Complexity and Information in Invariant Inference

YOTAM M. Y. FELDMAN, Tel Aviv University, Israel
NEIL IMMERMAN, UMass Amherst, USA
MOOLY SAGIV, Tel Aviv University, Israel
SHARON SHOHAM, Tel Aviv University, Israel

This paper addresses the complexity of SAT-based invariant inference, a prominent approach to safety verification. We consider the problem of inferring an inductive invariant of *polynomial length* given a transition system and a safety property. We analyze the complexity of this problem in a black-box model, called *the Hoare-query model*, which is general enough to capture algorithms such as IC3/PDR and its variants. An algorithm in this model learns about the system's reachable states by querying the validity of Hoare triples.

We show that in general an algorithm in the Hoare-query model requires an exponential number of queries. Our lower bound is information-theoretic and applies even to computationally unrestricted algorithms, showing that no choice of generalization from the partial information obtained in a polynomial number of Hoare queries can lead to an efficient invariant inference procedure in this class.

We then show, for the first time, that by utilizing rich Hoare queries, as done in PDR, inference can be exponentially more efficient than approaches such as ICE learning, which only utilize inductiveness checks of candidates. We do so by constructing a class of transition systems for which a simple version of PDR with a single frame infers invariants in a polynomial number of queries, whereas every algorithm using only inductiveness checks and counterexamples requires an exponential number of queries.

Our results also shed light on connections and differences with the classical theory of exact concept learning with queries, and imply that learning from counterexamples to induction is harder than classical exact learning from labeled examples. This demonstrates that the convergence rate of Counterexample-Guided Inductive Synthesis depends on the form of counterexamples.

CCS Concepts: • **Theory of computation → Theory and algorithms for application domains**; **Program verification**; • **Software and its engineering → Formal methods**.

Additional Key Words and Phrases: invariant inference, complexity, synthesis, exact learning, property-directed reachability

ACM Reference Format:
Yotam M. Y. Feldman, Neil Immerman, Mooly Sagiv, and Sharon Shoham. 2020. Complexity and Information in Invariant Inference. *Proc. ACM Program. Lang.* 4, POPL, Article 5 (January 2020), 29 pages. https://doi.org/10.1145/3371073

1 INTRODUCTION

The inference of inductive invariants is a fundamental technique in safety verification, and the focus of many works [e.g. Alur et al. 2015; Bradley 2011; Cousot and Cousot 1977; Dillig et al. 2013; Eén et al. 2011; Fedyukovich and Bodík 2018; McMillan 2003; Srivastava et al. 2013]. The task is to find an assertion I that holds in the initial states of the system, excludes all bad states, and is closed

Authors' addresses: Yotam M. Y. Feldman, Tel Aviv University, Israel, yotam.feldman@gmail.com; Neil Immerman, UMass Amherst, USA, immerman@cs.umass.edu; Mooly Sagiv, Tel Aviv University, Israel, msagiv@acm.org; Sharon Shoham, Tel Aviv University, Israel, sharon.shoham@gmail.com.

2475-1421/2020/1-ART5
https://doi.org/10.1145/3371073

under transitions of the system, namely, the Hoare triple $\{I\}\delta\{I\}$ is valid, where δ denotes one step of the system. Such an I overapproximates the set of reachable states and establishes their safety.

The advance of SAT-based reasoning has led to the development of successful algorithms inferring inductive invariants using SAT queries. A prominent example is IC3/PDR [Bradley 2011; Eén et al. 2011], which has led to a significant improvement in the ability to verify realistic hardware systems. Recently, this algorithm has been extended and generalized to software systems [e.g. Bjørner and Gurfinkel 2015; Cimatti et al. 2014; Hoder and Bjørner 2012; Karbyshev et al. 2017; Komuravelli et al. 2014].

Successful SAT-based inference algorithms are typically tricky and employ many clever heuristics. This is in line with the inherent asymptotic complexity of invariant inference, which is hard even with access to a SAT solver [Lahiri and Qadeer 2009]. However, the practical success of inference algorithms calls for a more refined complexity analysis, with the objective of understanding the principles on which these algorithms are based. This paper studies the asymptotic complexity of SAT-based invariant inference through the decision problem of *polynomial length inference* in the black-box *Hoare-query model*, as we now explain.

