SAT-Based Invariant Inference and Its Relation to Concept Learning

Yotam M. Y. Feldman and Sharon Shoham

Tel Aviv University, Israel

Abstract. This paper surveys results that establish formal connections and distinctions between SAT-based invariant inference and exact concept learning with queries, showing that learning techniques and algorithms can clarify foundational questions, illuminate existing algorithms, and suggest new directions for efficient invariant inference.

1 Introduction

SAT-based invariant inference algorithms such as IC3/PDR [8, 4] and Interpolation [25] have proven to be extremely successful in practice and have attracted tremendous interest in recent years. However, the essence of their practical success and their performance guarantees are far less understood. In a series of papers [10, 11, 12, 13] we set out to investigate these topics and provide new insights into the principles and complexity of SAT-based invariant inference. This paper surveys one of the key avenues pursued in these works, which focuses on the similarities and discrepancies between SAT-based invariant inference and exact concept learning from queries for propositional formulas, both as a way to explain and analyze existing inference algorithms, and as way to develop new algorithms.

Exact learning with queries [2] is one of the fundamental fields of theoretical machine learning. There, a *learner* (an algorithm) needs to learn an unknown concept, e.g., a formula from some class of formulas, with the help of a *teacher* who can answer certain queries about the concept. Typical queries include membership queries: “is a certain example a member of the desired concept?”, and equivalence queries: “is a certain candidate the desired concept?”. The theory of exact concept learning is well developed and provides ample efficient algorithms for learning different classes of concepts with different kinds of queries.

The goal of SAT-based invariant inference is also to learn a formula—an inductive invariant. Further, the way an inference algorithm uses a SAT solver to check inductiveness and bounded reachability w.r.t. the transition relation in the process of constructing candidate inductive invariants bears strong resemblance to how a learning algorithm uses the teacher to check equivalence or membership in concept learning.

The first step towards understanding invariant inference from the perspective of learning is to distill this connection and study it in a rigorous way that enables a transfer of ideas between the fields. To this end, we introduce a model of *invariant inference with queries* [10]. In this model, the transition relation of the system is only known to an oracle (implemented by a SAT solver), and an inference algorithm can only "query" it by posing queries to the oracle. Through a sequence of queries, the algorithm should gain enough information about the transition relation to be able to find an appropriate invariant. We consider queries that are common in existing invariant inference algorithms: *inductiveness queries*, where the solver is given a candidate invariant α and checks if it is inductive, and their generalization into *Hoare queries*, where the solver is given a precondition formula α, a postcondition formula β and a bound k, and checks if (some state in) β is reachable from (some state in) α in at most k steps (inductiveness queries correspond to Hoare queries with $\alpha = \beta$ and $k = 1$). Hoare queries naturally capture how many invariant inference algorithms use a SAT solver, including major versions of PDR and Interpolation, and so results about the Hoare-query model apply to these algorithms.

A previous learning-based model for invariant inference, ICE learning [15], corresponds to algorithms that use inductiveness queries only. In practice, many algorithms (that historically precede the ICE learning model) use more general Hoare queries to facilitate an incremental construction of invariants in complex syntactic forms. For example, PDR [4, 8] incrementally learns clauses in different frames via relative inductiveness checks, and Interpolation learns at each iteration a term of the invariant from an interpolant [25]. We show that this is in fact a significant difference: the Hoare-query model is strictly stronger than inference based solely on presenting whole candidate inductive invariants as in the inductiveness-query model. To this end, in [10], we identify a class of systems where a Hoare query algorithm, which is essentially a simplified version of PDR (and a dual version of Interpolation), can efficiently infer invariants, whereas every inference algorithm in the inductiveness query model requires an exponential number of queries in the worst case. This confirms the intuition from [39] that PDR cannot be implemented within the ICE model.

Having laid the foundations, we set out to compare invariant inference with queries to exact concept learning. We prove that neither membership queries nor equivalence queries to an unknown invariant can be implemented by Hoare queries in general [10]. In particular, even though inductiveness queries can determine if a formula is an inductive invariant, they are still unable to simulate equivalence queries since they can only return a *counterexample to induction*—a *pair* of states such that *if* the first state is part of the invariant *then* so should be the second. The non-implementability result implies that neither inductiveness nor Hoare queries are sufficient for identifying a (positive or negative) example that definitively differentiates the formula from an inductive invariant. This provides a formal justification to the introduction of *implication examples* in the ICE model [15] for learning from examples, as an addition to positive and negative examples.

The inability to implement exact learning queries is unfortunate, as it prevents porting the rich literature of exact learning algorithms and theory to invariant inference. However, we identify a condition, called the *fence condition* [11], that rectifies the situation and makes it possible to simulate certain kinds of exact learning queries in the Hoare-query model. The fence condition requires that the states in the *boundary* of the invariant—states outside of the invariant with a Hamming distance of 1 from states inside the invariant—can reach a bad state in a bounded number of steps. We show in [11] that when the membership and equivalence queries performed by an exact learning algorithm satisfy certain restrictions, it is possible to translate the learning algorithm into an invariant inference algorithm in the Hoare-query model that is always sound (i.e., never returns an incorrect inductive invariant), and enjoys the same complexity if the fence condition holds.

These translations are not just theoretical. In [11] we show that a model-based version due to Chockler et al. and Bjørner et al. [7, 3] of McMillan's Interpolation algorithm [25] can be obtained by such a translation from an exact learning algorithm for learning DNF formulas [1], which is efficient for monotone formulas [1, 2]. Not only is it fascinating that an inference algorithm turns out to be an incarnation of an earlier algorithm from a different discipline, but the translation also gives rise to a new efficiency result for the interpolation-based algorithm for monotone invariants when the fence condition holds. To the best of our knowledge this is the first result of its kind. The translation is also applicable to an exact learning algorithm for almost-monotone invariants (and its complexity) [5], which leads to the introduction of a new invariant inference algorithm with provable polynomial complexity guarantees for almost-monotone invariants when the fence condition holds.

The aforementioned simulation of exact learning algorithms is only possible when the queries are restricted in a certain way. Some algorithms, such as Bshouty's algorithm [5] for learning CDNF formulas—formulas that have a short CNF representation as well as a short DNF representation—do not meet these restrictions. Nonetheless, we present in [11] another translation that is applicable to *any* exact learning algorithm from membership and equivalence queries, and maintains the complexity of the algorithm if a stronger, *two-sided* fence condition holds.

The question whether it is possible to simulate Bshouty's CDNF algorithm under a (one-sided) fence condition remains open. However, inspired by the CDNF algorithm, and utilizing insights about properties of the boundary of an invariant, in [13], we develop a novel invariant inference algorithm from Hoare queries that is efficient for CDNF invariants under the assumption of a (one-sided) fence condition. Interestingly, this algorithm cannot be viewed as a concept learning algorithm, hinting that invariant inference can not only benefit from exact learning, but can also exceed it.

Not included in this survey is our investigation of PDR in [12] using the monotone theory [5] developed for concept learning. This work does not investigate PDR as a concept learning algorithm, but relates it to key principles used in learning monotone and almost-monotone invariants.

Outline. The rest of the paper is organized as follows. After a brief background in §2, we define the model of invariant inference from queries and show an exponential gap between inductiveness queries and Hoare queries in §3 (based on [10]). We contrast invariant inference with exact concept learning from queries in §4 (based on [10]), and present cases where the gap can be bridged through the fence condition in §5 (based on [11, 13]). We conclude in §6.

2 Preliminaries

Transition systems. We consider transition systems defined using propositional logic. Given a propositional vocabulary Σ, a *state* is a valuation to Σ. We denote by $\mathcal{F}(\Sigma)$ the set of well-formed formulas over Σ. A *transition system* is a triple $TS = (Init, \delta, Bad)$ such that $Init, Bad \in \mathcal{F}(\Sigma)$ define the *initial states* and the *bad states*, respectively, and $\delta \in \mathcal{F}(\Sigma \uplus \Sigma')$ defines the *transition relation*, where $\Sigma' = \{x' \mid x \in \Sigma\}$ is a copy of the vocabulary used to describe the post-state of a transition. Given a $\varphi \in \mathcal{F}(\Sigma)$, we denote by φ' the formula obtained from φ by replacing each variable with its counterpart in Σ'.

Safety and inductive invariants. A transition system TS is *safe* if all the states that are reachable from the initial states via steps of δ satisfy $\neg Bad$. An *inductive invariant* for TS is a formula $I \in \mathcal{F}(\Sigma)$ such that (i) $Init \Longrightarrow I$, (ii) $I \wedge \delta \Longrightarrow I'$, and (iii) $I \Longrightarrow \neg Bad$ (where $\Longrightarrow$ denotes validity of implication). A transition system is safe if and only if it has an inductive invariant. When I is not inductive, a *counterexample to induction (cti)* is a pair of states σ, σ' such that $\sigma, \sigma' \models I \wedge \delta \wedge \neg I'$ (where the valuation to Σ' is taken from σ').

Notation. We use formulas and the sets of states that they represent interchangeably. For a state σ, we denote by $cube(\sigma)$ the conjunction of all literals (variables or their negations) that hold in σ.

3 Invariant Inference with Queries

An investigation of SAT-based invariant inference and its relation to concept learning was initiated in [10], by identifying the common SAT queries carried out by existing algorithms, and introducing corresponding query models. A query-based approach allows to compare different invariant inference algorithms both to each other and to concept-learning algorithms that use queries.

