilde{E}$, E_ω and $\widetilde{E}_\omega$ (which only contain increasing edges), we get item 2, as well as: $E \parallel \widetilde{E}_\omega^*$ and $E_\omega^* \parallel \widetilde{E}$. Considering the definition of $\widetilde{E}_\omega$ as all increasing edges parallel to E, and of E_ω as all increasing edges parallel to $\widetilde{E}$, we therefore get $\widetilde{E}_\omega^* = \widetilde{E}_\omega$ and $E_\omega^* = E_\omega$. To conclude the proof of item 1, note that $\widetilde{E} \subseteq \widetilde{E}_\omega$ (by definition), and $E \subseteq E_\omega$ (since $E \parallel \widetilde{E}$). Lastly, items 3 and 4 follow directly from the definitions of $\widetilde{E}_\omega$ and E_ω. □

Note that we do not necessarily (or typically) have $E_\omega \parallel \widetilde{E}_\omega$. That is, some k-abstractly-reachable state can violate some inductive invariant (that is not k-provable).

3.3 Symmetric Connection Between Reachability and k-provability

We now discuss a particular symmetry of the induction duality structure. This symmetry, combined with the fact that we used the induction duality structure to define bounded-width incremental induction, leads to a surprising symmetric connection between reachability and k-provability. It is this connection, rather than the graph duality of Section 3.1, that we call *induction duality* in the rest of this paper. That is, the *induction dual* of any object represents its image under this symmetry. Table 1 lists the various concepts we have defined, relates them to the induction duality structure, and contrasts their induction duals with their Galois-connection duals, as discussed below.

Recall that the definition of induction-dual graphs and the $\parallel$ relation are symmetric, and observe that the properties stated in Theorem 3.8 are symmetric as well, w.r.t. swapping tilded and untilded objects. That is, Theorem 3.8 items 1 to 4 hold for $\langle V, \widetilde{V}, \bowtie, E, E_\omega, \widetilde{E}, \widetilde{E}_\omega \rangle$ iff they hold for $\langle \widetilde{V}, V, \bowtie^T, \widetilde{E}, \widetilde{E}_\omega, E, E_\omega \rangle$. Thus, we say that the following are *dual to each other w.r.t. induction duality*: states and predicates ($V \leftrightarrow \widetilde{V}$); transitions and k-width induction transitions ($E \leftrightarrow \widetilde{E}$); and inductive invariants and evidently k-abstractly-reachable sets, which we also call dual-inductive ($E_\omega \leftrightarrow \widetilde{E}_\omega$). Accordingly, reachability and k-provability are dual, as well as evident reachability and evident k-provability. Indeed, the reader can verify that any of the definitions of Section 3.2 respect this duality (e.g., the definition of evident k-provability, if we swap tilded and untilded objects, coincides with the definition of evident reachability). Note that the symmetry swaps $E_\omega \leftrightarrow \widetilde{E}_\omega$, even though (V, E_ω) and $(\widetilde{V}, \widetilde{E}_\omega)$ are typically not induction-dual graphs (i.e., $E_\omega \not\parallel \widetilde{E}_\omega$). Therefore, the reader should not be confused by this overloading of the term *induction duality*, and note that from now on, it refers to this symmetry of the induction duality structure and not to the graphs duality of Section 3.1.

Table 1. Concepts, how they relate to the induction duality structure (IDS), and their duals under induction duality and under the Galois connection. In the IDS column, a mnemonic notation is used: E^* stands for $\bigcup \{R \mid (\emptyset, R) \in E^*\}$ which defines the set of reachable states, $\widetilde{E}^*$ stands for $\bigcup \{Q \mid (\emptyset, Q) \in \widetilde{E}^*\}$ which defines the set of k-provable predicates, $\widetilde{E}_\omega$ stands for $(\emptyset, Q) \in \widetilde{E}_\omega$ which is the equivalent to Q being inductive, and E_ω stands for $(\emptyset, R) \in E_\omega$ which is equivalent to R being evidently k-abstractly-reachable (dual-inductive).

Concept	**IDS**	**Induction dual**	$\overset{\gamma}{\underset{\alpha}{\leftrightarrows}}$ **dual**
reachability	E^*	k-provability	invariance
k-provability	$\widetilde{E}^*$	reachability	k-abstract-reachability
invariance	$\alpha(E^*)$	k-abstract-reachability	reachability
k-abstract-reachability	$\gamma(\widetilde{E}^*)$	invariance	k-provability
inductiveness	$\widetilde{E}_\omega$	evident k-abstract-reachability	reachability[3]
evident k-abstract-reachability	E_ω	inductiveness	k-provability[3]

It is illustrative to contrast induction duality to the Galois connection $\langle \mathcal{P}(\mathbb{S}), \subseteq \rangle \overset{\gamma}{\underset{\alpha}{\leftrightarrows}} \langle \mathcal{P}(\mathbb{P}), \supseteq \rangle$. The Galois-connection dual of reachability is invariance, and the set of invariant predicates is by definition α applied to the set of reachable states. Similarly, the Galois-connection dual of k-provability is k-abstract-reachability, as the set of k-abstractly-reachable states is by definition γ applied to the set of k-provable predicates. Another important difference is that the Galois connection orders sets of states by $\subseteq$ and sets of predicates by $\supseteq$, whereas for induction duality, we swap between $V = \mathcal{P}_\omega(\mathbb{S})$ and $\widetilde{V} = \mathcal{P}_\omega(\mathbb{P})$, both ordered by $\subseteq$ (for example, both E transitions and $\widetilde{E}$ induction transitions are increasing according to $\subseteq$).

We are now equipped to develop a symmetric primal-dual search algorithm that explores reachability, invariance, k-provability, and k-abstract-reachability, utilizing all connections between them.

4 PRIMAL-DUAL HOUDINI

We now build on induction duality to develop a *primal-dual search algorithm* that explores the infinite space of states and predicates in a symmetric and primal-dual manner, i.e., new states drive the discovery of new predicates which drive the discovery of new states and so on. The resulting algorithm simultaneously underapproximates and overapproximates reachability, invariance, k-provability, and k-abstract-reachability, utilizing the connections between them.

Our plan is to: (i) recall the well-known Houdini procedure [Flanagan et al. 2001; Flanagan and Leino 2001] for computing the maximal inductive subset of a given set of predicates, (ii) systematically apply induction duality (i.e., the symmetry discussed in Section 3.3) to obtain the dual image of Houdini—a procedure for computing the maximal evidently k-abstractly-reachable subset of a given

[3] There is a subtlety in the relation of inductiveness and evident k-abstract-reachability to their Galois-connection duals, which is noted here for accuracy but otherwise not important for the rest of this paper. These concepts are an overapproximation of their Galois-connection duals that is coarser than necessary by the Galois connection itself. Intuitively, both invariance and inductiveness are Galois-connection dual to reachability as they overapproximate the set of reachable states; but the latter is a coarser overapproximation (a finite inductive invariant is not as precise as the possibly infinite set of all invariant predicates), while the former is the overapproximation that is inherent to the Galois connection (i.e., $\gamma \circ \alpha$ applied to the set of reachable states). The situation for evident k-abstract-reachability as it relates to k-provability is similar.

Algorithm 1 Primal Houdini

Input $P \in \mathcal{P}_\omega(\mathbb{P})$, $S \in \mathcal{P}_\omega(\mathbb{S})$
Output $P_H \subseteq P$ and $E_H \subseteq E$
▷ P_H is the maximal subset of P that is inductive
▷ E_H contains a CTI for every $P' \subseteq P$ s.t. $P' \not\subseteq P_H$

```
1: procedure PrimalHoudini(P, S)
2:     E_H := ∅ ; P_H := P
3:     while (some (X, Y) ∈ E|_S is a CTI for P_H) or
4:           (some (X, Y) ∈ E    is a CTI for P_H) do
5:         E_H := E_H ∪ {(X, Y)}
6:         P_H := P_H ∩ α(Y)                    (α as defined in eq. (1))
7:     return P_H, E_H
```

set of states, and (iii) combine both procedures into a primal-dual search algorithm, *primal-dual Houdini*, and explore its properties. Let us fix a set of states $\mathbb{S}$, a set of predicates $\mathbb{P}$, a set of initial states $\iota \subset \mathbb{S}$, a transition relation $\tau \subset \mathbb{S} \times \mathbb{S}$, an induction width $k \in \mathbb{N}$, and $\left\langle V, \widetilde{V}, \bowtie, E, E_\omega, \widetilde{E}, \widetilde{E}_\omega \right\rangle$ the corresponding induction duality structure (Definition 3.4).

We say that $(S, R) \in E$ is a *CTI for* $P \in \mathcal{P}_\omega(\mathbb{P})$ (counterexample to induction) if $(S, R) \not\Vdash (\emptyset, P)$, and that $(P, Q) \in \widetilde{E}$ is a *dual-CTI for* $S \in \mathcal{P}_\omega(\mathbb{S})$ if $(\emptyset, S) \not\Vdash (P, Q)$. Recall that P is inductive iff $(\emptyset, P) \in \widetilde{E}_\omega$, and by Theorem 3.8 item 3, P is inductive iff E does not contain a CTI for P. By Theorem 3.8 item 4, S is dual-inductive (i.e., $(\emptyset, S) \in E_\omega$) iff $\widetilde{E}$ does not contain a dual-CTI for S.

We assume the existence of two oracles that search for CTIs and dual-CTIs. Given P, the *E-solver* either finds a CTI for P in E or determines that P is inductive. The E-solver can be implemented using an SMT solver in a standard way. On the other side of the induction duality, given S, the *$\widetilde{E}$-solver* either finds a dual-CTI for S in $\widetilde{E}$ or determines that S is dual-inductive. The $\widetilde{E}$-solver is not standard, but the bound on induction width is intended to make it practical to implement, e.g. by adapting learning and synthesis techniques [Alur et al. 2013; Garg et al. 2014; Hu et al. 2019; Koenig et al. 2020]. Section 5 details our $\widetilde{E}$-solver for universally quantified formulas.

For $\mathcal{E}$ a subset of $V \times V$ or $\widetilde{V} \times \widetilde{V}$, we define $(\cup\mathcal{E}) = \{x \mid (X, Y) \in \mathcal{E} \wedge x \in X \cup Y\}$ and $\mathcal{E}|_X = \mathcal{E} \cap (\mathcal{P}(X) \times \mathcal{P}(X))$.

4.1 Primal Houdini and Dual Houdini

Algorithm 1 presents the Houdini procedure [Flanagan et al. 2001; Flanagan and Leino 2001], called here PrimalHoudini, for finding the maximal inductive subset of a given finite set of predicates P. We adapt the procedure to also take as input a set S of *preferred states* and return as output a set E_H of CTIs, as discussed below. Houdini eliminates up to $2^{|P|}$ potential inductive invariants with at most $|P|$ CTIs. The algorithm follows a simple greatest fixed point computation loop. Each iteration checks if a CTI exists for the current set of predicates, and if so removes the predicates violated by the post-state of the CTI. For our purposes here it is useful for PrimalHoudini to return the set of CTIs encountered during the fixed point computation. It is also useful for us to provide PrimalHoudini with a finite set of *preferred states* such that PrimalHoudini first checks for CTIs among those states. (The **or** in line 3 is short-circuiting.) If a CTI is not found in the preferred states then in line 4 PrimalHoudini uses the E-solver to find an arbitrary CTI or conclude that the current set of predicates is inductive.