Inference of polynomial-length CNF. Naturally, inference algorithms succeed when the invariant they infer is not too long. Therefore, this paper considers the complexity of *inferring invariants of polynomial length*. We follow the recent trend in invariant inference, advocated in [Bradley 2011; McMillan 2003], to search for invariants in rich syntactical forms, beyond those usually considered in template-based invariant inference [e.g. Alur et al. 2015; Colón et al. 2003; Jeannet et al. 2014; Sankaranarayanan et al. 2004; Srivastava and Gulwani 2009; Srivastava et al. 2013], with the motivation of achieving generality of the verification method and potentially improving the success rate. We thus study the inference of invariants expressed in Conjunctive Normal Form (CNF) of polynomial length. Interestingly, our results also apply to inferring invariants in Disjunctive Normal Form.

The Hoare-query model. Our study of SAT-based methods focuses on an algorithmic model called the *Hoare-query model*. The idea is that the inference algorithm is not given direct access to the program, but performs *queries* on it. In the Hoare-query model, algorithms repeatedly choose α, β and query for the validity of Hoare triples $\{\alpha\}\delta\{\beta\}$, where δ is the transition relation denoting one step of the system, inaccessible to the algorithm but via such Hoare queries. The check itself is implemented by an oracle, which in practice is a SAT solver. This model is general enough to capture algorithms such as PDR and its variants, and leaves room for other interesting design choices, but does not capture white-box approaches such as abstract interpretation [Cousot and Cousot 1977]. The advantage of this model for a theoretical study is that it enables an information-based analysis, which (i) sidesteps open computational complexity questions, and therefore results in unconditional lower bounds on the computational complexity of SAT-based algorithms captured by the model, and (ii) grants meaning to questions about generalization from partial information we discuss later.

Results. This research addresses two main questions related to the core ideas behind PDR, and theoretically analyzes them in the context of the Hoare-query model:

(1) These algorithms revolve around the question of *generalization*: from observing concrete states (to be excluded from the invariant), the algorithm seeks to produce assertions that hold for *all* reachable states. The different heuristics in this context are largely understood as clever ways of performing this generalization. The situation is similar in interpolation-based algorithms, only that generalization is performed from bounded safety proofs rather than states. How should generalization be performed to achieve efficient invariant inference?

(2) A key aspect of PDR is the form of SAT checks it uses, as part of *relative inductiveness checks*, of Hoare triples $\{\alpha\}\delta\{\beta\}$ in which in general $\alpha \neq \beta$.[1] Repeated queries of this form are potentially richer than presenting a series of candidate invariants, where the check is $\{\alpha\}\delta\{\alpha\}$. Is there a benefit in using relative inductiveness beyond inductiveness checks?

We analyze these questions in the foundational case of Boolean programs, which is applicable to infinite-state systems through predicate abstraction [Flanagan and Qadeer 2002; Graf and Saïdi 1997; Lahiri and Qadeer 2009], and is also a core part of other invariant inference techniques for infinite-state systems [e.g. Hoder and Bjørner 2012; Karbyshev et al. 2017; Komuravelli et al. 2014].

In §6, we answer question 1 with an impossibility result, by showing that no choice of generalization can lead to an inference algorithm using only a polynomial number of Hoare queries. Our lower bound is information-theoretic, and holds even with unlimited computational power, showing that the problem of generalization is chiefly a question of information gathering.

In §7, we answer question 2 in the affirmative, by showing an exponential gap between algorithms utilizing rich $\{\alpha\}\delta\{\beta\}$ checks and algorithms that perform only inductiveness checks $\{\alpha\}\delta\{\alpha\}$. Namely, we construct a class of programs for which a simple version of PDR can infer invariants efficiently, but every algorithm learning solely from counterexamples to the inductiveness of candidates requires an exponential number of queries. This result shows, for the first time theoretically, the significance of relative inductiveness checks as the foundation of PDR's mechanisms, in comparison to a machine learning approach pioneered in the ICE model [Garg et al. 2014, 2016] that infers invariants based on inductiveness checks only (but of course this result does not mean that PDR is *always* more efficient than every ICE algorithm).