In this section we define the invariant inference problem, the basic notions of queries and query-based inference algorithms, and the query models considered in this survey: Inductiveness and Hoare, which capture existing SAT-based invariant inference algorithms.

Invariant inference can be formulated as follows.

Definition 1 (Inductive Invariant Inference from Class of Invariants). *For a class of transition systems $\mathcal{P}$ and a class of invariants $\mathcal{L}$,* inductive invariant inference *is the problem: Given a transition system $TS \in \mathcal{P}$ over Σ,* find *an inductive invariant $I \in \mathcal{L}$ for TS or determine that none exists.*

When $\mathcal{L}$ is omitted, we mean that, for every Σ, it includes all formulas in $\mathcal{F}(\Sigma)$.

3.1 Inference with Queries

In the setting of invariant inference with queries, an algorithm accesses the transition relation through queries—corresponding to SAT queries performed by existing algorithms—but cannot read the transition relation directly. This *black-box* model reflects the way typical SAT-based invariant inference algorithms use the transition relation only in their SAT queries, as opposed to *white-box* algorithms that analyze the code directly. A black-box model of inference algorithms facilitates an analysis of the *information* of the transition relation the algorithm acquires. The advantage is that such an information-based analysis sidesteps open computational complexity questions, and therefore results in unconditional lower bounds on the complexity of SAT-based algorithms captured by the model.

Queries of the transition relation are modeled in the following way. A *query oracle* Q is an oracle that accepts a transition relation δ, as well as additional inputs, and returns some output. The additional inputs and the output, together also called the *interface* of the oracle, depend on the query oracle under consideration. A *family* of query oracles is a set of query oracles with the same interface.

Definition 2 (Inference algorithm in the query model). *An* inference algorithm from queries*, denoted* $\mathcal{A}^{Q}(\mathit{Init}, \mathit{Bad}, [\delta])$*, is an algorithm defined w.r.t. a query oracle* Q *that solves the invariant inference problem for* $(\mathit{Init}, \delta, \mathit{Bad})$*, given:*

- *access to the query oracle* Q*,*
- *the set of initial states (Init) and bad states (Bad);*
- *the transition relation* δ*, encapsulated—hence the notation* $[\delta]$*—meaning that the algorithm cannot access* δ *(not even read it) except for extracting its vocabulary;* δ *can only be passed as an argument to the query oracle* Q*.*

In the sequel, we consider two different families of query oracles: inductiveness and Hoare, representing different ways of obtaining information about the transition relation.

Time and Query Complexity. Much like a SAT solver, the query oracles solve NP-complete problems. When analyzing the complexity we consider each query as a single step, and count the number of queries and also the time of other steps the algorithm performs. (In lower bounds, we often only report on the query complexity, which in itself provides a lower bound on the time complexity.) We analyze the complexity in a *worst-case* model w.r.t. the possible transition systems in the class of interest as well as w.r.t. the possible query oracles in the family (the worst-case analysis is motivated by the property that in SAT-based algorithms, the oracle is implemented by a SAT solver, which the algorithm does not control). For a class of transition systems $\mathcal{P}$, the (time or query) complexity of $\mathcal{A}$ w.r.t. a query oracle family $\mathcal{Q}$ is defined as

$$\sup_{Q \in \mathcal{Q}} \sup_{\substack{(Init, \delta, Bad) \in \mathcal{P}, \\ |\Sigma| = n}} \Phi(\mathcal{A}^{Q}(Init, [\delta], Bad))$$

where $\Phi(\mathcal{A}^{Q}(Init, [\delta], Bad))$ measures the complexity (either number of steps or number of queries) of $\mathcal{A}$ given oracle $Q \in \mathcal{Q}$ and input $(Init, \delta, Bad) \in \mathcal{P}$. (These numbers might be infinite.)

3.2 The Inductiveness-Query Model

The first query model we consider only allows an algorithm to check inductiveness of a candidate invariant:

Definition 3 (Inductiveness-Query Model). *An* inductiveness-query oracle *is a query oracle $\mathcal{I}$ such that for every δ and $\alpha \in \mathcal{F}(\Sigma)$ satisfying $Init \Longrightarrow \alpha$ and $\alpha \Longrightarrow \neg Bad$,*

- *$\mathcal{I}(\delta, \alpha) = true$ if $\alpha \wedge \delta \Longrightarrow \alpha'$, and*
- *$\mathcal{I}(\delta, \alpha) = (\sigma, \sigma')$ such that $(\sigma, \sigma') \models \alpha \wedge \delta \wedge \neg\alpha'$ otherwise.*

An algorithm in the inductiveness-query model*, also called an* inductiveness-query algorithm*, is an inference from queries algorithm expecting any inductiveness query oracle.*

Inductiveness-query oracles form a *family* of oracles since different oracles can choose different (σ, σ') for each δ, α.

ICE learning and inductiveness-queries The inductiveness-query model is closely related to ICE learning [15], except here the learner is provided with full information on $Init, Bad$ instead of positive and negative examples (and the algorithm refrains from querying on candidates that do not include $Init$ or do not exclude Bad). This model captures several interesting algorithms, including include Houdini [14] and symbolic abstraction [30, 35], as well as designated algorithms [15, 16]. Our complexity definition in the inductiveness-query model being the worst-case among all possible oracle responses is in line with the analysis of strong convergence in Garg et al. [15]. Hence, lower bounds on the query complexity in the inductiveness query model imply lower bounds for the strong convergence of ICE learning.

3.3 The Hoare-Query Model

The Hoare-query model captures SAT-based invariant inference algorithms querying the reachability of one set of states from a possibly different set of states through a sequence of at most k-steps of the transition relation, for a fixed k.

Definition 4 (Hoare-Query Model). *A* Hoare-query oracle *is a query oracle $\mathcal{H}$ such that for every δ, $\alpha, \beta \in \mathcal{F}(\Sigma)$, and k,*

- $\mathcal{H}^{(k)}(\delta, \alpha, \beta) = \mathit{true}$ *if* $\alpha(\Sigma^0) \wedge \delta(\Sigma^0, \Sigma^1) \wedge \ldots \wedge \delta(\Sigma^{k-1}, \Sigma^k) \implies \bigwedge_{i=1}^{k} \beta(\Sigma^i)$, *where* $\Sigma^0, \ldots, \Sigma^k$ *are* $k+1$ *distinct copies of the vocabulary, and*
- $\mathcal{H}^{(k)}(\delta, \alpha, \beta) = (\sigma_0, \ldots, \sigma_k)$ *such that* $\sigma_0, \ldots, \sigma_k \models \alpha(\Sigma^0) \wedge \delta(\Sigma^0, \Sigma^1) \wedge \ldots \wedge \delta(\Sigma^{k-1}, \Sigma^k) \wedge \bigvee_{i=1}^{k} \neg\beta(\Sigma^k)$*, otherwise.*

An algorithm in the Hoare-query model*, also called a* Hoare-query algorithm*, is an inference from queries algorithm expecting any Hoare-query oracle, where k is bounded by a polynomial in n in all queries.*

Hoare-query oracles form a *family* of oracles since different oracles can choose different counterexample traces $(\sigma_0, \ldots, \sigma_k)$ for every δ, α, β, k.

Example: PDR as a Hoare-query algorithm The Hoare-query model captures the prominent PDR algorithm, facilitating its theoretical analysis. In general, PDR maintains a sequence of frames $F_0, F_1, \ldots$ such that $F_0 = \mathit{Init}$, $F_i \implies F_{i+1}$, $F_i \wedge \delta \implies F'_{i+1}$ and $F_i \implies \neg \mathit{Bad}$ (for every i). These properties ensure that if at some point $F_{i+1} \implies F_i$ then F_i is an inductive invariant. To update the frames, PDR accesses the transition relation via checks of unreachability in one step and counterexamples to those checks. These operations are captured in the Hoare query model by checking $\mathcal{H}^{(1)}(\delta, F, \alpha)$ or $\mathcal{H}^{(1)}(\delta, F \wedge \alpha, \alpha)$. This is illustrated using Alg. 1 which roughly corresponds to PDR with just one frame. The only accesses to δ are in lines 5, 6 and 9, which are all done through the Hoare-query oracle, showing that Alg. 1 is a Hoare-query algorithm. The (basic) full PDR can similarly be modeled as a Hoare-query algorithm [10]. Furthermore, the Hoare-query model is general enough to express a broad range of PDR variants that differ in the way they use such checks but still access the transition relation only through such queries.[1]

Example: Interpolation-based inference as a Hoare-query algorithm Another operation supported by SAT solvers is *interpolation*. Interpolation has been introduced to invariant inference by McMillan [25], and extended in many works since [e.g. 37, 26, 22, 18, 38]. Interpolation algorithms infer invariants from facts obtained from bounded unreachability of the bad states, checked by Hoare queries of the form $\mathcal{H}^{(k)}(\delta, F, \neg \mathit{Bad})$. In McMillan's original paper these facts are *interpolants* extracted from a resolution proof computed by the solver. As such, to account for McMillan's original interpolation-based inference algorithm [25], the oracle also needs to return an interpolant when the Hoare query checking unreachability of *Bad* returns *true*. This model was investigated in [10]. Our focus here is on model-based interpolation [7, 3], as displayed in Alg. 2, for which such an extension is not necessary—model-based interpolation computes interpolants

[1] A notable exception is ternary simulation [8], which is not a SAT-based operation. However, the query model can be extended to support it while maintaining our results.

as part of the inference procedure (from bounded unreachability and counterexamples), rather than inside the solver (from the proof of bounded unreachability itself). Alg. 2 is a Hoare-query algorithm: the only accesses to δ are in lines 3, 4 and 9, and all invoke the Hoare-query oracle.