The guarantees of PrimalHoudini irrespective of the preferred states are formalized in the following proposition.

Algorithm 2 Dual Houdini

Input $S \in \mathcal{P}_\omega(\mathbb{S}), P \in \mathcal{P}_\omega(\mathbb{P})$
Output $S_H \subseteq S$ and $\widetilde{E}_H \subseteq \widetilde{E}$
▷ S_H is the maximal subset of S that is dual-inductive
▷ $\widetilde{E}_H$ contains a dual-CTI for every $S' \subseteq S$ s.t. $S' \not\subseteq S_H$
1: **procedure** DualHoudini(S, P)
2: $\widetilde{E}_H := \emptyset$; $S_H := S$
3: **while** (some $(X, Y) \in \widetilde{E}|_P$ is a dual-CTI for S_H) **or**
4: (some $(X, Y) \in \widetilde{E}$ is a dual-CTI for S_H) **do**
5: $\widetilde{E}_H := \widetilde{E}_H \cup \{(X, Y)\}$
6: $S_H := S_H \cap \gamma(Y)$ (γ as defined in eq. (2))
7: **return** $S_H, \widetilde{E}_H$

Proposition 4.1 (PrimalHoudini). *PrimalHoudini$(P, _)$ terminates for any $P \in \mathcal{P}_\omega(\mathbb{P})$, and if (P_H, E_H) are returned and the E-solver is called N times then:*

(1) P_H is the maximal inductive subset of P: $(\emptyset, P_H) \in \widetilde{E}_\omega$ and P_H is has no outgoing $\widetilde{E}_\omega|_P$ edges.
(2) $P_H = P \cap \alpha(\cup E_H)$;
(3) for any $P' \subseteq P$ such that $P' \not\subseteq P_H$, some $(S', R') \in E_H$ is a CTI for P', i.e., $E_H \not\Vdash (\emptyset, P')$; and
(4) $N \leq |E_H| \leq |P| - |P_H|$.

Proof. If we let N count the number of E-solver calls so far, then the conjunction of items 2 to 4 is a loop invariant. Termination is guaranteed because $|P_H|$ strictly decreases in every loop iteration. Item 1 is provided by the negation of the loop condition combined with item 3. □

The return value of P_H is uniquely determined by P (follows from Proposition 4.1 item 1), while that of E_H is not, and depends on the nondeterministic choices at lines 3 and 4, including the nondeterminism of the E-solver.

The following proposition shows PrimalHoudini utilizes (and implicitly identifies) evidently reachable preferred states to rule out all predicates violated by such states, a fact we will later use.

Proposition 4.2 (PrimalHoudini Preferred States). *For $P \in \mathcal{P}_\omega(\mathbb{P})$, $S \in \mathcal{P}_\omega(\mathbb{S})$, and $S_R \subseteq S$ that is evidently reachable, if PrimalHoudini(P, S) has an execution that returns (P_H, E_H) and performs N calls to the E-solver, then:*

(1) $N = \left|E_H \setminus E_H|_S\right| \leq |P \cap \alpha(S_R)| - |P_H|$; and
(2) if $P' \subseteq P$ and $S_R \not\models P'$ then $E_H|_S$ contains a CTI for P'.

Proof. By Theorems 3.2 and 3.8, $E|_{S_R}$ contains a CTI for any $Q \not\subseteq \alpha(S_R)$. This observation allows us to conclude that during the execution of PrimalHoudini(P, S), the E-solver can only be called after P_H becomes a subset of $\alpha(S_R)$. Formally, the following is a loop invariant: (i) $N = \left|E_H \setminus E_H|_S\right| \leq |P \cap \alpha(S_R)| - |P_H \cap \alpha(S_R)|$; and (ii) if $P' \subseteq P$, $S_R \not\models P'$, and $P' \not\subseteq P_H$ then some $(S', R') \in E_H|_S$ is a CTI for P'. Upon termination $S_R \models P_H$, reducing (i) to (1) and (ii) to (2). □

Dual Houdini. We now apply induction duality to obtain a dual version of Houdini. Recall the symmetry of the induction duality structure discussed in Section 3.3, which conveniently swaps tilded and untilded symbols. We obtain the dual version of Houdini by applying this transformation to Algorithm 1, which can be seen as PrimalHoudini applied to $\langle \widetilde{V}, V, \bowtie^T, \widetilde{E}, \widetilde{E}_\omega, E, E_\omega \rangle$ rather than $\langle V, \widetilde{V}, \bowtie, E, E_\omega, \widetilde{E}, \widetilde{E}_\omega \rangle$. Applying this transformation is sensical, since all the properties of the induction duality structure $\langle V, \widetilde{V}, \bowtie, E, E_\omega, \widetilde{E}, \widetilde{E}_\omega \rangle$ used by Algorithm 1 and Propositions 4.1 and 4.2

also hold for $\langle \widetilde{V}, V, \bowtie^T, \widetilde{E}, \widetilde{E}_\omega, E, E_\omega \rangle$. We call the resulting procedure DualHoudini, and spell it out in Algorithm 2. DualHoudini takes as input a set of states and a set of preferred predicates. DualHoudini relies on the *$\widetilde{E}$-solver* solver (Algorithm 2 line 4) to find a dual-CTI for the current set of states or determine that it is evidently k-abstractly-reachable (dual-inductive). As the induction dual of PrimalHoudini, DualHoudini satisfies the duals of Propositions 4.1 and 4.2, which we spell out below.

Proposition 4.3 (DualHoudini). *DualHoudini$(S, _)$ terminates for any $S \in \mathcal{P}_\omega(\mathbb{S})$, and if $(S_H, \widetilde{E}_H)$ are returned and the $\widetilde{E}$-solver is called $\widetilde{N}$ times then:*

(1) S_H is the maximal dual-inductive subset of S: $(\emptyset, S_H) \in E_\omega$ and S_H is has no outgoing $E_\omega|_S$ edges;
(2) $S_H = S \cap \gamma(\cup \widetilde{E}_H)$;
(3) for any $S' \subseteq S$ such that $S' \not\subseteq S_H$, some $(P, Q) \in \widetilde{E}_H$ is a dual-CTI for S', i.e., $(\emptyset, S') \not\models \widetilde{E}_H$; and
(4) $\widetilde{N} \leq \left|\widetilde{E}_H\right| \leq |S| - |S_H|$.

Proposition 4.4 (DualHoudini Preferred Predicates). *For $S \in \mathcal{P}_\omega(\mathbb{S})$, $P \in \mathcal{P}_\omega(\mathbb{P})$, and $P_R \subseteq P$ that is evidently k-provable, if DualHoudini(S, P) has an execution that returns $(S_H, \widetilde{E}_H)$ and performs $\widetilde{N}$ calls to the $\widetilde{E}$-solver, then:*

(1) $\widetilde{N} = \left|\widetilde{E}_H \setminus \widetilde{E}_H|_P\right| \leq |S \cap \gamma(P_R)| - |S_H|$; and
(2) if $S' \subseteq S$ and $S' \not\models P_R$ then $\widetilde{E}_H|_P$ contains a dual-CTI for S'.

Intuitively, Proposition 4.4 means that in a run of DualHoudini(S, P), any state from S that can be proven to be unreachable using a k-width proof that uses only preferred predicates (i.e., from P) will be eliminated during the fixed point iterations solely using preferred predicates.

4.2 Primal-Dual Search

We now combine PrimalHoudini and DualHoudini into a primal-dual search algorithm, PrimalDualHoudini, which is listed in Algorithm 3. The combined algorithm simultaneously underapproximates and overapproximates reachability, invariance, k-provability, and k-abstract-reachability, utilizing the connections between them listed in Table 1. The algorithm is symmetric w.r.t. the symmetry discussed in Section 3.3 (i.e., it would be its own induction dual), and benefits from Houdini's feature of eliminating an exponential set of potential proofs with a linear number of counter-proofs (CTIs or dual-CTIs) on both sides of the duality.

The key observation behind PrimalDualHoudini is that as PrimalHoudini$(P, _)$ computes the maximal inductive subset of P, it discovers new states via CTIs that explain why no larger subset of P is inductive. Dually, DualHoudini$(S, _)$ discovers new predicates via dual-CTIs that explain why no subset of S larger than its result is evidently k-abstractly-reachable (dual-inductive). Algorithm 3 thus alternates between Houdini in the primal and Houdini in the dual, learning states from the former and predicates from the latter.

Intuitively, a run of PrimalHoudini hopes the current set of predicates suffices to prove the safety property is invariant, and upon failing discovers new states that explain why not. Then, a run of DualHoudini hopes the current set of states suffices to prove the safety property is not k-provable, i.e., prove that an unsafe state is k-abstractly-reachable, and upon failing discovers new predicates that explain why not. We augment this ping-pong between invariance and k-abstract-reachability with least fixed point computations of reachability and k-provability that do not discover new states or predicates.

Algorithm 3 is seeded with an input *safety property* to check $p_0 \in \mathbb{P}$, and it maintains a set S of discovered states, a set P of discovered predicates, and four subsets of these: a set $S_R \subseteq S$ of reachable states, a set $S_I \subseteq S$ of k-abstractly-reachable states, a set $P_I \subseteq P$ of invariant predicates,

Algorithm 3 Primal-Dual Houdini

Input safety property $p_0 \in \mathbb{P}$

Output $\langle k\text{-PROVABLE}, P_R{\in}\mathcal{P}_\omega(\mathbb{P})\rangle \mid \langle \text{SAFE}, P_I{\in}\mathcal{P}_\omega(\mathbb{P})\rangle \mid \langle \text{UNSAFE}, S_R{\in}\mathcal{P}_\omega(\mathbb{S})\rangle \mid \langle k\text{-UNPROVABLE}, S_I{\in}\mathcal{P}_\omega(\mathbb{S})\rangle$

▷ if $\langle k\text{-PROVABLE}, P_R\rangle$ then $p_0 \in P_R$ and P_R is evidently k-provable

▷ if $\langle \text{SAFE}, P_I\rangle$ then $p_0 \in P_I$ and P_I is inductive

▷ if $\langle \text{UNSAFE}, S_R\rangle$ then $S_R \not\models p_0$ and S_R is evidently reachable (so p_0 is noninvariant)

▷ if $\langle k\text{-UNPROVABLE}, S_I\rangle$ then $S_I \not\models p_0$ and S_I is evidently k-abstractly-reachable (so p_0 is not k-provable)