Our results also clarify the relationship between the problem of invariant inference and the classical theory of exact concept learning with queries [Angluin 1987]. In particular, our results imply that learning from counterexamples to induction is harder than learning from positive & negative examples (§8), providing a formal justification to the existing intuition [Garg et al. 2014]. This demonstrates that the convergence rate of learning in Counterexample-Guided Inductive Synthesis [e.g. Jha et al. 2010; Jha and Seshia 2017; Solar-Lezama et al. 2006] depends on the form of examples. We also establish impossibility results for directly applying algorithms from concept learning to invariant inference.

The contributions of the paper are summarized as follows:

- We define the problem of polynomial-length invariant inference, and show it is Σ_2^P-complete (§4), strengthening the hardness result of template-based abstraction by Lahiri and Qadeer [2009].
- We introduce the Hoare-query model, a black-box model of invariant inference capable of modeling PDR (§5), and study the query complexity of polynomial-length invariant inference in this model.
- We show that in general an algorithm in this model requires an exponential number of queries to solve polynomial-length inference, even though Hoare queries are rich and versatile (§6).
- We also extend this result to a model capturing interpolation-based algorithms (§6.2).
- We show that Hoare queries are more powerful than inductiveness queries (§7). This also proves that ICE learning cannot model PDR, and that the extension of the model by Vizel et al. [2017] is necessary.
- We prove that exact learning from counterexamples to induction is harder than exact learning from positive & negative examples, and derive impossibility results for translating some exact concept learning algorithms to the setting of invariant inference (§8).

[1] For the PDR-savvy: β is typically a candidate clause, and α is derived from the previous frame.

2 OVERVIEW

Coming up with inductive invariants is one of the most challenging tasks of formal verification—it is often referred to as the *"Eureka!"* step. This paper studies the asymptotic complexity of automatically inferring CNF invariants of polynomial length, a problem we call ***polynomial-length inductive invariant inference***, in a SAT-based black-box model.

Consider the dilemmas Abby faces when she attempts to develop an algorithm for this problem from first principles. Abby is excited about the popularity of SAT-based inference algorithms. Many such algorithms operate by repeatedly performing checks of Hoare triples of the form $\{\alpha\}\delta\{\beta\}$, where α, β are a precondition and postcondition (resp.) chosen by the algorithm in each query and δ is the given transition relation (loop body). A SAT solver implements the check. We call such checks ***Hoare queries***, and focus in this paper on *black-box* inference algorithms in the ***Hoare-query model***: algorithms that access the transition relation solely through Hoare queries.

Fig. 1 displays one example program that Abby is interested in inferring an inductive invariant for. In this program, a number $\mathbf{x}$, represented by n bits, is initialized to zero, and at each iteration incremented by an even number that is decided by the input variables $\mathbf{y}$ (all computations are mod 2^n). The representation of the number $\mathbf{x}$ using the bits $x_1, \ldots, x_n$ is determined by another set of bits $c_1, \ldots, c_n$, which are all immutable, and only one of them is true: if $c_1 = \mathit{true}$, the number is represented by $x_1, x_2, \ldots, x_n$, if $c_2 = \mathit{true}$ the least-significant bit (lsb) shifts and the representation is $x_2, x_3, \ldots, x_n, x_1$ and so on. The safety property is that $\mathbf{x}$ is never equal to the number with all bits 1. Intuitively, this holds because the number $\mathbf{x}$ is always even. An *inductive invariant* states this fact, taking into account the differing representations, by stating that the lsb (as chosen by $\mathbf{c}$) is always 0: $I = (c_1 \rightarrow \neg x_1) \wedge \ldots (c_n \rightarrow \neg x_n)$. Of course, Abby aims to verify many systems, of which Fig. 1 is but one example.