3.4 Hoare-Queries vs. Inductiveness-Queries

Inductiveness queries are specific instances of Hoare queries, where the precondition and postcondition are the same, and reachability is examined along a single step of the transition relation ($k = 1$). Therefore, inductiveness-query algorithms can be simulated by Hoare-query algorithms. This raises the question whether the seemingly more general Hoare queries are indeed so. In this section we answer this question affirmatively and show that the Hoare query model (Def. 4) is *strictly stronger* than the inductiveness query model (Def. 3), even when $k = 1$. To this end we show that there exists a class of transition systems for which a simple Hoare-query algorithm can infer invariants in polynomial time, but every inductiveness-query algorithm requires an exponential number of queries.

The exponential gap between the Hoare-query model and the inductiveness query model is summarized by the following theorem:

Theorem 1 ([10]). *There exists a class of transition systems $\mathcal{M}_E$ for which*

- *invariant inference has* polynomial *time complexity in the Hoare-query model, but*
- *every inference algorithm in the inductiveness-query model requires an* exponential *query complexity.*

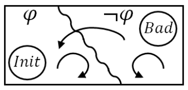
The class $\mathcal{M}_E$ consists of *maximal systems* for *monotone* CNF invariants together with certain unsafe systems. [2] We refer the reader to [10] for the precise definition of $\mathcal{M}_E$ and to the proof of the lower bound in the inductiveness-query model. Here we only highlight two properties of the safe systems in $\mathcal{M}_E$ that facilitate efficient inference in the Hoare-query model: *maximality* of the system, defined below, and existence of a polynomial *monotone* invariant. Beyond establishing the upper bound in the Hoare-query model, these properties also spur the research on efficient inference discussed in subsequent sections.

Definition 5 (Monotone Invariants). *We denote by* Mon-CNF$_{p(n)}$ *the class of CNF formulas where variables appear only positively and where, for a vocabulary Σ with $n = |\Sigma|$, the number of clauses in formulas over Σ is bounded by $p(n)$, for a fixed polynomial $p(\cdot)$.*

Definition 6 (Maximal System). *Let $Init, Bad \not\equiv false$ and let φ be a formula such that $Init \Longrightarrow \varphi$ and $\varphi \Longrightarrow \neg Bad$. The* maximal transition system *for φ is $(Init, \delta^{\mathcal{M}}_{\varphi}, Bad)$ where $\delta^{\mathcal{M}}_{\varphi} = \varphi \to \varphi'$.*

[2] In [10], the invariants are *antimonotone* rather than monotone; the algorithm establishing the upper bound is efficient also for monotone invariants, and the proof of the lower bound can also be adapted to monotone invariants.

A maximal transition system is illustrated as follows:



Note that $\delta_\varphi^\mathcal{M}$ goes from any state satisfying φ to any state satisfying φ, and from any state satisfying $\neg\varphi$ to all states, good or bad. $\delta_\varphi^\mathcal{M}$ is *maximal* in the sense that it allows all transitions that do not violate φ being an inductive invariant.

In $\mathcal{M}_E$ we consider maximal systems for every formula in Mon-CNF$_{p(n)}$, together with an unsafe transition system whose transition relation is *true* for each vocabulary. In particular, this means that every safe transition system in $\mathcal{M}_E$ has an inductive invariant in Mon-CNF$_{p(n)}$ (and others have no inductive invariant). Therefore, solving invariant inference for $\mathcal{M}_E$ without restricting the class of invariants coincides with restricting it to $\mathcal{L}$ = Mon-CNF$_{p(n)}$; this is important in §4.1 when comparing the complexity of invariant inference to exact concept learning.

Upper bound for Hoare-query algorithms for maximal systems w.r.t. monotone invariants. The upper bound is obtained by a simple algorithm, called PDR-1, that can find inductive invariants for safe systems in $\mathcal{M}_E$ with a polynomial number of Hoare queries. PDR-1, depicted in Alg. 1, is a backward-reachability algorithm, operating by repeatedly checking for the existence of a counterexample to induction, and obtaining one when it exists. The invariant is then strengthened by conjoining the candidate invariant with the negation of a subset of the cube of the pre-state: starting with *cube*(σ), which is a conjunction of literals that holds only on σ, a subset of the literals leaves a smaller conjunction, which represents a larger set of states, thereby "generalizing" σ. Generalization is performed by dropping a literal from the cube whenever the remaining conjunction does not hold for any state reachable in at most one step from *Init*. The result is a minimal conjunction whose negation does not exclude any state reachable in at most one step. This might exclude reachable states in general transition systems, but not in maximal systems, since maximality ensures that their diameter is one.

Algorithm 1 PDR-1 in the Hoare-query model

```
1: procedure PDR-1($Init$, $[\delta]$, $Bad$)
2:     $I \leftarrow \neg Bad$
3:     if $\mathcal{H}^{(1)}(\delta, Init, \neg Bad) \neq true$ then
4:         unsafe
5:     while $\mathcal{H}^{(1)}(\delta, I, I) \neq true$ do                  // $I$ not inductive
6:         $(\sigma, \sigma') \leftarrow \mathcal{H}^{(1)}(\delta, I, I)$            // counterexample to induction of $I$
7:         $d \leftarrow cube(\sigma)$
8:         for $l \in cube(\sigma)$ do
9:             $t \leftarrow d \setminus \{l\}$
10:            if $Init \Longrightarrow \neg t$ and $\mathcal{H}^{(1)}(\delta, Init, \neg t)$ then     // $Init \Longrightarrow \neg t \wedge Init \wedge \delta \Longrightarrow \neg t'$
11:                $d \leftarrow t$
12:        $I \leftarrow I \wedge \neg d$
13:    return $I$
```

Maximality therefore allows to determine if a state needs to be part of the invariant by a simple Hoare-query, examining reachability in 1-step, and ensures that generalization returns a prime consequence of the invariant (a clause implied by the invariant which is not strictly weaker than any other clause implied by the invariant). Efficiency of the algorithm results from the monotonicity of the CNF invariants, which lets PDR-1 efficiently reconstruct them as the conjunction of their prime consequences, via a theorem that goes back to Quine [29].

However, our lower bound for the inductinvess-query model for the same class of transition systems and invariants shows that this incremental process inherently relies on rich Hoare queries.

3.5 PDR and Interpolation-Based Inference Cannot Be Implemented with Inductiveness Queries

PDR-1, the Hoare-query algorithm we use to establish the exponential gap, is essentially PDR with a single frame[3]. Hence, building on the proof of Thm. 1, which shows that no inductiveness-query algorithm can simulate PDR-1 on the class $\mathcal{M}_E$, we conclude that PDR cannot be efficiently simulated in the inductiveness-query model:

Theorem 2. *There is no inductiveness-query algorithm that solves invariant inference with a number of inductiveness queries that has at most polynomial overhead on the number of Hoare queries performed by PDR.*

A similar result applies to interpolation-based inference: the exponential gap between the inductiveness and Hoare query models can also be established for maximal systems for Mon-DNF$_{p(n)}$ invariants (defined similarly to Mon-CNF$_{p(n)}$, as the class of DNF formulas where variables appear only positively and where the number of terms is bounded by $p(n)$), in which case the upper bound is obtained by an algorithm dual to PDR-1, the model-based interpolation-based algorithm displayed in Alg. 2 with a reachability bound of $k = 1$. This shows that Hoare-queries are inherent to both PDR and interpolation-based inference in the sense that neither can be implemented with inductiveness queries only, confirming the intuition from [39] regarding PDR. (Profound differences between PDR and interpolation manifest when PDR uses more than one frame and interpolation uses $k > 1$, a topic we explored in [12].)

4 Invariant Learning & Concept Learning with Queries

Query-based models of invariant inference highlight its similarity to exact concept learning with queries. What are the connections and differences between

[3] To be precise, in PDR, counterexamples are states that reach a bad state, whereas PDR-1 uses counterexamples to induction, but these coincide in maximal systems; additionally, PDR may use an additional frame to discover the counterexamples and one more to detect convergence.

concept-learning *formulas* in $\mathcal{L}$ and learning *invariants* in $\mathcal{L}$? Can concept learning algorithms be translated to inference algorithms? These questions have spurred much research [e.g. 15, 27, 9, 20, 16, 33, 34, 32, 31, 23, 21]. In this section we study these questions from the perspective of our aforementioned results.

In *exact concept learning* [2], an algorithm's task is to identify an unknown formula[4] ψ using queries it poses to a *teacher*. The most studied queries are:

– *Membership*: The algorithm *chooses* a state σ, and the teacher answers whether $\sigma \models \psi$; and
– *Equivalence*: The algorithm *chooses* a candidate θ, and the teacher returns true if $\theta \equiv \psi$ or a differentiating counterexample otherwise: a σ s.t. $\sigma \not\models \theta, \sigma \models \psi$ or $\sigma \models \theta, \sigma \not\models \psi$.