1: **procedure** PRIMALDUALHOUDINI(p_0)
2: $P := \{p_0\}$; $P_I := \emptyset$; $P_R := \emptyset$; $S := \emptyset$; $S_I := \emptyset$; $S_R := \emptyset$ **ghost:** $P_G := \{p_0\}$; $S_G := \emptyset$
3: **while** $\top$ **do**
4: $P_I, E_H := \text{PRIMALHOUDINI}(P, S)$
5: $P_R := \text{REACH}(\widetilde{E}|_{P_I})$
6: **if** $p_0 \in P_R$ **then return** $\langle k\text{-PROVABLE}, P_R\rangle$
7: **if** $p_0 \in P_I$ **then return** $\langle \text{SAFE}, P_I\rangle$
8: $S := S \cup (\cup E_H)$ **ghost:** $S_G := S_G \cup (\cup E_H)$
9: **choose** S' **such that** $(S \cap \gamma(P_I)) \cup S_I \subseteq S' \subseteq S$
10: $S := S'$
11: $S_I, \widetilde{E}_H := \text{DUALHOUDINI}(S, P)$
12: $S_R := \text{REACH}(E|_{S_I})$
13: **if** $S_R \not\models p_0$ **then return** $\langle \text{UNSAFE}, S_R\rangle$
14: **if** $S_I \not\models p_0$ **then return** $\langle k\text{-UNPROVABLE}, S_I\rangle$
15: $P := P \cup (\cup \widetilde{E}_H)$ **ghost:** $P_G := P_G \cup (\cup \widetilde{E}_H)$
16: **choose** P' **such that** $(P \cap \alpha(S_I)) \cup P_I \subseteq P' \subseteq P$
17: $P := P'$
18: **procedure** REACH($\mathcal{E}$)
19: $\mathcal{R} := \emptyset$
20: **while** some $\mathcal{R}'$ s.t. $(\mathcal{R}, \mathcal{R} \cup \mathcal{R}') \in \mathcal{E}$ **do**
21: $\mathcal{R} := \mathcal{R} \cup \mathcal{R}'$
22: **return** $\mathcal{R}$

and a set $P_R \subseteq P$ of k-provable predicates. Initially, P contains only the input safety property, and all other sets are empty. Each iteration of PRIMALDUALHOUDINI starts by using PRIMALHOUDINI to compute the set of invariant predicates P_I (line 4), and using the REACH auxiliary procedure to compute the set of k-provable predicates P_R by following paths in $\widetilde{E}^*|_{P_I}$ (line 5). As a result of PRIMALHOUDINI, new states are discovered and added to S (line 8), so the iteration proceeds by updating the set of k-abstractly-reachable states S_I using DUALHOUDINI (line 11) and updating the set of reachable states S_R using REACH (line 12), which in this case simply follows paths in E (i.e., transitions of τ) to find an evidently reachable subset of S. As a result of DUALHOUDINI, new predicates are discovered and added to P (line 15), setting the stage for the next iteration.

In addition to this algorithmic skeleton, lines 9 and 10 may prune states violating a known invariant unless they are known to be k-abstractly-reachable, and similarly lines 16 and 17 may prune predicates violated by a k-abstractly-reachable state unless they are known to be invariant (the reason for these conditions is clarified by the proof of Theorem 4.7). The nondeterministic choices in lines 9 and 16 do not affect the algorithm's theoretical properties, and can be used to implement different pruning heuristics.

Ultimately, PRIMALDUALHOUDINI may terminate by determining that the safety property is: k-provable (line 6), invariant (line 7), noninvariant (line 13), or k-unprovable (line 14).

The correctness of PRIMALDUALHOUDINI rests on the following theorem.

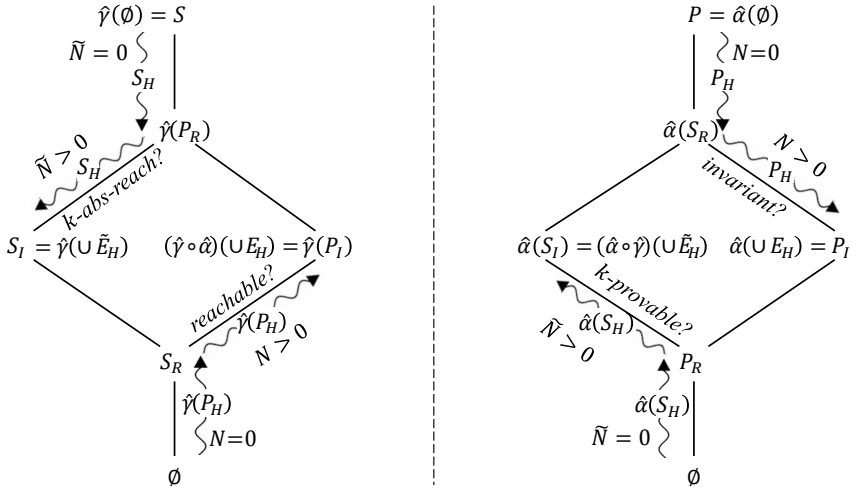

Fig. 4. Illustration of primal-dual Houdini, Propositions 4.1 to 4.4, and Theorem 4.5. The state of PrimalDualHoudini (Algorithm 3) is represented by S, P, S_I, P_I, S_R, P_R, E_H, $\widetilde{E}_H$, $\hat{\alpha}$, and $\hat{\gamma}$, which are as in Theorem 4.5 (see below). The state of PrimalHoudini (Algorithm 1) is represented by P_H and N, and the state of DualHoudini (Algorithm 2) is represented by S_H and $\widetilde{N}$, where N and $\widetilde{N}$ are the number of calls to the E- and $\widetilde{E}$-solvers as in Propositions 4.2 and 4.4. The left portion depicts the lattice of explored states $\langle \mathcal{P}(S), \subseteq \rangle$ and the right the lattice of explored predicates $\langle \mathcal{P}(P), \subseteq \rangle$, with straight lines denoting $\subseteq$, e.g., $P_I \subseteq \hat{\alpha}(S_R)$ and $S_I \subseteq \hat{\gamma}(P_R)$. Questions (e.g. *invariant?*) label set differences that represent *uncertainties*, e.g., it is unknown if the predicates in $\hat{\alpha}(S_R) \setminus P_I$ are invariant. The iterations of PrimalHoudini and DualHoudini are illustrated using wavy arrows, depicting the progression of P_H from P to $\hat{\alpha}(S_R)$ without E-solver calls ($N = 0$) and then to the return value P_I by performing E-solver calls ($N > 0$); and similarly the progression of S_H from S to $\hat{\gamma}(P_R)$ without $\widetilde{E}$-solver calls ($\widetilde{N} = 0$) and then to the return value S_I by $\widetilde{E}$-solver calls ($\widetilde{N} > 0$), as per Propositions 4.2 and 4.4. Induction duality and Galois-connection duality as discussed in Section 3.3 and Table 1 both manifest as symmetries between the left and right portions of this figure. Vertical reflection corresponds to the $\langle \mathcal{P}(S), \subseteq \rangle \underset{\hat{\alpha}}{\overset{\hat{\gamma}}{\leftrightarrows}} \langle \mathcal{P}(P), \supseteq \rangle$ Galois connection; letting $x \leftrightarrow y$ denote that an object x from the left portion is related to an object y from the right portion via *vertical* reflection, we have: $\hat{\gamma}(\emptyset) = S \leftrightarrow \emptyset$, $\hat{\gamma}(P_R) \leftrightarrow P_R$, $S_I \leftrightarrow \hat{\alpha}(S_I)$, $\hat{\gamma}(P_I) \leftrightarrow P_I$, $S_R \leftrightarrow \hat{\alpha}(S_R)$, $\emptyset \leftrightarrow P = \hat{\alpha}(\emptyset)$, *k-abs-reach?* $\leftrightarrow$ *k-provable?* and *reachable?* $\leftrightarrow$ *invariant?*. Horizontal reflection corresponds to induction duality; letting $x \leftrightarrow y$ denote that x is related to y via *horizontal* reflection, we have: $S \leftrightarrow P$, $\hat{\gamma}(P_R) \leftrightarrow \hat{\alpha}(S_R)$, $S_I \leftrightarrow P_I$, $\hat{\gamma}(P_I) \leftrightarrow \hat{\alpha}(S_I)$, $S_R \leftrightarrow P_R$, $\emptyset \leftrightarrow \emptyset$, *k-abs-reach?* $\leftrightarrow$ *invariant?* and *reachable?* $\leftrightarrow$ *k-provable?*. (Compare to Table 1.)

Theorem 4.5 (PrimalDualHoudini Invariants). *The following invariants hold in Algorithm 3 both at line 6 and at line 13, letting $\hat{\alpha} = \lambda X.P \cap \alpha(X)$ and $\hat{\gamma} = \lambda X.S \cap \gamma(X)$:*

(1) P_I is the maximal inductive subset of P;
(2) P_R is the maximal evidently k-provable subset of P;
(3) S_I is the maximal evidently k-abstractly-reachable subset of S;
(4) S_R is the maximal evidently reachable subset of S;
(5) $P_I = \hat{\alpha}(\cup E_H)$ and if $P' \subseteq P_I$, $Q' \subseteq P$, and $Q' \not\subseteq P_I$ then $E_H \not\Vdash (P', Q')$; and
(6) $S_I = \hat{\gamma}(\cup \widetilde{E}_H)$ and if $S' \subseteq S_I$, $R' \subseteq S$, and $R' \not\subseteq S_I$ then $(S', R') \not\Vdash \widetilde{E}_H$.

Proof. By combining Propositions 4.1 and 4.3, the observation that $\textsc{Reach}(E|_{S'})$ is the maximal evidently reachable subset of S' and $\textsc{Reach}(\widetilde{E}|_{P'})$ is the maximal evidently k-provable subset of P', and the connections between invariance, k-provability, k-abstract-reachability, and reachability explored in Section 3.2 and summarized in the Galois connection column of Table 1. □

As a corollary, we get the following guarantees when PrimalDualHoudini terminates.

Corollary 4.6 (PrimalDualHoudini Partial Correctness). *If PrimalDualHoudini(p_0) terminates it returns one of:*

(1) $\langle k$-provable, $P_R\rangle$ with $p_0 \in P_R$ and P_R evidently k-provable;
(2) $\langle$safe, $P_I\rangle$ with $p_0 \in P_I$ and P_I inductive;
(3) $\langle$unsafe, $S_R\rangle$ with $S_R \not\models p_0$ and S_R evidently reachable; or
(4) $\langle k$-unprovable, $S_I\rangle$ with $S_I \not\models p_0$ and S_I evidently k-abstractly-reachable.

Proof. By conjoining the invariants stated in Theorem 4.5 with each of the four termination conditions of Algorithm 3. □

Note that while a k-unprovable result means there is no proof with induction width up to k, a safe result includes a monolithic inductive invariant and may use predicates that are not k-provable. Therefore, in some cases both safe and k-unprovable are valid outputs of PrimalDualHoudini. This allows PrimalDualHoudini to be *opportunistic*, in the sense that it will use a proof of induction width $> k$ if it happens to find the right predicates.

Figure 4 illustrates the relationships between S_I, P_I, S_R, P_R, E_H and $\widetilde{E}_H$ (Theorem 4.5), as well as the internal computations of PrimalHoudini and DualHoudini. It highlights the two dualities that are at play: the Galois connection (vertical reflection) and induction duality (horizontal reflection) as also listed in Table 1. Figure 4 includes a detailed caption explaining the diagram. The progression of PrimalDualHoudini is a monotonic process of knowledge discovery. With each iteration, predicates may be discovered to be invariant (P_I), k-provable (P_R), noninvariant ($\notin \alpha(S_R)$), or k-unprovable ($\notin \alpha(S_I)$). Similarly, states may be discovered to be k-abstractly-reachable (S_I), reachable (S_R), not k-abstractly-reachable ($\notin \gamma(P_R)$), or unreachable ($\notin \alpha(P_I)$). Note that S_I and P_I and are incomparable, since: (i) a state s may be known to be both k-abstractly-reachable and unreachable ($s \in S_I \setminus \gamma(P_I)$), in terms of predicates, a predicate p may be known to be both invariant and k-unprovable ($p \in P_I \setminus \alpha(S_I)$); and (ii) a state s may be neither known to be k-abstractly-reachable nor known to be unreachable ($s \in \hat{\gamma}(P_I) \setminus S_I$), in terms of predicates, a predicate p may be neither known to be invariant nor known to be k-unprovable ($p \in \alpha(S_I) \setminus P_I$). The algorithm's progress, as well as its cost in terms of calls to the E- and $\widetilde{E}$-solvers, is driven by knowledge gaps (using Propositions 4.2 and 4.4): predicates that are possibly invariant but not yet proven lead to calls to the E-solver that discover new states (the $N > 0$ edge in Figure 4), and states possibly k-abstractly-reachable but not yet proven to be so lead to $\widetilde{E}$-solver calls that discover new predicates (the $\widetilde{N} > 0$ edge).

We now consider PrimalDualHoudini's termination, both theoretically and heuristically.

4.3 Progress and Termination

We now analyze the progress and termination of Algorithm 3. Proofs in this section highlight the primal-dual interaction between states and predicates: progress on predicates requires the right states and progress on states requires the right predicates.

Not getting stuck. There are two ways that PrimalDualHoudini could fail to learn new information, and thus cease to make progress: one is for PrimalHoudini and DualHoudini to only return known states and predicates (leaving S and P unchanged in lines 8 and 15), and the other is for the pruning in lines 10 and 17 to prune states and predicates that are rediscovered later, such that every new state or predicate discovered has already been pruned before. The following theorem uses ghost variables S_G and P_G to track all states and predicates ever considered (lines 2, 8, and 15), and shows these sets must increase at every non-final iteration. That is, every iteration (except that in which the algorithm terminates) explores at least one new state and one new predicate.

Theorem 4.7 (PrimalDualHoudini Exploration). *Both $|S_G|$ and $|P_G|$ increase on every iteration of PrimalDualHoudini that does not immediately lead to the algorithm's termination.*

The proof of Theorem 4.7 (given below) provides insight into the primal-dual nature of the algorithm, as it critically depends on the fact that the input to primal Houdini comes from the output of dual Houdini, and vice versa. The key to proving Theorem 4.7 is following pair of lemmas.

Lemma 4.8 (State Exploration). *For $S \in \mathcal{P}_\omega(\mathbb{S})$, $(S_H, \widetilde{E}_H) = \textit{DualHoudini}(S, \_)$, $p_0 \in \mathbb{P}$ such that $S_H \models p_0$, $P \in \mathcal{P}_\omega(\mathbb{P})$ such that $p_0 \in P$ and $(\cup\widetilde{E}_H) \subseteq P$, and $(P_H, E_H) = \textit{PrimalHoudini}(P, \_)$, either $\{p_0\} \cup (\cup\widetilde{E}_H) \subseteq P_H$ or there exists $(S', R') \in E_H$, a CTI for $\{p_0\} \cup (\cup\widetilde{E}_H)$ such that $(S' \cup R') \notin S$.*

Proof. Let $P' = \{p_0\} \cup (\cup\widetilde{E}_H)$. By Proposition 4.1, either $P' \subseteq P_H$ or there exists $(S', R') \in E_H$, a CTI for P'. To see that in the latter case we must have $(S' \cup R') \notin S$, assume the contrary. Then we must have: (i) $S' \subseteq S$, (ii) $R' \subseteq S$, (iii) $(S', R') \in E$, (iv) $S' \models P'$, (v) $R' \not\models P'$, and (vi) $S_H = S \cap \gamma(P')$ (by Proposition 4.3 item 2 and the fact that $S_H \models p_0$). By combining items i, iv and vi we get $S' \subseteq S_H$ and by combining items ii, v and vi we get $R' \notin S_H$. By Proposition 4.3 item 1, S_H is the maximal dual-inductive subset of S. However, from the above it follows that $S_H \cup R'$ is also dual-inductive, contradicting the maximality of S_H. To see that $S_H \cup R'$ is dual-inductive, note that if $(P'', Q'') \in \widetilde{E}$ is a dual-CTI for $S_H \cup R'$ then we must have $S_H \cup R \models P''$ and $S_H \cup R \not\models Q''$. Since S_H is dual-inductive we must have $R \not\models Q''$ (otherwise (P'', Q'') is a dual-CTI for S_H), and since $S' \subseteq S_H$ we obtain $E \ni (S', R') \not\parallel (P'', Q'') \in \widetilde{E}$ contradicting the fact that $E \parallel \widetilde{E}$. □

Lemma 4.9 (Predicate Exploration). *For $P \in \mathcal{P}_\omega(\mathbb{P})$, $(P_H, E_H) = \textit{PrimalHoudini}(P, \_)$, $S \in \mathcal{P}_\omega(\mathbb{S})$ such that $(\cup E_H) \subseteq S$, and $(S_H, \widetilde{E}_H) = \textit{DualHoudini}(S, \_)$, either $(\cup E_H) \subseteq S_H$ or there exists $(P', Q') \in \widetilde{E}_H$, a dual-CTI for $(\cup E_H)$ such that $(P' \cup Q') \notin P$.*

Proof. This lemma is induction dual to Lemma 4.8 (ignoring p_0) and the proof is analogous, by setting $S' = (\cup E_H)$ and observing that a dual-CTI for S' cannot solely use predicates from P. □

Intuitively, in the context of PrimalDualHoudini, Lemma 4.8 means that if the input to PrimalHoudini includes the safety property p_0 and the output of a previous DualHoudini run, then its output is guaranteed to either prove p_0 or explore a new state. Similarly, Lemma 4.9 means that if the input to DualHoudini includes the output of a previous PrimalHoudini run, then its output is guaranteed to either prove that p_0 is k-unprovable (since in this context $(\cup E_H) \not\models p_0$) or explore a new predicate. We note that the lemmas are incorrect without the conditions $(\cup\widetilde{E}_H) \subseteq P$ (Lemma 4.8) and $(\cup E_H) \subseteq S$ (Lemma 4.9), which demonstrates the critical role of PrimalHoudini in the discovery of *predicates* and that of DualHoudini in the discovery of *states*, highlighting the primal-dual interaction.

We now use Lemmas 4.8 and 4.9 to prove Theorem 4.7.

Proof of Theorem 4.7. Let us first show that we can ignore the pruning by Algorithm 3 lines 10 and 17. States are only pruned if they violate a predicate known to be invariant, and such predicates are never pruned. Thus, a state that has been pruned for violating some $p \in P_I$ never has a chance to be discovered again by PrimalHoudini, since $p \in P_I$ is an invariant from that point onward. A similar argument holds for predicates pruned by states known to be k-abstractly-reachable. We now prove the theorem ignoring the pruning, i.e., as if $S = S_G$ and $P = P_G$.

Every call to PrimalHoudini in line 4 satisfies the conditions of Lemma 4.8, and therefore if it does not prove p_0 (i.e., $p_0 \notin P_I$) then it must discover a new state. This argument holds for the first iteration as well, since $(\emptyset, \emptyset) = \textit{DualHoudini}(\emptyset, \_)$. Similarly, every call to DualHoudini in line 11 satisfies the conditions of Lemma 4.9 while in addition satisfying $(\cup E_H) \not\models p_0$, and therefore if it does not prove p_0 to be k-unprovable then it must discover a new predicate. □

Making actual progress. Theorem 4.7 shows that as long as PrimalDualHoudini has not terminated, it will keep exploring new states and predicates. The question then arises whether these new states and predicates bring the algorithm any closer to termination, i.e., closer to finding a proof (inductive invariant, k-provable or not) or a counter-proof (reachable or k-abstractly-reachable unsafe state). The fact that new states are only learned via CTIs to the current set of predicates, and that new predicates are only learned via dual-CTIs to the current set of states, restricts PrimalDualHoudini's exploration of the search space. This restriction is desirable as it potentially provides heuristic guidance, but it can possibly be too restrictive by preventing the algorithm from finding a solution. This can happen if at some point PrimalDualHoudini is unable to learn any useful state or predicate, i.e., it will satisfy Theorem 4.7 by learning states and predicates that are not part of either a proof or a counter-proof. The following two theorems show that if p_0 is k-provable or noninvariant, learning states and predicates from CTIs and dual-CTIs is not too restrictive. That is, each iteration of PrimalDualHoudini has the opportunity to learn a new predicate/state from the proof/counter-proof.

Theorem 4.10 (Possible State Progress). *If $\pi \in E^*$, $n = |\pi|$, $\pi_0 = \emptyset$, and $\pi_n \not\models p_0$ (i.e., p_0 is noninvariant with counterexample π) then in every iteration of PrimalDualHoudini(p_0) there is a possible execution of PrimalHoudini that increases $S \cap \left(\bigcup_{i=0}^{n} \pi_i\right)$ (i.e., learns a new state from π).*

Proof of Theorem 4.10. We obtain the desired execution of PrimalHoudini(P, S) (Algorithm 3 line 4) as an execution of PrimalHoudini$(P, S \cup \Pi)$, where we let $\Pi = \bigcup_{i=0}^{n} \pi_i$. This is justified by the fact that any possible execution of PrimalHoudini$(P, S \cup \Pi)$ is a possible execution of PrimalHoudini(P, S). To see that any execution of PrimalHoudini$(P, S \cup \Pi)$ learns a new state from Π, apply Proposition 4.2 item 2 for $P' = \{p_0\} \cup (\cup \widetilde{E}_H)$ and obtain $(S', R') \in E_H$, a CTI for P' such that $S' \cup R' \subseteq S \cup \Pi$. To see that $(S' \cup R') \not\subseteq S$, we can apply the same reasoning used in the proof of Lemma 4.8, noting that, as in the proof of Theorem 4.7, the conditions of Lemma 4.8 hold in every iteration of PrimalDualHoudini. □

Theorem 4.11 (Possible Predicate Progress). *If $\tilde{\pi} \in \widetilde{E}^*$, $n = |\tilde{\pi}|$, $\tilde{\pi}_0 = \emptyset$, and $p_0 \in \tilde{\pi}_n$ (i.e., p_0 is k-provable with proof $\tilde{\pi}$) then in every iteration of PrimalDualHoudini(p_0) there is a possible execution of DualHoudini that increases $P \cap \left(\bigcup_{i=0}^{n} \tilde{\pi}_i\right)$ (i.e., learns a new predicate from $\tilde{\pi}$).*

Proof. This theorem is the induction dual of Theorem 4.10 and the proof is analogous, observing that in all but the last iteration of PrimalDualHoudini we will have $(\cup E_H) \not\models p_0$ and therefore $(\cup E_H) \not\models \bigcup_{i=0}^{n} \tilde{\pi}_i$, so we can apply Proposition 4.4 as well as the reasoning used in the proof of Lemma 4.9 to obtain a new predicate from $\bigcup_{i=0}^{n} \tilde{\pi}_i$ as part of the dual-CTI for $(\cup E_H)$. □

Another way to understand Theorems 4.10 and 4.11 by considering *angelic nondeterminism*, i.e., a semantics under which nondeterministic choices are resolved in favor of termination. If the nondeterminism of the $\widetilde{E}$-solver (in case p_0 is k-provable) or the E-solver (in case p_0 is noninvariant) is angelic, then progress towards termination will be made on each iteration of PrimalDualHoudini. Therefore, under angelic nondeterminism, PrimalDualHoudini terminates in a number of iterations given by the size of the proof or counter-proof, as formalized by the following corollary.