In this section, we compare invariant inference to exact concept learning and show: (1) that classical queries in exact concept learning cannot be efficiently implemented as queries in order to find an unknown inductive invariant, and (2) that ICE-learning is provably harder than classical learning: namely, that, as advocated by Garg et al. [15], learning from counterexamples to induction is inherently harder than learning from examples labeled positive or negative.

4.1 Complexity Comparison

This section compares the complexity of inferring formulas to concept-learning the same class of formulas.

Thm. 1 effectively studies the complexity of inferring $\mathcal{L} = \text{Mon-CNF}_{p(n)}$ invariants using Hoare/inductiveness queries for *maximal systems*. The next theorem studies the complexity for *general systems*:

Theorem 3. *Every Hoare-query inference algorithm solving invariant inference for the class of all propositional transition systems and the class of invariants* $\mathcal{L} = \text{Mon-CNF}_{p(n)}$ *has query complexity of* $2^{\Omega(n)}$*, where* $n = |\Sigma|$.

We emphasize that the lower bound considers inference of short, polynomial, invariants, which ensures that the exponential complexity is not an artifact of the length of the invariant, but, rather, of the need to infer it. We also point out that in the more standard setting, when the algorithm is not restricted to access the transition relation only through Hoare queries, the computational complexity of inferring invariants of polynomial length is Σ_2^P-complete (NP-complete with access to a SAT solver as oracle), as shown in [10] (strengthening a similar hardness result by Lahiri and Qadeer for inferring invariants over a template [24]).

Table 1 displays these results for invariant inference with a query oracle, and compares them with known complexity results for exact concept learning. For the sake of the comparison, the table maps inductiveness queries to equivalence

[4] In general, a concept is a set of elements; here we focus on logical concepts.

Table 1: Concept vs. invariant learning: complexity of learning Mon-CNF$_{p(n)}$

Invariant Inference			Concept Learning	
	Maximal Systems	General Systems		
Inductiveness	Exponential (Thm. 1)	Exponential (Thm. 3)	Equivalence	Subexponential[1] / Polynomial[2] [17, 2]
Hoare	Polynomial (Thm. 1)	Exponential (Thm. 3)	Equivalence + Membership	Polynomial [2]

[1] proper learning
[2] with exponentially long candidates

queries (as these are similar at first sight) and maps the more powerful setting of Hoare queries to the more powerful setting of equivalence together with membership queries.

The comparison in the table demonstrates that *invariant inference in general systems is harder* than exact learning. The implications of the complexity gaps are elaborated in §4.2. The complexity gap is eliminated when considering only *maximal systems*, which is the source of the upper bound in Thm. 1. However, that is true only for the Hoare-query model, and gaps remain when considering only inductiveness queries; this is elaborated in §4.3.

4.2 Invariant Learning Cannot Be Reduced to Concept Learning

This section builds on the above complexity comparison to check which concept learning queries can be simulated and used in invariant inference.

Table 2: Concept vs. invariant learning: implementability of concept learning queries

	Maximal Systems		General Systems	
	Inductiveness	Hoare	Inductiveness	Hoare
Equivalence	✗	✓	✗	✗
Membership	✗	✓	✗	✗

Table 2 summarizes our results for the possibility and impossibility of simulating concept learning algorithms in invariant learning with queries. This table depicts implementability (✓) or unimplementability (✗) of membership and equivalence queries used in concept learning through inductiveness and Hoare queries used in learning invariants for maximal systems and for general systems.

Formally, the implementability of a (concept learning) query in a class of transition systems $\mathcal{P}$ means that for every class of invariants $\mathcal{L}$ there is an inference algorithm that, given a transition system $TS \in \mathcal{P}$ that admits some (unknown) invariant $I \in \mathcal{L}$, correctly answers the query w.r.t. I with a polynomial number of queries in the respective model.

The proofs of impossibilities are based on the differences in complexity from Table 1 for $\mathcal{L} = \text{Mon-CNF}_{p(n)}$. The only possibility result in the table is of simulating equivalence and membership queries using Hoare queries over maximal systems (for every $\mathcal{L}$); the idea is that a Hoare query $\mathcal{H}^{(1)}(\delta_{\varphi}^{\mathcal{M}}, \mathit{Init}, \neg \mathit{cube}(\sigma)) \overset{?}{\neq} \mathit{true}$ implements a membership query on σ, thanks to fact that the inductive invariant is exactly the set of states reachable in one step. Such a membership query (together with an inductiveness query) can also be used to implement an equivalence query, specifically to convert a counterexample to induction into a differentiating counterexample as required when answering an equivalence query negatively: given the counterexample to induction $(\sigma, \sigma') = \mathcal{I}(\delta_{\varphi}^{\mathcal{M}}, \theta)$, use a membership query to determine if $\sigma \not\models I$ or $\sigma' \models I$, and return σ or σ' accordingly. We pick up on these ideas for implementing membership queries in §5, with more sophisticated translations that are related to more realistic algorithms.

4.3 Counterexamples in Invariant Learning Are Inherently Ambiguous

As we have seen, equivalence queries cannot be implemented using inductiveness queries, even in the simple case of maximal systems. The reason is that when the query fails—returns "not inductive" or "not equivalent"—then the counterexample provided to the inference algorithm is inherently weaker than the counterexample for the learning algorithm. In inference, the result is a counterexample to induction (an implication example, in the terminology of Garg et al. [15]), which is a *pair* of examples (σ, σ'), where σ is a negative example *or* σ' is a positive example, but there is no indication in the query itself of which is the case. In contrast, in classical equivalence queries, the counterexample is a single state σ, and it is in effect labelled—by checking whether the proposed candidate is satisfied by σ or not the learner can tell whether σ is a positive or negative example.

This discrepancy can be reformulated in the context of concept learning, as the difference between classical learning from equivalence queries (using labeled examples) and ICE learning [15], in which (essentially) the result of an equivalence query is an implication example. We have thus obtained a complexity result separating the two:

Corollary 1. *There exists a class of formulas $\mathcal{L}$ that can be learned using a subexponential number of equivalence queries, but requires an exponential number of ICE-equivalence queries.*

This result quantitatively corroborates the difference between *counterexamples to induction* and *examples labeled positive or negative*, a distinction advocated by Garg et al. [15].

5 From Exact Learning to Invariant Inference via the Fence Condition

We have seen that Hoare-queries cannot, in general, simulate equivalence and membership queries used in exact concept learning, but can do so for maximal

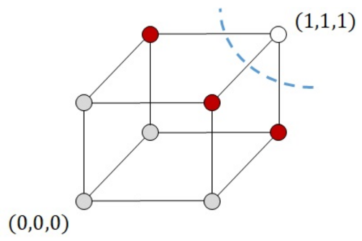


Fig. 1: The (outer) boundary of an invariant $I = x \wedge y \wedge z$, denoting the singleton set containing the far-top-right vertex of the 3-dimensional Boolean hypercube, $\{(1,1,1)\}$. Its neighbors are I's boundary (depicted in red): $\{(1,1,0),(1,0,1),(0,1,1)\}$. The rest of the vertices are in $\neg I$ but not in the boundary (depicted in gray). (Illustration inspired by [28, Fig. 2.1].)

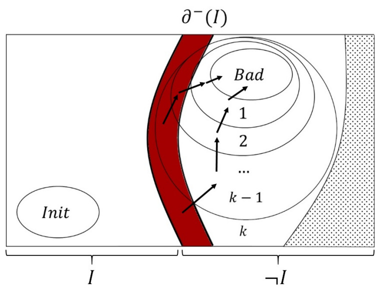


Fig. 2: An illustration of the fence condition. The boundary $\partial^-(I)$ of the invariant (the states in $\neg I$ nearest to I, in red) are backwards k-reachable (reach a bad state in k steps, for example by the transitions depicted by the arrows), but not all states in $\neg I$ are backwards k-reachable (or even backwards reachable at all, in the dotted area).

systems. This is somewhat disheartening; it would have been much nicer to apply an algorithm for learning a class of formulas $\mathcal{L}$ to the problem of inferring invariants from $\mathcal{L}$. In this section we present the *fence condition*, which was introduced in [11]. This condition relaxes the maximality property, and we show that it facilitates simulation of certain exact concept learning algorithms. In particular, we obtain the model-based interpolation-based algorithm of [7, 3]—as well as a new algorithm extending it—by a translation from exact learning algorithms that satisfy certain restrictions. The translation also lets us import complexity upper bounds for the obtained inference algorithms from the learning algorithms, revealing new results on the efficiency of inference algorithms provided that the fence condition holds. We further show that when a two-sided fence condition holds, *every* algorithm for exact learning from equivalence and membership queries can be transformed to a Hoare-query inference algorithm.

5.1 The Boundary of Inductive Invariants

The fence condition relates reachability in the transition system and the geometric notion of the *boundary* of the invariant.

Definition 7 (Boundary). *Let I be a set of states. Then the (outer) boundary of I, denoted $\partial^-(I)$, is the set of states $\sigma^- \not\models I$ s.t. there is a state σ^+ that differs from σ^- in exactly one variable, and $\sigma^+ \models I$.*

Definition 8 (Backwards k-Fenced). *For a transition system $(Init, \delta, Bad)$, an inductive invariant I is* backwards k-fenced *for $k \in \mathbb{N}$ if every state in $\partial^-(I)$ can reach Bad in at most k steps.*

More explicitly, an invariant I is backwards k-fenced if every state in $\neg I$ that has a Hamming neighbor in I (these are the states in the outer boundary of I) can reach Bad in at most k steps. For an illustration of the boundary and the fence condition see Figs. 1 and 2.