Corollary 4.12 (PrimalDualHoudini Angelic Termination). *If $\pi \in E^*$, $n = |\pi|$, $\pi_0 = \emptyset$, and $\pi_n \not\models p_0$, an angelic E-solver can force PrimalDualHoudini to terminate (returning UNSAFE or k-UNPROVABLE) after at most $\left|\bigcup_{i=0}^{n} \pi_i\right|$ iterations. If $\tilde{\pi} \in \widetilde{E}^*$, $n = |\tilde{\pi}|$, $\tilde{\pi}_0 = \emptyset$, and $p_0 \in \tilde{\pi}_n$, an angelic $\widetilde{E}$-solver can force PrimalDualHoudini to terminate (returning k-PROVABLE or SAFE) after at most $\left|\bigcup_{i=0}^{n} \tilde{\pi}_i\right|$ iterations.*

Like Theorem 4.7, Theorems 4.10 and 4.11 depend on the fact that the input to primal Houdini comes from the output of dual Houdini, and vice versa. To find a proof (inductive invariant) it is critical to find the right states, and to find a counter-proof it is critical to find the right predicates, even though states do not appear in proofs nor predicates in counter-proofs. Thus, these theorems capture properties of the *primal-dual combination* of primal and dual Houdini.

Termination by stratification. In practice, we cannot hope for angelic E- and $\widetilde{E}$-solvers. However, the following theorem shows that we can guarantee termination by *stratification* [Jhala and McMillan 2006], that is, partitioning $\mathbb{S}$ and $\mathbb{P}$ into infinite sequences of finite *layers* and constructing E- and $\widetilde{E}$-solvers that return a solution from the lowest possible layer. In such a setting, PrimalDualHoudini is guaranteed to terminate if p_0 is k-provable or noninvariant.

Theorem 4.13 (PrimalDualHoudini Termination by Stratification). *If $\mathbb{S} = \bigcup_{n=1}^{\infty} \mathbb{S}_n$ and $\mathbb{P} = \bigcup_{n=1}^{\infty} \mathbb{P}_n$ such that every $\mathbb{S}_n$ and $\mathbb{P}_n$ is finite, and the E-solver and the $\widetilde{E}$-solver always return a solution from $E|_{\bigcup_{n=1}^{L} \mathbb{S}_n}$ and $\widetilde{E}|_{\bigcup_{n=1}^{L} \mathbb{P}_n}$ with the smallest L possible, then PrimalDualHoudini(p_0) terminates for every p_0 that is k-provable or noninvariant.*

Proof. Suppose p_0 is k-provable, and let $L \in \mathbb{N}$ be such that $P_L = \bigcup_{n=1}^{L} \mathbb{P}_n$ contains a k-width proof of p_0. For a stratified $\widetilde{E}$-solver, every execution of DualHoudini(S, P) in Algorithm 3 line 11 can be seen as an execution of DualHoudini$(S, P \cup P_L)$. For such an execution, by reasoning similar to that used in the proofs of Lemma 4.9 and Theorem 4.11, we obtain that in every iteration of PrimalDualHoudini, a new predicate from P_L must be discovered (pruning can be ignored as argued in the proof of Theorem 4.7). Therefore, PrimalDualHoudini must terminate, since P_L is finite. The case in which p_0 is noninvariant is analogous (it is induction dual to the case of k-provable): letting $L \in \mathbb{N}$ be the such that $S_L = \bigcup_{n=1}^{L} \mathbb{S}_n$ contains an execution trace violating p_0, and regarding executions of PrimalHoudini(P, S) in Algorithm 3 line 4 as PrimalHoudini$(P, S \cup S_L)$, we can apply reasoning similar to the proofs of Lemma 4.8 and Theorem 4.10 to show that every iteration of PrimalDualHoudini must discover a new state from S_L. □

As an example for stratification, we can partition the states by some size parameter (e.g., number of nodes in a distributed system) such that for a fixed size there are finitely many states, and implement an E-solver that finds the smallest CTI possible. Similarly, we can partition the predicates by some complexity metric (e.g., number of quantifiers or terms), and implement an $\widetilde{E}$-solver that returns the simplest possible dual-CTI (using techniques such as [Koenig et al. 2020]). Such stratification may also be heuristically helpful as it focuses the search on simple predicates and small states.

4.4 Heuristic Discussion

The practical effectiveness of primal-dual Houdini depends mostly on the amount of uncertainty that develops and the guidance states provide for predicates and vice versa. By Propositions 4.2 and 4.4 and as illustrated in Figure 4, the number of E- and $\widetilde{E}$-solver calls and new states and predicates added in each iteration of primal-dual Houdini is linear in the *uncertainty*, i.e., predicates whose invariance is unknown ($\hat{\alpha}(S_R) \setminus P_H$) and states whose k-abstract-reachability is unknown ($\hat{\gamma}(P_R) \setminus S_H$). Thus, in the worst case, S and P grow exponentially in the number of primal-dual Houdini iterations. However, if most discovered states and predicates take a small number of iteration from being discovered until they become known to be invariant/noninvariant or k-abstractly-reachable/k-abstractly-unreachable, then exponential growth may be avoided. The pruning in Algorithm 3 lines 10 and 17 can also help by removing states known to be unreachable even if they may be k-abstractly-reachable (and dually for k-unprovable predicates), but that presents a heuristic trade-off, as it may prune some useful states/predicates.

Guidance provided by states is crucial even for safe/k-provable instances, and the guidance provided by predicates is crucial even for unsafe/k-unprovable examples. For example, when searching for an invariant for a safe/k-provable instance, discovering reachable states prunes the search space for invariants, and discovering backward reachable states and disconnected CTIs directs the search. This interaction is present in IC3/PDR as well, but IC3/PDR only uses invariants proven to hold (at a certain frame), and states known to be backward reachable. In contrast, primal-dual Houdini uses both forward reachable, backward reachable, and disconnected paths in both (V, E) and $(\widetilde{V}, \widetilde{E})$ in a systematic and symmetric way, potentially leading to better heuristic guidance.

Unlike IC3/PDR and interpolation, primal-dual Houdini does not consider distance from initial states and unsafe states for heuristic guidance, as it uses dual-CTIs to guide the search for predicates. Recall that by Corollary 3.3, a dual-CTI $(P, Q) \in \widetilde{E}$ means every execution that violates Q must violate P, but the violation can be *simultaneous*. In contrast, IC3's notion of Q being inductive relative to P entails P must be violated *before* Q. This indifference of primal-dual Houdini to path length is in contrast to both IC3/PDR and interpolation, thus presenting a different and potentially complementary heuristic.

Finally, as primal-dual Houdini is based on the notion of bounded-width incremental induction, we expect it to perform well for systems that are k-provable for small k. Our evaluation provides some evidence that this bounded proof notion is a good fit for distributed protocols. The applicability of bounded-width proofs for additional domains is an important question for future research, as is the possibility of generalizing primal-dual Houdini to other bounded proof notions.

5 PROTOTYPE IMPLEMENTATION & EVALUATION

We describe our prototype implementation of primal-dual Houdini in the domain of distributed protocol verification using universally quantified invariants and evaluate it by comparing to several state-of-the-art tools.

Our implementation considers the set of states $\mathbb{S}_{\forall^*}$ and the set of predicates $\mathbb{P}_{\forall^*}$. $\mathbb{S}_{\forall^*}$ is the set of finite first-order structures. $\mathbb{P}_{\forall^*}$ is the set of universally quantified clauses of the form $\forall x_1, \cdots, x_n.\ \left(\bigwedge_{1 \le i < j \le n} x_i \neq x_j\right) \rightarrow \varphi$, where φ is a disjunction of literals. That is, the predicates are universally quantified clauses whose variables have *distinct* interpretations. Our prototype currently supports induction width bound $k = 1$. Our E- and $\widetilde{E}$-solvers are stratified (by structure size and number of quantifiers), and termination of the entire algorithm is therefore guaranteed for k-provable or unsafe instances (Theorem 4.13).

The transition relation τ and the safety property p_0 are expressed symbolically as formulas, such that all resulting verification conditions are in the decidable EPR (effectively propositional reasoning) fragment [Padon et al. 2017]. In this framework, the E-solver can be implemented by checking the verification conditions given by eqs. (11) and (12) using an SMT solver that supports EPR formulas. Our prototype follows this approach and uses both Z3 [de Moura and Bjørner 2008] and CVC4 [Barrett et al. 2011] as EPR solvers.

5.1 Implementing an $\widetilde{E}$-Solver

Primal-dual Houdini relies on an $\widetilde{E}$-solver to generate dual-CTIs. We implement the $\widetilde{E}$-solver using Algorithm 4, which relies on both an E-solver and a *SEP-solver*, defined below.

For states $\mathbb{S}$ and predicates $\mathbb{P}$, a *separation query with implications* [Koenig et al. 2020] or *SEP-query* is $\mathrm{SEP}(S^+, S^-, S^{\rightarrow}, k)$ where $S^+, S^- \in \mathcal{P}_\omega(\mathbb{S})$, $S^{\rightarrow} \in \mathcal{P}_\omega(\mathbb{S} \times \mathbb{S})$, and $k \in \mathbb{N}$. $\mathrm{SEP}(S^+, S^-, S^{\rightarrow}, k)$ is *satisfied* by $P \in \mathcal{P}_\omega(\mathbb{P})$, denoted $P \in \mathrm{SEP}(S^+, S^-, S^{\rightarrow}, k)$, if: (i) $|P| \le k$; (ii) for every $s \in S^+$, $s \models P$; (iii) for every $s \in S^-$, $s \not\models P$; and (iv) for every $(s, t) \in S^{\rightarrow}$ if $s \models P$ then $t \models P$. A SEP-query is

Algorithm 4 $\widetilde{E}$-Solver

Input $S \in \mathcal{P}_\omega(\mathbb{S})$
Output $(P, Q) \in \widetilde{E}$ a dual-CTI for S, or $\bot$ if S is dual-inductive

```
1:  procedure Ẽ-Solver(S)
2:    W := ∅; P := ∅
3:    while ⊤ do
4:      while some Q ∈ SEP(ι ∩ W, {t}, τ ∩ W × W, k) for some t ∈ S do
5:        if some (S′, R′) ∈ E such that a (S′, R′) ⊭ (P, P ∪ Q) then
6:          W := W ∪ S′ ∪ R′
7:        else return (P, P ∪ Q)
8:      if some {p} ∈ SEP(S, {t}, ∅, 1) for some t ∈ W then
9:        P := P ∪ {p}
10:       W := W ∩ γ(P)                                  (γ as defined in eq. (2))
11:     else return ⊥
```

satisfiable if some P satisfies it, and otherwise it is *unsatisfiable*. A *SEP-solver* is a procedure that given a SEP-query, either returns a P that satisfies it or determines that it is unsatisfiable.