Example 1. In a maximal system, the (unique) inductive invariant I is backwards 1-fenced, since every state that is not part of I, in particular a state in the outer boundary, has a transition to every other state, including Bad.

In every system, this condition holds for at least one inductive invariant and for some finite k: the weakest inductive invariant, which allows all states but those that can reach Bad in any number of steps, satisfies the condition with the co-diameter, the number of steps that takes for all states that can reach Bad to do so.

Lemma 1. *Every safe transition system $TS = (Init, \delta, Bad)$ admits an inductive invariant $gfp = \{\sigma \mid \forall \sigma' \in Bad.\ (\sigma, \sigma') \notin \delta^*\}$ that is backwards k-fenced for k that is the co-diameter: the minimal k such that for every state σ: $\big(\exists \sigma' \in Bad.\ (\sigma, \sigma') \in \delta^*\big) \implies \big(\exists \sigma'' \in Bad.\ (\sigma, \sigma'') \in \delta^{\leq k}\big)$.*

While this lemma shows the existence of a backwards-fenced invariant through the gfp and co-diameter, the k-fence condition is more liberal: it can hold also for an invariant when not *every* state in $\neg I$ reaches Bad in k steps (or at all), and only the states in $\partial^-(I)$ do. An example demonstrating this follows.

Example 2. Consider an example of a (doubly)-linked list traversal, using `i` to traverse the list backwards, modeled via predicate abstraction following Itzhaky et al. [19]. The list starts at `h`. Initially, `i` points to some location that may or may not be part of the list, and in each step the system goes from `i` to its predecessor, until that would reach `x`. We write $\mathtt{s} \rightsquigarrow r$ to denote that r is reachable from `s` by following zero or more links. Consider the initial assumption $\mathtt{h} \rightsquigarrow \mathtt{x}$, but $\mathtt{i} \not\rightsquigarrow \mathtt{x}$ (it may be that $\mathtt{x} \rightsquigarrow \mathtt{i}$, or that `i` is not at all in the list). The bad states are those where $\mathtt{i} = \mathtt{h}$.

An inductive invariant for this system is $\mathtt{h} \rightsquigarrow \mathtt{x} \wedge \neg \mathtt{i} \rightsquigarrow \mathtt{x}$. In predicate abstraction, we may take the predicates $p_{h,x} = \mathtt{h} \rightsquigarrow \mathtt{x}$, $p_{i,x} = \mathtt{i} \rightsquigarrow \mathtt{x}$, and write $I = p_{h,x} \wedge \neg p_{i,x}$, which is a DNF invariant with one term. Hence $\neg I \equiv \neg p_{h,x} \vee p_{i,x}$. The outer boundary $\partial^-(I)$ consists of the states (1) $p_{h,x} = \mathit{false}, p_{i,x} = \mathit{false}$ and (2) $p_{h,x} = \mathit{true}, p_{i,x} = \mathit{true}$. Both states are in fact bad states under the abstraction: both include a state where $\mathtt{i} = \mathtt{h}$, from which `x` is unreachable (in (1)) or reachable (in (2)). Thus, I is backwards k-fenced for every $k \geq 0$.

In contrast, not all the states in $\neg I$ reach bad states (in particular, I is not the gfp): the state $p_{h,x} = \mathit{false}, p_{i,x} = \mathit{true}$ abstracts only states where $\mathtt{h} \not\rightsquigarrow \mathtt{i}$, and this remains true after going to the predecessor of `i`. This shows that the fence condition may hold even though I is not the gfp, and not all states in $\neg I$ reach bad states (in k steps or at all).

5.2 Inference From One-Sided Fence and Exact Learning With Restricted Queries

The challenge in harnessing exact learning algorithms for invariant inference is the need to also implement the teacher, which is problematic because the inference algorithm does not know any inductive invariant in advance [15], and, as we have shown in §4.2, is unable to efficiently implement a classical teacher that answers equivalence and membership queries, even in the more general Hoare-query model. In this section we overcome this problem using the fence condition, provided that the learning algorithm satisfies some conditions.

First, for equivalence queries, as discussed in §4.3, inductiveness queries can determine if a candidate formula is an inductive invariant. However, when it is not, the difficulty is the ambiguity of counterexamples to induction (σ, σ'), which makes it difficult to know which of σ or σ' should be returned to the learner as an example that differentiates the candidate from the invariant. We circumvent this problem by simply considering algorithms that query only on candidates which are underapproximations of the target I:

Lemma 2 (Implementing positive equivalence queries). *Let $(Init, \delta, Bad)$ be a transition system and I an inductive invariant. Given θ such that $\theta \Longrightarrow I$, it is possible to decide whether θ is an inductive invariant or provide a counterexample $\sigma \models I, \sigma \not\models \theta$, by*

- *checking whether there is a counterexample $\sigma' \models Init \wedge \neg\theta$ and returning σ' if one exists; and*
- *using an inductiveness query $\mathcal{I}(\delta, \theta)$ to check whether there is a counterexample to induction $(\sigma, \sigma') \models \theta \wedge \delta \wedge \neg\theta'$, and returning σ' if one exists.*

Otherwise, θ is an inductive invariant.

Note that $\theta \not\equiv I$ could be an inductive invariant, which does not amount to an equivalence query *per se*, but then the algorithm has already found an inductive invariant and can stop.

To implement membership queries, we rely on the fence condition. Our main observation here is that if the fence condition holds for I, then it is possible to efficiently implement restricted versions of membership queries:

Lemma 3 (Implementing positive-adjacent membership queries). *Let $(Init, \delta, Bad)$ be a transition system and I an inductive invariant that is backwards k-fenced. Given σ s.t. $\sigma \models I$ or $\sigma \in \partial^-(I)$, it is possible to decide whether $\sigma \models I$ by a single Hoare query that checks if $\mathcal{H}^{(k)}(\delta, cube(\sigma), \neg Bad) = true$ and answers accordingly.*

In fact, under similar restrictions, we can implement *subset* queries, which generalize membership queries. In a subset query, the learning algorithm *chooses* a formula θ, and the teacher answers whether $\theta \Longrightarrow I$, where I is the target formula. (A membership query for σ is a subset query with $\theta = \{\sigma\}$.)

Lemma 4 (Implementing positive-adjacent subset queries). *Let $(\mathit{Init}, \delta, \mathit{Bad})$ be a transition system and I an inductive invariant that is backwards k-fenced. Given θ s.t. $\theta \Longrightarrow I$ or $\theta \land \partial^-(I) \not\equiv \bot$, it is possible to decide whether $\theta \Longrightarrow I$ by a single Hoare query that checks if $\mathcal{H}^{(k)}(\delta, \theta, \neg \mathit{Bad}) = \mathit{true}$ and answers accordingly.*

A learning algorithm that only performs such queries induces a Hoare-query invariant inference algorithm that simulates it by implementing its queries as above. If the fence condition holds, all queries are answered correctly by the simulation, perhaps except for an equivalence query on θ returning *true* although $\theta \not\equiv I$, but then we have already found an inductive invariant θ and can stop. An additional inductiveness check is used in the inference algorithm before an invariant is returned to ensure that the result is a correct inductive invariant even when the fence condition does not hold. If the latter inductiveness check fails, the algorithm returns "failure".

Corollary 2. *Let C be a class of formulas. Let $\mathcal{A}$ be an exact concept learning algorithm that can identify every $\varphi \in C$ in at most s_1 equivalence queries and s_2 subset queries (including membership queries). Assume further that when $\mathcal{A}$ performs an equivalence query on θ, always $\theta \Longrightarrow \varphi$, and when $\mathcal{A}$ performs a subset query on θ, always $\theta \Longrightarrow \varphi$ or $\theta \land \partial^-(\varphi) \not\equiv \bot$. Then there exists a Hoare-query invariant inference algorithm that is sound (returns only correct invariants), and, furthermore, can find an inductive invariant for every transition system that admits an inductive invariant $I \in C$ that is backwards k-fenced using at most $s_1 + 1$ inductiveness and s_2 Hoare queries (with argument k), band time the same as of $\mathcal{A}$ up to a constant factor.*

The inference algorithm is sound even when the fence condition does not hold, although in this case successful and efficient convergence is not guaranteed.

Efficient interpolation-based inference of monotone invariants through exact learning Alg. 2 presents the interpolation-based invariant inference algorithm due to Chockler et al. and Bjørner et al. [7, 3], which uses a model-based method for interpolant construction, inspired by IC3/PDR [4, 8], rather than constructing interpolants from proofs as in McMillan's original algorithm [25]. Alg. 2 starts with the candidate invariant $\varphi = \mathit{Init}$, which is gradually increased to include more states. In each iteration, the algorithm performs an inductiveness query (lines 3 and 4) and terminates if an inductive invariant has been found. If a counterexample to induction (σ, σ') exists, the algorithm generates a term d which includes the post-state σ', and disjoins d to φ to obtain the new candidate (line 11). To obtain d, the algorithm starts with $\mathit{cube}(\sigma')$—the conjunction that exactly captures σ'—and drops literals as long as no state in d can reach a bad state in k steps or less (line 9). These checks are done via Hoare queries. If σ' itself reaches a bad state in k steps, no invariant weaker than φ exists, and the algorithm restarts with a larger bound k (line 6).

Interestingly, Alg. 2 is essentially the result of the transformation of Corollary 2 applied to the exact learning algorithm for DNF formulas of Aizenstein and Pitt [1] as it appears in Alg. 3 (similar to the algorithm by Angluin [2]), where EQ denotes an equivalence query and SQ denotes a subset query. The differences are the additional check in line 6 in Alg. 2, meant to detect failure, and the initialization of φ to *Init* instead of *false*, which can be viewed as an optimization.

Algorithm 2 Interpolation-based inference by term minimization

```
1:  procedure MB-ITP($Init$, $[\delta]$, $Bad$, $k$)
2:      $\varphi \leftarrow Init$
3:      while $\mathcal{I}(\delta, \varphi) \neq true$ do
4:          $(\sigma, \sigma') \leftarrow \mathcal{I}(\delta, I)$
5:          if $\mathcal{H}^{(k)}(\delta, \sigma', \neg Bad) \neq true$ then
6:              restart with larger $k$
7:          $d \leftarrow cube(\sigma')$
8:          for $\ell$ in $d$ do
9:              if $\mathcal{H}^{(k)}(\delta, d \setminus \{\ell\}, \neg Bad) = true$ then
10:                 $d \leftarrow d \setminus \{\ell\}$
11:         $\varphi \leftarrow \varphi \vee d$
12:     return $I$
```

Algorithm 3 Exact concept learning of DNF formulas [36, 2, 1]

```
1:  procedure LEARN-DNF
2:      $\varphi \leftarrow false$
3:      while EQ $(\varphi)$ is not $\bot$ do
4:          $\sigma' \leftarrow$ EQ $(\varphi)$
5:
6:
7:          $d \leftarrow cube(\sigma')$
8:          for $\ell$ in $d$ do
9:              if SQ $(d \setminus \{\ell\}) = true$ then
10:                 $d \leftarrow d \setminus \{\ell\}$
11:         $\varphi \leftarrow \varphi \vee d$
12:     return $\varphi$
```

The queries performed in Alg. 3 satisfy the conditions of the transformation: the hypothesis φ is always below the true formula, as required for equivalence queries; the subset queries are always positive adjacent, because if d is a term s.t. $d \Longrightarrow \psi$, and $d' \not\Longrightarrow \psi$ where $d' = d \setminus \{\ell\}$, then taking a state $\sigma^- \models d' \wedge \neg\psi$ and flipping the variable in ℓ results in a state $\sigma^+ \models d$ and hence $\sigma^+ \models \psi$, hence $\sigma^- \models \partial^-(\psi)$ and $\sigma^- \models d'$, as required. As such, the transformation also yields an efficiency result for Alg. 2 which is carried over from the efficiency of Alg. 3 for monotone DNF formulas [1, 2]:

Theorem 4. *Let $(Init, \delta, Bad)$ be a transition system with $|\Sigma| = n$ and $k \in \mathbb{N}$. If there is an inductive invariant $I \in$ Mon-DNF$_m$ that is backwards k-fenced, then* MB-ITP*$(Init, [\delta], Bad, k)$ converges to an inductive invariant in $O(m)$ inductiveness queries, $O(mn)$ Hoare queries (with argument k), and $O(mn)$ time.*

Thm. 4 focuses on efficiency of MB-ITP for monotone invariants under the fence condition; in [11] we also show that if *any* k-fenced inductive invariant I exists (not necessarily monotone), then the check in line 5 never fails, hence convergence with reachability bound k is guaranteed. Together with Lemma 1 this provides an alternative proof of termination for MB-ITP.

Efficient inference of invariants with a known monotone basis through exact learning Bshouty [5] investigated exact learning of formulas that are not monotone. To this end, he introduced the *monotone theory*. The idea is that a formula φ can be reconstructed as the conjunction of its *monotonizations*, $\mathcal{M}_b(\varphi)$, w.r.t. elements b in a set B that forms a *monotone basis* for φ; a set of states B is a monotone basis for φ if $\varphi \equiv \bigwedge_{b \in B} \mathcal{M}_b(\varphi)$ (such a set always exists, and is related to CNF representations of φ). Bshouty's Λ-algorithm can efficiently learn φ, while using equivalence and membership queries, provided that the monotone basis is known a-priori, and is amenable to the transformation in Corollary 2, resulting in a Hoare-query algorithm, Λ-Inference, that can *efficiently* learn every formula for which $B = \{b_1, \ldots, b_t\}$ is a basis when the k-fenced condition holds:

Theorem 5. *Let* $(\mathit{Init}, \delta, \mathit{Bad})$ *be a transition system with* $|\Sigma| = n$*, and* $k \in \mathbb{N}$*. If there exists an inductive invariant* I *that is backwards* k*-fenced,* $I \in \mathrm{DNF}_m$*, and* $B = \{b_1, \ldots, b_t\}$ *is a monotone basis for* I*, then* Λ-Inference$(\mathit{Init}, [\delta], \mathit{Bad}, k)$ *converges to an inductive invariant in* $O(m \cdot t)$ *inductiveness checks,* $O(m \cdot t \cdot n^2)$ k*-BMC checks, and* $O(m \cdot t \cdot n^2)$ *time.*

Choosing a Monotone Basis Some important classes of formulas admit a known basis that the algorithm can use. The class of *r-almost-monotone DNF* is the class of DNF formulas with at most r terms which include negative literals. The set of all states with at most r variables assigned *true* is a basis for this class [5]. When $r = O(1)$, the size of this basis is polynomial in $n = |\Sigma|$. Another interesting class with a known base of size polynomial in n is the class of (arbitrary) DNF formulas with $O(\log n)$ terms, although the construction is less elementary [5].

Applying Thm. 5 with the known basis for r-almost-monotone DNF yields:

Corollary 3. *Let* $(\mathit{Init}, \delta, \mathit{Bad})$ *be a transition system with* $|\Sigma| = n$*,* $k \in \mathbb{N}$*, and* $r = O(1)$*. If there exists an inductive invariant* I *that is backwards* k*-fenced, and* I *is* r*-almost-monotone DNF with* m *terms, then* Λ-Inference$(\mathit{Init}, [\delta], \mathit{Bad}, k)$ *with an appropriate basis converges to an inductive invariant in poly*$(m \cdot n)$ *inductiveness checks, poly*$(m \cdot n)$ k*-BMC checks, and poly*$(m \cdot n)$ *time.*

Dual inference under the dual fence condition Safety problems enjoy a duality between the initial states and the bad states: a formula I is an inductive invariant w.r.t. $(\mathit{Init}, \delta, \mathit{Bad})$ iff the dual formula $\neg I$ is an inductive invariant w.r.t. the dual transition system $(\mathit{Bad}, \delta^{-1}, \mathit{Init})$. This gives rise to dual algorithms that, given as input $(\mathit{Init}, \delta, \mathit{Bad})$, infer an invariant for the dual problem and return the dual invariant. Dual algorithms allow us to translate complexity results from the inference of CNF invariants to the inference of DNF invariants and vice versa. Since our results are conditioned upon the backwards-fence condition, we need to dualize it as well:

Definition 9 (Forward k-Fenced). *I is* k-forward fenced *if every state in* $\partial^+(I)$ *is reachable from Init in at most* k *steps, where* $\partial^+(I)$ *is the inner boundary of* I*, the set of states* $\sigma^+ \models I$ *s.t. there is a state* $\sigma^- \not\models I$ *that differs from* σ^+ *in exactly one variable.*

Using duality, we derive efficiency results under the k-forward fence condition for antimonotone CNF invariants (from Thm. 4), and for r-almost antimonotone CNF invariants, which are CNF formulas with at most r clauses that include positive literals (from Corollary 3).

5.3 Inference From Two-Sided Fence and Exact Learning

The previous section has provided a translation of exact learning algorithms, but only those that admit certain requirements on their queries. In this section we simulate arbitrary exact learning algorithms (going beyond the requirements in Corollary 2) relying on a *two-sided* fence condition. An important example of such an exact learning algorithm is the CDNF algorithm by Bshouty [5]. The conditions of the transformation in §5.2 do not hold because this algorithm performs equivalence queries that can return either positive or negative examples. We now show how to implement any membership or equivalence query to the invariant using the two-sided fence condition.

Lemma 5 (Implementing membership queries). *Let $(Init, \delta, Bad)$ be a transition system with $|\Sigma| = n$ and I an (unknown) inductive invariant that is backwards k_1-fenced and forwards k_2-fenced. Then membership queries to I can be implemented in at most n Hoare queries with reachability bound k_1 and n Hoare queries with reachability bound k_2.*[5]

Given a membership query "$\sigma \in I$?", the idea is to choose some known state $\sigma_0 \in Init$, and gradually walk from σ to σ_0, that is, in each step change one variable in σ to match σ_0 and return *true* if $\mathcal{H}^{(k_1)}(\delta, Init, \neg cube(\sigma)) \neq true$, and false if $\mathcal{H}^{(k_1)}(\delta, cube(\sigma), \neg Bad) \neq true$ (otherwise continue the walk). The rational is that if $\sigma \in I$ but it is not reachable from *Init* in k_1 steps, the walk will eventually hit the inner boundary of I, which is guaranteed to be reachable in k_1 steps—as can be detected using a Hoare query with $k = k_1$—so that the corresponding Hoare query will return such a counterexample trace; similarly, if $\sigma \notin I$ the walk will eventually hit the outer boundary which is guaranteed to reach *Bad* in k_2 steps and the Hoare query will detect it.