Algorithm 4 uses an E-solver (line 5) and a SEP-solver (lines 4 and 8) to implement an $\widetilde{E}$-solver, which can be seen as an adaptation of ICE learning [Garg et al. 2014] to induction transitions. To find a dual-CTI $(P, P \cup Q)$, it alternates between SEP(_, _, _, k) queries used to find candidates for Q, and SEP(_, _, $\emptyset$, 1) queries used to monotonically grow P, both guided by counterexamples generated using the E-solver.

Algorithm 4 begins with an empty set of counterexamples, W, and an empty candidate for P (corresponding to $\top$), and maintains the invariants $S \models P$ and $W \models P$. Line 4 uses a SEP-query with implications to find a Q such that $(P, P \cup Q)$ is a possible dual-CTI for S. Since $S \models P$ is an invariant, it remains to find Q such that $S \not\models Q$ and $(P, P \cup Q) \in \widetilde{E}$. The former is ensured by including a state from S in S^-, and the latter amounts to $|Q| \leq k$ (ensured by using k in the SEP-query) and satisfying eqs. (11) and (12). In the SEP-query of line 4, eqs. (11) and (12) are relaxed by only considering initial states and transitions from W. If the query is satisfiable, line 5 checks the unrelaxed version of eqs. (11) and (12) (using the E-solver, which as discussed above is implemented by an SMT solver). If the equations hold, line 7 returns $(P, P \cup Q)$ which is a valid dual-CTI; otherwise, line 6 refines the relaxation by extending W. If the SEP-query for Q in line 4 is unsatisfiable, we turn to updating P and eliminating some states from W using the SEP-query without implications in line 8. If S and W are inseparable, we can conclude no dual-CTI exists.

Algorithm 4 may diverge by indefinitely checking Q candidates for a fixed P or by indefinitely increasing P. However, it is guaranteed to terminate if every state is only violated by finitely many predicates, which is the case for $\mathbb{S}_{\forall^*}$ and $\mathbb{P}_{\forall^*}$.

We implement a SEP-solver for $\mathbb{S}_{\forall^*}$ and $\mathbb{P}_{\forall^*}$ that returns a solution with the smallest possible number of quantifiers by reduction to SAT, similar to [Koenig et al. 2020], but using the notion of a *diagram*, which is also used by PDR$^\forall$ [Karbyshev et al. 2017].

5.2 Implementation Details

Our prototype is written in Python and implemented in the `mypyvy` verification system. It is publicly available as part of the open-source `mypyvy` project[4], as well as in the artifact supporting this paper [Padon et al. 2021]. Our prototype includes the following heuristics and engineering choices:

[4] https://github.com/wilcoxjay/mypyvy

- Our $\widetilde{E}$-solver and SEP-solver consider some predicates that are outside the formal definition of $\mathbb{P}_{\forall^*}$ by opportunistically dropping disequalities between variables, which effectively explores all subclauses of the negation of the diagram, similarly to PDR$^\forall$.
- In Algorithm 4 line 5, we greedily optimize the counterexample to prune as much of the search space as possible.
- When we search for CTIs (Algorithm 1 line 4), we try to extend known reachable states (similar to [Miltner et al. 2020]) and other states in S, before considering general CTIs.
- Our $\widetilde{E}$-solver only considers predicates that are implied by ι when strengthening P.
- For the nondeterministic choices in Algorithm 3 our implementation chooses $P' = P$ and $S' = (S \cap \gamma(P_I)) \cup S_I$. That is, we never prune predicates according to k-abstractly-reachable states, and always prune as many states as possible.
- Our E-solver uses both Z3 [de Moura and Bjørner 2008] and CVC4 [Barrett et al. 2011], run in parallel over several random seeds.

5.3 Benchmarks

We evaluate our prototype on distributed protocols verification benchmarks. All benchmarks have a universally quantified inductive invariant, and for most we also include unprovable variants that are either unsafe or do not have a universally quantified inductive invariant, which we constructed by over-strengthening the safety property, manually injecting a bug, or removing some ghost state required for a universally quantified invariant. Our benchmarks, described in more detail in the following list, are primarily asynchronous message-passing concurrent systems, such as consensus and cache-coherence protocols, drawn from previous works [Feldman et al. 2019; Hawblitzel et al. 2015; Ma et al. 2019; Padon et al. 2017, 2016; Taube et al. 2018; Wilcox et al. 2015].

- ring: A simple leader election in a ring protocol from [Padon et al. 2016].
- cons: A simplified version of paxos (see below), with leader election and decision on values but without rounds, together with an unsafe variant and a variant that is safe but unprovable with a universally quantified invariant.
- paxos: The single-decree Paxos [Lamport 2001] distributed algorithm modeled in EPR following [Padon et al. 2017], and instrumented by a derived relation capturing the notion of a *choosable value*; this example has a universal invariant which implies safety and makes crucial use of the derived relation; we also include three unprovable variants.
- spaxos: The Stoppable Paxos algorithm [Malkhi et al. 2008] modeled in EPR following [Padon et al. 2017] and augmented with a derived relation for choosable values; this is a very complex and challenging distributed algorithm.
- paxos-h: Another variant of paxos with a universal invariant, supported by additional history variables rather than the choosable derived relation; this variant is harder than paxos because the invariant itself must include parts of the definition of the choosable derived relation.
- spaxos-h: as above, but for Stoppable Paxos.
- locksrv: a mutual exclusion protocol from [Wilcox et al. 2015], with an unprovable variant;
- skv: a key-value store from [Hawblitzel et al. 2015], and two unprovable variants;
- skvr: a variant of sharded_kv that tolerates duplicate messages with sequence numbers; this example is challenging due to the high arity of the relations involved; we also include an unprovable variant that incorrectly reuses sequence numbers;
- cache: a model of a MESI cache coherence protocol from [Feldman et al. 2019], and an unprovable variant that sends incorrect memory values on the shared bus.

All benchmarks are openly available in the artifact supporting this paper [Padon et al. 2021].

Table 2. Comparison of pdH, PDR$^\forall$, FOL-IC3, SWISS, IC3PO, and DistAI on 10 safe and 10 unsafe benchmarks, with 10 runs using different random seeds. Each table entry lists how many runs ended in each result. For pdH, possible results are: "1p" for 1-PROVABLE, "s" for SAFE, "us" for UNSAFE, "1up " for 1-UNPROVABLE, and "to" for timeout at 24 hours or other runtime error. For the other tools, results are grouped to "c" for correct (safe or unsafe as appropriate) and "to." The columns labeled "time" list the mean wall-clock time in seconds for successful runs, i.e. excluding timeouts and errors. Entries that are not applicable are left blank.

		pdH						**PDR$^\forall$**			**FOL-IC3**			**SWISS**			**IC3PO**			**DistAI**		
	Example	1p	s	us	1up	to	time	c	to	time	c	to	time	c	to	time	c	to	time	c	to	time
Safe	ring	10	0		0	0	34	10	0	59	10	0	22	10	0	201	10	0	5	10	0	344
	cons	10	0		0	0	5760	10	0	202	10	0	2844	10	0	146	10	0	1572	10	0	1266
	paxos	10	0		0	0	2814	10	0	9256	0	10		10	0	6359	0	10		0	10	
	spaxos	6	0		0	4	32482	0	10		0	10		0	10		10	0	5632	0	10	
	paxos-h	2	0		0	8	25991	0	10		0	10		0	10		0	10		0	10	
	spaxos-h	0	0		0	10		0	10		0	10		0	10		0	10		0	10	
	locksrv	0	1		4	5	134	10	0	10	10	0	9	10	0	6744	10	0	3	10	0	2
	skv	0	0		10	0	43	10	0	3	10	0	22	10	0	6065	10	0	2	10	0	2
	skvr	0	0		0	10		2	8	15127	0	10		0	10		0	10		0	10	
	cache	0	0		0	10		0	10		0	10		0	10		10	0	757	0	10	
Unsafe or Unprovable	cons-u1			10	0	0	52	10	0	1	10	0	8				10	0	2			
	cons-u2			2	0	8	61802	10	0	63	10	0	1098				0	10				
	paxos-u1			0	10	0	17	10	0	2	10	0	26				0	10				
	paxos-u2			0	0	10		0	10		0	10					0	10				
	paxos-u3			10	0	0	266	10	0	1	10	0	35				10	0	2			
	locksrv-u			9	0	1	2050	10	0	10	10	0	28				10	0	2			
	skv-u1			10	0	0	15	10	0	1	10	0	19				10	0	1			
	skv-u2			10	0	0	261	10	0	6	10	0	60				10	0	1			
	skvr-u			0	0	10		0	10		0	10					0	10				
	cache-u			0	0	10		10	0	137	3	7	6541				10	0	7			

5.4 Results

Table 2 compares our prototype, pdH, with five state-of-the-art invariant inference techniques. PDR$^\forall$ is a variant of IC3/PDR for universally quantified invariants [Karbyshev et al. 2017]. FOL-IC3 is a variant of IC3/PDR for general quantified invariants [Koenig et al. 2020]. SWISS performs an explicit search for general quantified invariants leveraging incremental induction [Hance et al. 2021]. IC3PO is a variant of IC3/PDR that analyzes finite instances of protocols and lifts propositional clauses to quantified formulas [Goel and Sakallah 2021a,b]. DistAI performs an explicit search for universally quantified invariants guided by reachable states discovered by random simulation [Yao et al. 2021]. Unlike the other techniques, SWISS and DistAI cannot solve unsafe benchmarks, so we only compare to them on safe benchmarks. Each tool[5] was run on each benchmark 10 times with different random seeds. All experiments were performed on AWS EC2 `z1d.metal` instances, which "provide a Intel Xeon Scalable processor with a sustained all core frequency of up to 4.0 GHz." pdH and SWISS use parallelism, while the rest of the tools are sequential.

[5] For pdH and PDR$^\forall$, we used `mypyvy`[4] commit `5dcbb19`. For FOL-IC3, we used https://doi.org/10.1145/3395650. For SWISS, we used https://github.com/secure-foundations/SWISS commit `348b9c4`. For IC3PO we used https://github.com/aman-goel/fmcad2021exp commit `9d3a54b`. For DistAI we used https://github.com/VeriGu/DistAI commit `0ec9389`.