An equivalence query can be implemented by an inductiveness query and a membership query (as was also noted in §4.2):

Lemma 6 (Implementing equivalence queries). *Let $(Init, \delta, Bad)$ be a transition system with $|\Sigma| = n$, and I an (unknown) inductive invariant that is forwards k_1-fenced and backwards k_2-fenced. Then given θ it is possible to answer whether θ is an inductive invariant, or provide a counterexample σ such that $\sigma \models \theta, \sigma \not\models I$ or $\sigma \not\models \theta, \sigma \models I$, using an inductiveness query, and at most n Hoare queries with bound k_1 and n Hoare queries with bound k_2.*

We can use these procedures to implement every exact learning algorithm from (arbitrary) equivalence and membership queries.

Corollary 4. *Let C be a class of formulas. Let $\mathcal{A}$ be an exact concept learning algorithm that can identify every $\varphi \in C$ in at most s_1 equivalence queries and s_2 membership queries. Then there exists a sound invariant inference algorithm that can find an inductive invariant for every transition system that admits an*

[5] The proof of this also implies that an invariant that is both forwards k_1-fenced and backwards k_2-fenced is unique, seeing that the implementation of the membership query for both is the same.

inductive invariant $I \in C$ that is forwards k_1-fenced and backwards k_2-fenced using at most $s_1 + 1$ inductiveness queries, $n(s_1 + s_2)$ Hoare queries with bound k_1, $n(s_1 + s_2)$ Hoare queries with bound k_2, and time $O(n(s_1 + s_2)t_{\mathcal{A}})$ where $t_{\mathcal{A}}$ is the worst-case time of $\mathcal{A}$ learning I and $n = |\Sigma|$.

Next, we demonstrate an application of Corollary 4 to the inference of a larger class of invariants.

Inference Beyond Almost-Monotone Invariants Earlier, we have shown that almost-monotone *DNF* invariants are efficiently inferrable when the *backwards* fence condition holds, and similarly for almost-antimonotone *CNF* when the *forwards* fence condition holds. We now apply Corollary 4 to the CDNF algorithm by Bshouty [5] to show that the class of invariants that can be succinctly expressed *both* in DNF and in CNF (not necessarily in an almost-monotone way) can be efficiently inferred when the fence condition holds in *both* directions:

Theorem 6. *There is an algorithm $\mathcal{A}$ that for every input transition system $(Init, \delta, Bad)$ with $|\Sigma| = n$ and $k \in \mathbb{N}$, if the system admits an inductive invariant I such that $I \in \mathrm{DNF}_{m_1}$, $I \in \mathrm{CNF}_{m_2}$, and I is both backwards- and forwards- k-fenced, then $\mathcal{A}(Init, [\delta], Bad, k)$ converges to an inductive invariant in $O(m_1 \cdot m_2)$ inductiveness queries, $O(m_1 \cdot m_2 \cdot n^3)$ Hoare queries with bound k, and $O(m_1 \cdot m_2 \cdot n^3)$ time.*

Such a complexity guarantee is significant, because, put differently, it shows that an invariant can be learned efficiently in terms of its smallest DNF and CNF representations (provided that the two-sided fence condition holds). Through an observation by Bshouty [5], this implies that it is possible to efficiently infer an invariant that admits a succinct representation as a *decision tree*: a binary tree in which every internal node is labeled by a variable and a leaf by *true*/*false*, and σ satisfies the formula if the path defined by starting from the root, turning left when the σ assigns *false* to the variable labeling the node and right otherwise, reaches a leaf *true*. The size of a decision tree is the number of leaves in the tree.

Corollary 5. *There is an algorithm $\mathcal{A}$ that for every input transition system $(Init, \delta, Bad)$ with $|\Sigma| = n$ and $k \in \mathbb{N}$, if the system admits an inductive invariant I that can be expressed as a decision tree of size m, and I is both backwards- and forwards- k-fenced, then $\mathcal{A}(Init, [\delta], Bad, k)$ converges to an inductive invariant in $O(m^2)$ inductiveness queries, $O(m^2 \cdot n^3)$ Hoare queries with bound k, and $O(m^2 \cdot n^3)$ time.*

Similarly, when an r-almost-unate invariant with $O(\log n)$ non-unate variables is fenced both backwards and forwards, it can be inferred by an adaptation of an algorithm by Bshouty [6].

Inference Beyond Concept Learning: Efficient Inference of CDNF Invariants from One-Sided Fence Condition The previous section has arrived at an extremely almost-satisfying result: that any invariant can be inferred in time proportional to the size of its smallest representations in DNF and CNF and the number of variables. The culprit is that the translation from Bshouty's CDNF algorithm is possible only under the *two-sided* fence condition, which is significantly stronger than a one-sided fence condition. In [13] we show that the same result is also attainable under the one-sided fence condition, where the Hoare-query algorithm we use cannot be understood as a direct translation of an exact concept learning algorithm—it builds heavily on Bshouty's CDNF algorithm, but modifies it in important ways, while still accessing the transition relation only through Hoare queries. Specifically, the forwards k-fence condition ensures that the set S of states reachable in at most k steps satisfies $\partial^+(I) \subseteq S \subseteq I$. The algorithm relies on this property to construct a formula H that contains I by sampling and generalizing states from S in a certain way that guarantees that $\partial^+(I) \subseteq H \implies I \subseteq H$. In this way, the algorithm relies on the fact that $\partial^+(I) \subseteq S$ (thanks to the fence condition) to ensure that after sampling enough states from S and using them to increase H, once $\mathcal{H}^{(k)}(\delta, \mathit{Init}, H){=}\mathit{true}$ holds (i.e., $S \subseteq H$), then it is also guaranteed that $I \subseteq H$. This process has no analog in exact concept learning, because, there, we are not given any set S that is related to the boundary of the target concept.

6 Conclusion

This paper surveyed results that formally established the relation between SAT-based invariant inference and exact learning with queries, and utilized it to illuminate some of the fundamental questions about invariant inference. There is still much to understand about this topic. In particular, it is interesting to show separation between Hoare queries that use different lengths of executions k, which could indicate that bounded model checking in principle provides additional power. The boundaries of the ability to translate learning algorithms to invariant inference under the fence condition could be clarified by showing that general membership queries are impossible to implement even under the fence condition, justifying the two-sided condition for a general transformation. Finally, other translations that build on reachability conditions other than the fence condition could help explain inference algorithms other than model-based interpolation, and pave the way for new algorithms that are efficient in practice.

Acknowledgement The research leading to these results has received funding from the European Research Council under the European Union's Horizon 2020 research and innovation programme (grant agreement No [759102-SVIS]). This research was partially supported by the United States-Israel Binational Science Foundation (BSF) grant No. 2016260, and the Israeli Science Foundation (ISF) grant No. 1810/18.

Bibliography

[1] Aizenstein, H., Pitt, L.: On the learnability of disjunctive normal form formulas. Mach. Learn. **19**(3), 183–208 (1995). https://doi.org/10.1007/BF00996269, https://doi.org/10.1007/BF00996269

[2] Angluin, D.: Queries and concept learning. Machine Learning **2**(4), 319–342 (1987)

[3] Bjørner, N., Gurfinkel, A., Korovin, K., Lahav, O.: Instantiations, zippers and EPR interpolation. In: LPAR 2013, 19th International Conference on Logic for Programming, Artificial Intelligence and Reasoning, December 12-17, 2013, Stellenbosch, South Africa, Short papers proceedings. pp. 35–41 (2013), https://easychair.org/publications/paper/XtN

[4] Bradley, A.R.: 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. pp. 70–87 (2011). https://doi.org/10.1007/978-3-642-18275-4_7, http://dx.doi.org/10.1007/978-3-642-18275-4_7

[5] Bshouty, N.H.: Exact learning boolean function via the monotone theory. Inf. Comput. **123**(1), 146–153 (1995). https://doi.org/10.1006/inco.1995.1164, https://doi.org/10.1006/inco.1995.1164

[6] Bshouty, N.H.: Simple learning algorithms using divide and conquer. Comput. Complex. **6**(2), 174–194 (1997). https://doi.org/10.1007/BF01262930, https://doi.org/10.1007/BF01262930

[7] Chockler, H., Ivrii, A., Matsliah, A.: Computing interpolants without proofs. In: Hardware and Software: Verification and Testing - 8th International Haifa Verification Conference, HVC 2012, Haifa, Israel, November 6-8, 2012. Revised Selected Papers. pp. 72–85 (2012). https://doi.org/10.1007/978-3-642-39611-3_12, https://doi.org/10.1007/978-3-642-39611-3_12

[8] Eén, N., Mishchenko, A., Brayton, R.K.: Efficient implementation of property directed reachability. In: International Conference on Formal Methods in Computer-Aided Design, FMCAD '11, Austin, TX, USA, October 30 - November 02, 2011. pp. 125–134 (2011), http://dl.acm.org/citation.cfm?id=2157675

[9] Ezudheen, P., Neider, D., D'Souza, D., Garg, P., Madhusudan, P.: Horn-ice learning for synthesizing invariants and contracts. PACMPL **2**(OOPSLA), 131:1–131:25 (2018)

[10] Feldman, Y.M.Y., Immerman, N., Sagiv, M., Shoham, S.: Complexity and information in invariant inference. Proc. ACM Program. Lang. **4**(POPL), 5:1–5:29 (2020). https://doi.org/10.1145/3371073, https://doi.org/10.1145/3371073

[11] Feldman, Y.M.Y., Sagiv, M., Shoham, S., Wilcox, J.R.: Learning the boundary of inductive invariants. Proc. ACM Program. Lang. **5**(POPL), 1–30 (2021). https://doi.org/10.1145/3434296, https://doi.org/10.1145/3434296

[12] Feldman, Y.M.Y., Sagiv, M., Shoham, S., Wilcox, J.R.: Property-directed reachability as abstract interpretation in the monotone theory. Proc. ACM Program. Lang. **6**(POPL), 1–31 (2022). https://doi.org/10.1145/3498676, https://doi.org/10.1145/3498676

[13] Feldman, Y.M.Y., Shoham, S.: Invariant inference with provable complexity from the monotone theory. In: Static Analysis - 29th International Symposium, SAS 2022, Auckland, New Zealand. To appear. (2022)

[14] Flanagan, C., Leino, K.R.M.: 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. pp. 500–517 (2001)

[15] Garg, P., Löding, C., Madhusudan, P., Neider, D.: Ice: A robust framework for learning invariants. In: Computer Aided Verification. pp. 69–87. Springer (2014)

[16] Garg, P., Neider, D., Madhusudan, P., Roth, D.: Learning invariants using decision trees and implication counterexamples. In: Proceedings of the 43rd Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL 2016, St. Petersburg, FL, USA, January 20 - 22, 2016. pp. 499–512 (2016). https://doi.org/10.1145/2837614.2837664, https://doi.org/10.1145/2837614.2837664

[17] Hellerstein, L., Kletenik, D., Sellie, L., Servedio, R.A.: Tight bounds on proper equivalence query learning of DNF. In: COLT 2012 - The 25th Annual Conference on Learning Theory, June 25-27, 2012, Edinburgh, Scotland. pp. 31.1–31.18 (2012), http://proceedings.mlr.press/v23/hellerstein12/hellerstein12.pdf

[18] Henzinger, T.A., Jhala, R., Majumdar, R., McMillan, K.L.: Abstractions from proofs. In: Proceedings of the 31st ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL 2004, Venice, Italy, January 14-16, 2004. pp. 232–244 (2004). https://doi.org/10.1145/964001.964021, https://doi.org/10.1145/964001.964021

[19] Itzhaky, S., Bjørner, N., Reps, T.W., Sagiv, M., Thakur, A.V.: Property-directed shape analysis. 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. pp. 35–51 (2014). https://doi.org/10.1007/978-3-319-08867-9_3, http://dx.doi.org/10.1007/978-3-319-08867-9_3

[20] Jha, S., Gulwani, S., Seshia, S.A., Tiwari, A.: Oracle-guided component-based program synthesis. In: Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering - Volume 1, ICSE 2010, Cape Town, South Africa, 1-8 May 2010. pp. 215–224 (2010). https://doi.org/10.1145/1806799.1806833, https://doi.org/10.1145/1806799.1806833

[21] Jha, S., Seshia, S.A.: A theory of formal synthesis via inductive learning. Acta Inf. **54**(7), 693–726 (2017). https://doi.org/10.1007/s00236-017-0294-5, https://doi.org/10.1007/s00236-017-0294-5

[22] Jhala, R., McMillan, K.L.: Interpolant-based transition relation approximation. Logical Methods in Computer Science **3**(4) (2007). https://doi.org/10.2168/LMCS-3(4:1)2007, https://doi.org/10.2168/LMCS-3(4:1)2007

[23] Koenig, J.R., Padon, O., Immerman, N., Aiken, A.: First-order quantified separators. In: Donaldson, A.F., Torlak, E. (eds.) Proceedings of the 41st ACM SIGPLAN International Conference on Programming Language Design and Implementation, PLDI 2020, London, UK, June 15-20, 2020. pp. 703–717. ACM (2020), https://doi.org/10.1145/3385412.3386018

[24] Lahiri, S.K., Qadeer, S.: Complexity and algorithms for monomial and clausal predicate abstraction. In: Automated Deduction - CADE-22, 22nd International Conference on Automated Deduction, Montreal, Canada, August 2-7, 2009. Proceedings. pp. 214–229 (2009)

[25] McMillan, K.L.: Interpolation and sat-based model checking. In: Computer Aided Verification, 15th International Conference, CAV 2003, Boulder, CO, USA, July 8-12, 2003, Proceedings. pp. 1–13 (2003)

[26] McMillan, K.L.: Lazy abstraction with interpolants. In: Computer Aided Verification, 18th International Conference, CAV 2006, Seattle, WA, USA, August 17-20, 2006, Proceedings. pp. 123–136 (2006). https://doi.org/10.1007/11817963_14, https://doi.org/10.1007/11817963_14

[27] Neider, D., Madhusudan, P., Saha, S., Garg, P., Park, D.: A learning-based approach to synthesizing invariants for incomplete verification engines. J. Autom. Reason. **64**(7), 1523–1552 (2020). https://doi.org/10.1007/s10817-020-09570-z, https://doi.org/10.1007/s10817-020-09570-z

[28] O'Donnell, R.: Analysis of Boolean Functions. Cambridge University Press (2014), http://www.cambridge.org/de/academic/subjects/computer-science/algorithmics-complexity-computer-algebra-and-computational-g/analysis-boolean-functions

[29] Quine, W.: Two theorems about truth-functions. Boletín de la Sociedad Matemática Mexicana **10**(1–2), 64–70 (1954)

[30] Reps, T.W., Sagiv, S., Yorsh, G.: Symbolic implementation of the best transformer. In: Verification, Model Checking, and Abstract Interpretation, 5th International Conference, VMCAI 2004, Venice, Italy, January 11-13, 2004, Proceedings. pp. 252–266 (2004). https://doi.org/10.1007/978-3-540-24622-0_21, https://doi.org/10.1007/978-3-540-24622-0_21

[31] Sharma, R., Aiken, A.: From invariant checking to invariant inference using randomized search. Formal Methods in System Design **48**(3), 235–256 (2016). https://doi.org/10.1007/s10703-016-0248-5, https://doi.org/10.1007/s10703-016-0248-5

[32] Sharma, R., Gupta, S., Hariharan, B., Aiken, A., Liang, P., Nori, A.V.: A data driven approach for algebraic loop invariants. In: Programming Languages and Systems - 22nd European Symposium on Programming, ESOP 2013, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2013, Rome, Italy, March 16-24, 2013. Proceedings. pp. 574–592 (2013). https://doi.org/10.1007/978-3-642-37036-6_31, https://doi.org/10.1007/978-3-642-37036-6_31

[33] Sharma, R., Gupta, S., Hariharan, B., Aiken, A., Nori, A.V.: Verification as learning geometric concepts. In: Static Analysis - 20th International Symposium, SAS 2013, Seattle, WA, USA, June 20-22, 2013. Proceedings. pp. 388–411 (2013)

[34] Sharma, R., Nori, A.V., Aiken, A.: Interpolants as classifiers. In: Computer Aided Verification - 24th International Conference, CAV 2012, Berkeley, CA, USA, July 7-13, 2012 Proceedings. pp. 71–87 (2012). https://doi.org/10.1007/978-3-642-31424-7_11, https://doi.org/10.1007/978-3-642-31424-7_11

[35] Thakur, A.V., Lal, A., Lim, J., Reps, T.W.: Posthat and all that: Automating abstract interpretation. Electr. Notes Theor. Comput. Sci. **311**, 15–32 (2015). https://doi.org/10.1016/j.entcs.2015.02.003, https://doi.org/10.1016/j.entcs.2015.02.003

[36] Valiant, L.G.: A theory of the learnable. Commun. ACM **27**(11), 1134–1142 (1984). https://doi.org/10.1145/1968.1972, https://doi.org/10.1145/1968.1972

[37] Vizel, Y., Grumberg, O.: Interpolation-sequence based model checking. In: Proceedings of 9th International Conference on Formal Methods in Computer-Aided Design, FMCAD 2009, 15-18 November 2009, Austin, Texas, USA. pp. 1–8 (2009). https://doi.org/10.1109/FMCAD.2009.5351148, https://doi.org/10.1109/FMCAD.2009.5351148

[38] Vizel, Y., Grumberg, O., Shoham, S.: Intertwined forward-backward reachability analysis using interpolants. In: Tools and Algorithms for the Construction and Analysis of Systems - 19th International Conference, TACAS 2013, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2013, Rome, Italy, March 16-24, 2013. Proceedings. pp. 308–323 (2013). https://doi.org/10.1007/978-3-642-36742-7_22, https://doi.org/10.1007/978-3-642-36742-7_22

[39] Vizel, Y., Gurfinkel, A., Shoham, S., Malik, S.: IC3 - flipping the E in ICE. In: Verification, Model Checking, and Abstract Interpretation - 18th International Conference, VMCAI 2017, Paris, France, January 15-17, 2017, Proceedings. pp. 521–538 (2017)