As seen in Table 2, performance varies across the benchmark suite and among the six tools. Some examples, like ring and cons-u1, are easy for all tools, while some examples, like spaxos-h and skvr-u, are hard and unsolved by all tools. All six techniques solve challenging benchmarks, and some solve benchmarks that are not solved by others. For example, PDR$^\forall$ uniquely solves the difficult skvr benchmark, which is beyond the reach of other techniques, while pdH is uniquely able to solve paxos-h. The spaxos example, which we consider the most algorithmically complex protocol in our benchmark suite, is only solved by pdH and IC3PO, and its more difficult variant spaxos-h is unsolved by any technique. For unsafe benchmarks, pdH is often able to discover concrete counterexamples, which indicates that the primal-dual search stimulates discovery of error traces.

Even with $k = 1$, pdH is able to solve some challenging benchmarks such as spaxos and paxos-h. However, the restricted induction width can hurt pdH even for simple examples. The locksrv and skv benchmarks are relatively easy examples for which most tools prove safety in seconds, while pdH either diverges or returns 1-UNPROVABLE (except for one lucky locksrv run). These examples are hard for pdH because their invariants consist of 7–9 predicates that are all mutually dependent. This fact also makes these examples more difficult for SWISS, but it is still able to solve them. In contrast, spaxos for example requires more predicates (and over a richer vocabulary), but is 1-provable. The correspondence between the theory on which pdH is based on and its behavior in practice is notable, as is the fact that the $\widetilde{E}$-solver can be made practical for challenging problems.

Overall, we view these results as encouraging: a new invariant inference algorithm, not a variant of an established technique like interpolation or IC3/PDR, originating from a theoretical treatment of duality, can cope with examples that are on the frontier of state-of-the-art techniques.

6 RELATED WORK

The interaction between proof and counter-proof has long been exploited as a heuristic in searching for proofs and counterexamples. A major example is counterexample-guided abstraction refinement (CEGAR) [Clarke et al. 2000] and its extensions to software verification such as predicate abstraction [Ball et al. 2001] and trace abstraction [Heizmann et al. 2009]. In this framework, an abstraction can be seen as a restriction on the proof search space. In searching this space, we may discover an *abstract counterexample*. This is a sequence of states that proves that there is no proof in the restricted space. The abstract counterexample gives us a heuristic for refining the abstraction, that is, expanding the proof search space. Our refinement must at least refute the abstract counterexample, and ideally should also generalize it.

CEGAR can be viewed as a *relaxation* of a constraint solving problem, in the style of the classical relaxation from integer linear programming (ILP) to linear programming (LP). A proof system can be thought of as constraints on the space of counterexamples. Abstraction corresponds to removal of constraints, which may give spurious solutions (abstract counterexamples). As in ILP, we *generalize* a spurious solution into a new constraint that rules it out, along with a class of spurious solutions. In this method, proofs and counter-proofs interact, but there is a distinct asymmetry. That is, we relax in the space of counterexamples, and restrict in the space of proofs. In primal-dual Houdini, we simultaneously expand the space of proofs and counter-proofs, and each restricts the future development of the other.

In ICE learning [Garg et al. 2014], we guess a proof in the form of a conjectured inductive invariant ϕ. If ϕ is not inductive, we get a CTI as a counter-proof. The CTIs are accumulated, acting as constraints on future proofs. This approach is also asymmetric, but in the opposite way to CEGAR. That is, in ICE learning we relax in the space of proofs and restrict in the space of counter-proofs. The primal-dual approach has several heuristic advantages over ICE learning. First, using Houdini, it eliminates large parts of the proof space with relatively little computation, while

ICE learning eliminates only one candidate invariant. Second, it has an inductive bias toward narrow proofs with simple predicates. Third, work done by the learner is not discarded as in ICE learning. Rather, the clauses of a candidate proof, like the CTIs, can be accumulated and re-used.

In Craig interpolation techniques [McMillan 2003, 2006], we typically relax in the space of proofs, e.g., by bounding executions. If the resulting proof is spurious, in the sense that it fails to generalize to an inductive invariant, we tighten the constraint by increasing the bound on execution steps.

IC3 [Bradley 2011, 2012; Somenzi and Bradley 2011], like primal-dual Houdini, monotonically builds up both proof and refutation attempts. IC3 differs in some key aspects, however. First, in IC3 inference of a new predicate Q are local, repairing a single CTI relative to an existing frame P, while in primal-dual Houdini they are global, driven by many CTIs and generating both P and Q as a dual-CTI. Second, IC3 uses only backward reachable states and predicates proven to be preserved by bounded executions, while primal-dual Houdini uses CTIs and dual-CTIs which may be forward reachable, backward reachable, or disconnected in the graphs of executions and incremental proofs. Third, IC3 also uses a notion of incremental induction, but the one we use here is both generalized to arbitrary width, and modified to obtain the symmetry of induction duality.

Sorcar [Neider et al. 2019] extends Houdini to a property-driven algorithm. It selects predicates from a known set based on relevance, where a predicate is deemed relevant if it eliminates a pre-state of a CTI. In primal-dual Houdini, if a dual-CTI (P, Q) is discovered by dual Houdini, then Q must eliminate some pre-state of a CTI from the previous run of primal Houdini. However, that is not the case for P, which is guided by the requirement that the dual-CTI must be an induction transition, and thus affected by all transitions and not just the discovered CTIs. Therefore, in addition to the fact that primal-dual Houdini discovers predicates and Sorcar does not, dual-CTIs provide a different refinement strategy compared to Sorcar's relevant predicates.

In all of the prior approaches, proofs and counter-proofs interact heuristically, but there is an asymmetry between the two. In this paper, we introduce the first approach in which proofs and counter-proofs are fully symmetric. Our duality theory also opens a path to possible discovery of primal-dual versions of additional algorithms such as IC3 and Sorcar.

7 CONCLUSION

We developed a symmetric duality between proofs by induction and counterexample traces, and saw how this duality can be exploited using a simple invariant generation algorithm, Houdini, such that the algorithm and its dual heuristically guide each other. A preliminary evaluation shows that this approach has certain strengths for difficult infinite-state verification problems, where it is easy to diverge in an infinite sequence of irrelevant generalizations. We see this as a promising start, and work remains to be done to better understand the inductive bias introduced by the primal-dual interaction, and to contain the states-predicates explosion problem. Moreover, the theory presented here opens a path to explore primal-dual versions of more advanced invariant generation algorithms, and may have further applications.

DATA AVAILABILITY STATEMENT

An artifact supporting this paper is openly available in Zenodo [Padon et al. 2021]. This artifact includes our prototype implementation of primal-dual Houdini, and both input files (benchmarks) and results (log files) for all six tools included in the experiments.

ACKNOWLEDGMENTS

We thank the anonymous reviewers, artifact evaluation reviewers, and shepherd for comments and suggestions which improved this paper. This work was supported by National Science Foundation grants CCF-1160904 and CCF-1409813 as well as a grant of cloud credits from Amazon Web Services.

REFERENCES

Rajeev Alur, Rastislav Bodík, Garvit Juniwal, Milo M. K. Martin, Mukund Raghothaman, Sanjit A. Seshia, Rishabh Singh, Armando Solar-Lezama, Emina Torlak, and Abhishek Udupa. 2013. Syntax-guided synthesis. In *Formal Methods in Computer-Aided Design, FMCAD 2013, Portland, OR, USA, October 20-23, 2013*. IEEE, 1–8. https://doi.org/10.1109/FMCAD.2013.6679385

Thomas Ball, Rupak Majumdar, Todd D. Millstein, and Sriram K. Rajamani. 2001. Automatic Predicate Abstraction of C Programs. In *Proceedings of the 2001 ACM SIGPLAN Conference on Programming Language Design and Implementation (PLDI), Snowbird, Utah, USA, June 20-22, 2001*, Michael Burke and Mary Lou Soffa (Eds.). ACM, 203–213. https://doi.org/10.1145/378795.378846

Clark W. Barrett, Christopher L. Conway, Morgan Deters, Liana Hadarean, Dejan Jovanovic, Tim King, Andrew Reynolds, and Cesare Tinelli. 2011. CVC4. In *Computer Aided Verification - 23rd International Conference, CAV 2011, Snowbird, UT, USA, July 14-20, 2011. Proceedings (Lecture Notes in Computer Science, Vol. 6806)*, Ganesh Gopalakrishnan and Shaz Qadeer (Eds.). Springer, 171–177. https://doi.org/10.1007/978-3-642-22110-1_14

Aaron R. Bradley. 2011. SAT-Based Model Checking without Unrolling. In *Verification, Model Checking, and Abstract Interpretation - 12th International Conference, VMCAI 2011, Austin, TX, USA, January 23-25, 2011. Proceedings (Lecture Notes in Computer Science, Vol. 6538)*, Ranjit Jhala and David A. Schmidt (Eds.). Springer, 70–87. https://doi.org/10.1007/978-3-642-18275-4_7

Aaron R. Bradley. 2012. Understanding IC3. In *Theory and Applications of Satisfiability Testing - SAT 2012 - 15th International Conference, Trento, Italy, June 17-20, 2012. Proceedings*. 1–14. https://doi.org/10.1007/978-3-642-31612-8_1

Aaron R. Bradley and Zohar Manna. 2008. Property-directed incremental invariant generation. *Formal Asp. Comput.* 20, 4-5 (2008), 379–405. https://doi.org/10.1007/s00165-008-0080-9

Edmund M. Clarke, Orna Grumberg, Somesh Jha, Yuan Lu, and Helmut Veith. 2000. Counterexample-Guided Abstraction Refinement. In *Computer Aided Verification, 12th International Conference, CAV 2000, Chicago, IL, USA, July 15-19, 2000, Proceedings (Lecture Notes in Computer Science, Vol. 1855)*, E. Allen Emerson and A. Prasad Sistla (Eds.). Springer, 154–169. https://doi.org/10.1007/10722167_15

Patrick Cousot and Radhia Cousot. 1977. Abstract Interpretation: A Unified Lattice Model for Static Analysis of Programs by Construction or Approximation of Fixpoints. In *Proceedings of the 4th ACM SIGACT-SIGPLAN Symposium on Principles of Programming Languages* (Los Angeles, California) *(POPL '77)*. ACM, New York, NY, USA, 238–252. https://doi.org/10.1145/512950.512973

Patrick Cousot and Radhia Cousot. 1979. Systematic Design of Program Analysis Frameworks. In *Proceedings of the 6th ACM SIGACT-SIGPLAN Symposium on Principles of Programming Languages* (San Antonio, Texas) *(POPL '79)*. ACM, New York, NY, USA, 269–282. https://doi.org/10.1145/567752.567778

Leonardo de Moura and Nikolaj Bjørner. 2008. Z3: An Efficient SMT Solver. In *Tools and Algorithms for the Construction and Analysis of Systems, 14th International Conference, TACAS 2008, Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2008, Budapest, Hungary, March 29-April 6, 2008. Proceedings (Lecture Notes in Computer Science, Vol. 4963)*, C. R. Ramakrishnan and Jakob Rehof (Eds.). Springer, 337–340. https://doi.org/10.1007/978-3-540-78800-3_24

Yotam M. Y. Feldman, James R. Wilcox, Sharon Shoham, and Mooly Sagiv. 2019. Inferring Inductive Invariants from Phase Structures. In *Computer Aided Verification - 31st International Conference, CAV 2019, New York City, NY, USA, July 15-18, 2019, Proceedings, Part II (Lecture Notes in Computer Science, Vol. 11562)*, Isil Dillig and Serdar Tasiran (Eds.). Springer, 405–425. https://doi.org/10.1007/978-3-030-25543-5_23

Cormac Flanagan, Rajeev Joshi, and K. Rustan M. Leino. 2001. Annotation inference for modular checkers. *Inf. Process. Lett.* 77, 2-4 (2001), 97–108. https://doi.org/10.1016/S0020-0190(00)00196-4

Cormac Flanagan and K. Rustan M. Leino. 2001. Houdini, an Annotation Assistant for ESC/Java. In *FME 2001: Formal Methods for Increasing Software Productivity, International Symposium of Formal Methods Europe, Berlin, Germany, March 12-16, 2001, Proceedings (Lecture Notes in Computer Science, Vol. 2021)*, José Nuno Oliveira and Pamela Zave (Eds.). Springer, 500–517. https://doi.org/10.1007/3-540-45251-6_29

Pranav Garg, Christof Löding, P. Madhusudan, and Daniel Neider. 2014. ICE: A Robust Framework for Learning Invariants. In *Computer Aided Verification - 26th International Conference, CAV 2014, Held as Part of the Vienna Summer of Logic, VSL 2014, Vienna, Austria, July 18-22, 2014. Proceedings (Lecture Notes in Computer Science, Vol. 8559)*, Armin Biere and Roderick Bloem (Eds.). Springer, 69–87. https://doi.org/10.1007/978-3-319-08867-9_5

Patrice Godefroid, Aditya V. Nori, Sriram K. Rajamani, and SaiDeep Tetali. 2010. Compositional may-must program analysis: unleashing the power of alternation. In *Proceedings of the 37th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL 2010, Madrid, Spain, January 17-23, 2010*, Manuel V. Hermenegildo and Jens Palsberg (Eds.). ACM, 43–56. https://doi.org/10.1145/1706299.1706307

Aman Goel and Karem A. Sakallah. 2021a. On Symmetry and Quantification: A New Approach to Verify Distributed Protocols. In *NASA Formal Methods - 13th International Symposium, NFM 2021, Virtual Event, May 24-28, 2021, Proceedings (Lecture Notes in Computer Science, Vol. 12673)*, Aaron Dutle, Mariano M. Moscato, Laura Titolo, César A. Muñoz, and

Ivan Perez (Eds.). Springer, 131–150. https://doi.org/10.1007/978-3-030-76384-8_9

Aman Goel and Karem A. Sakallah. 2021b. Towards an Automatic Proof of Lamport's Paxos. In *Proceedings of the 21st Conference on Formal Methods in Computer-Aided Design, FMCAD 2021*, Vol. 2. TU Wien Academic Press, 112–122. https://doi.org/10.34727/2021/isbn.978-3-85448-046-4_20

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 2015, Monterey, CA, USA, October 4-7, 2015*, Ethan L. Miller and Steven Hand (Eds.). ACM, 1–17. https://doi.org/10.1145/2815400.2815428

Matthias Heizmann, Jochen Hoenicke, and Andreas Podelski. 2009. Refinement of Trace Abstraction. In *Static Analysis, 16th International Symposium, SAS 2009, Los Angeles, CA, USA, August 9-11, 2009. Proceedings (Lecture Notes in Computer Science, Vol. 5673)*, Jens Palsberg and Zhendong Su (Eds.). Springer, 69–85. https://doi.org/10.1007/978-3-642-03237-0_7

Qinheping Hu, Jason Breck, John Cyphert, Loris D'Antoni, and Thomas W. Reps. 2019. Proving Unrealizability for Syntax-Guided Synthesis. In *Computer Aided Verification - 31st International Conference, CAV 2019, New York City, NY, USA, July 15-18, 2019, Proceedings, Part I (Lecture Notes in Computer Science, Vol. 11561)*, Isil Dillig and Serdar Tasiran (Eds.). Springer, 335–352. https://doi.org/10.1007/978-3-030-25540-4_18

Ranjit Jhala and Kenneth L. McMillan. 2006. A Practical and Complete Approach to Predicate Refinement. In *Tools and Algorithms for the Construction and Analysis of Systems, 12th International Conference, TACAS 2006 Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2006, Vienna, Austria, March 25 - April 2, 2006, Proceedings (Lecture Notes in Computer Science, Vol. 3920)*, Holger Hermanns and Jens Palsberg (Eds.). Springer, 459–473. https://doi.org/10.1007/11691372_33

Aleksandr Karbyshev, Nikolaj Bjørner, Shachar Itzhaky, Noam Rinetzky, and Sharon Shoham. 2017. Property-Directed Inference of Universal Invariants or Proving Their Absence. *J. ACM* 64, 1 (2017), 7:1–7:33. https://doi.org/10.1145/3022187

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. https://doi.org/10.1145/3385412.3386018

Leslie Lamport. 2001. Paxos made simple. *ACM SIGACT News (Distributed Computing Column)* 32, 4 (December 2001), 51–58. https://doi.org/10.1145/568425.568433

F William Lawvere. 1969. Adjointness in foundations. *Dialectica* 23, 3-4 (1969), 281–296. http://www.tac.mta.ca/tac/reprints/articles/16/tr16.pdf Republished in Reprints in Theory Appl. Categ.

Haojun Ma, Aman Goel, Jean-Baptiste Jeannin, Manos Kapritsos, Baris Kasikci, and Karem A. Sakallah. 2019. I4: incremental inference of inductive invariants for verification of distributed protocols. In *Proceedings of the 27th ACM Symposium on Operating Systems Principles, SOSP 2019, Huntsville, ON, Canada, October 27-30, 2019*. 370–384. https://doi.org/10.1145/3341301.3359651

Dahlia Malkhi, Leslie Lamport, and Lidong Zhou. 2008. *Stoppable Paxos.* Technical Report MSR-TR-2008-192. https://www.microsoft.com/en-us/research/publication/stoppable-paxos/

Zohar Manna and Amir Pnueli. 1995. *Temporal Verification of Reactive Systems - Safety.* Springer. https://doi.org/10.1007/978-1-4612-4222-2

Kenneth L. McMillan. 2003. Interpolation and SAT-Based Model Checking. In *Computer Aided Verification, 15th International Conference, CAV 2003, Boulder, CO, USA, July 8-12, 2003, Proceedings (Lecture Notes in Computer Science, Vol. 2725)*, Warren A. Hunt Jr. and Fabio Somenzi (Eds.). Springer, 1–13. https://doi.org/10.1007/978-3-540-45069-6_1

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 (Lecture Notes in Computer Science, Vol. 4144)*, Thomas Ball and Robert B. Jones (Eds.). Springer, 123–136. https://doi.org/10.1007/11817963_14

Kenneth L. McMillan. 2014. Lazy Annotation Revisited. In *Computer Aided Verification - 26th International Conference, CAV 2014, Held as Part of the Vienna Summer of Logic, VSL 2014, Vienna, Austria, July 18-22, 2014. Proceedings*. 243–259. https://doi.org/10.1007/978-3-319-08867-9_16

Anders Miltner, Saswat Padhi, Todd D. Millstein, and David Walker. 2020. Data-driven inference of representation invariants. In *Proceedings of the 41st ACM SIGPLAN International Conference on Programming Language Design and Implementation, PLDI 2020, London, UK, June 15-20, 2020*. 1–15. https://doi.org/10.1145/3385412.3385967

Daniel Neider, Shambwaditya Saha, Pranav Garg, and P. Madhusudan. 2019. Sorcar: Property-Driven Algorithms for Learning Conjunctive Invariants. In *Static Analysis - 26th International Symposium, SAS 2019, Porto, Portugal, October 8-11, 2019, Proceedings (Lecture Notes in Computer Science, Vol. 11822)*, Bor-Yuh Evan Chang (Ed.). Springer, 323–346. https://doi.org/10.1007/978-3-030-32304-2_16

Oded Padon, Giuliano Losa, Mooly Sagiv, and Sharon Shoham. 2017. Paxos Made EPR: Decidable Reasoning About Distributed Protocols. *Proc. ACM Program. Lang.* 1, OOPSLA, Article 108 (Oct. 2017), 31 pages. https://doi.org/10.1145/3140568

Oded Padon, Kenneth L. McMillan, Aurojit Panda, Mooly Sagiv, and Sharon Shoham. 2016. Ivy: safety verification by interactive generalization. In *Proceedings of the 37th ACM SIGPLAN Conference on Programming Language Design and Implementation, PLDI 2016, Santa Barbara, CA, USA, June 13-17, 2016*, Chandra Krintz and Emery Berger (Eds.). ACM, 614–630. https://doi.org/10.1145/2908080.2908118

Oded Padon, James R. Wilcox, Jason Koenig, Kenneth L. McMillan, and Alex Aiken. 2021. Artifact for POPL 2022 Paper: Induction Duality: Primal-Dual Search for Invariants. (November 2021). https://doi.org/10.5281/zenodo.5703081

Peter Smith. 2010. *The Galois connection of syntax and semantics.* Technical Report. Cambridge University. http://www.logicmatters.net/resources/pdfs/Galois.pdf

Fabio Somenzi and Aaron R. Bradley. 2011. IC3: where monolithic and incremental meet. In *International Conference on Formal Methods in Computer-Aided Design, FMCAD '11, Austin, TX, USA, October 30 - November 02, 2011*, Per Bjesse and Anna Slobodová (Eds.). FMCAD Inc., 3–8. http://dl.acm.org/citation.cfm?id=2157657

Marcelo Taube, Giuliano Losa, Kenneth L. McMillan, Oded Padon, Mooly Sagiv, Sharon Shoham, James R. Wilcox, and Doug Woos. 2018. 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. https://doi.org/10.1145/3192366.3192414

James R. Wilcox, Doug Woos, Pavel Panchekha, Zachary Tatlock, Xi Wang, Michael D. Ernst, and Thomas E. Anderson. 2015. Verdi: a framework for implementing and formally verifying distributed systems. In *Proceedings of the 36th ACM SIGPLAN Conference on Programming Language Design and Implementation, Portland, OR, USA, June 15-17, 2015*, David Grove and Stephen M. Blackburn (Eds.). ACM, 357–368. https://doi.org/10.1145/2737924.2737958

Jianan Yao, Runzhou Tao, Ronghui Gu, Jason Nieh, Suman Jana, and Gabriel Ryan. 2021. DistAI: Data-Driven Automated Invariant Learning for Distributed Protocols. In *15th USENIX Symposium on Operating Systems Design and Implementation, OSDI 2021, July 14-16, 2021*, Angela Demke Brown and Jay R. Lorch (Eds.). USENIX Association, 405–421. https://www.usenix.org/conference/osdi21/presentation/yao