2.1 Example: Backward-Reachability with Generalization

How should Abby's algorithm go about finding inductive invariants? One known strategy is that of *backward reachability*, in which the invariant is strengthened to exclude states from which bad states may be reachable.[2] Alg. 1 is an algorithmic backward-reachability scheme: it repeatedly checks for the existence of a counterexample to induction (a transition σ, σ' of δ from $\sigma \models I$ to $\sigma' \not\models I$), and strengthens the invariant to exclude the pre-state σ using the formula Block returns.

Alg. 1 depends on the choice of Block. The most basic approach is of Alg. 2, which excludes *exactly* the pre-state, by conjoining to the invariant the negation of the cube of σ (the cube is the conjunction of all literals that hold in the state; the only state that satisfies $\mathit{cube}(\sigma)$ is σ itself, and thus the only one to be excluded from I in this approach). For example, when Alg. 1 needs to block the state $\mathbf{x} = 011\ldots1, \mathbf{c} = 000\ldots1$ (this state reaches the bad state $\mathbf{x} = 111\ldots1, \mathbf{c} = 000\ldots1$), Alg. 2 does so by conjoining to the invariant the negation of $\neg x_n \wedge x_{n-1} \wedge x_{n-2} \wedge \ldots x_1 \wedge \neg c_n \wedge \neg c_{n-1} \wedge \neg c_{n-2} \wedge \ldots c_1$, and this is a formula that other states do not satisfy.

Alas, Alg. 1 with blocking by Alg. 2 is not efficient. In essence it operates by enumerating and excluding the states backward-reachable from bad. The number of such states is potentially exponential, making Alg. 2 unsatisfactory. For instance, the example of Fig. 1 requires the exclusion of all states in which $\mathbf{x}$ is odd for every choice of lsb, a number of states exponential in n. The algorithm would thus require an exponential number of queries to arrive at a (CNF) inductive invariant, even though a CNF invariant with only n clauses exists (as above).

Efficient inference hence requires Abby to exclude more than a single state at each time, namely, to *generalize* from a counterexample—as real algorithms do. What generalization strategy could Abby choose that would lead to efficient invariant inference?

[2]Our results are not specific to backward-reachability algorithms; we use them here for motivation and illustration.

```
1  init  x_1 = ... = x_n = 0
2  axiom ∃!i, 1 ≤ i ≤ n.  c_i = 1
3
4  function add-double(a,b) = (a + 2 · b) mod 2^n
5
6  while *
7      input y_1, ..., y_n
8      if c_1:
9          (x_1, x_2, ..., x_{n-1}, x_n)      := add-double((x_1, x_2, ..., x_{n-1}, x_n),(y_1, y_2, ..., y_{n-1}, y_n))
10     if c_2:
11         (x_2, x_3, ..., x_n, x_1)          := add-double((x_2, x_3, ..., x_n, x_1),(y_2, y_3, ..., y_n, y_1))
12     ...
13     if c_n:
14         (x_n, x_1, ..., x_{n-2}, x_{n-1})  := add-double((x_n, x_1, ..., x_{n-2}, x_{n-1}),(y_n, y_1, ..., y_{n-2}, y_{n-1}))
15     assert ¬(x_1 = ... = x_n = 1)
```

Fig. 1. An example propositional transition system for which we would like to infer an inductive invariant. The state is over $x_1, \ldots, x_n$. The variables $y_1, \ldots, y_n$ are inputs and can change arbitrarily in each step. $c_1, \ldots, c_n$ are immutable, with the assumption that exactly one is true.

Algorithm 1
Backward-reachability

```
1: procedure BLOCK-CUBE(δ)
2:     I ← ¬Bad
3:     while {I}δ{I} not valid do
4:         σ, σ' ← CTI(δ, I)
5:         d ← BLOCK(δ,σ)
6:         I ← I ∧ ¬d
       return I
```

Algorithm 2 Naive Block

```
1: procedure BLOCK-CUBE(δ,σ)
2:     return ⋀_{i, σ⊨p_i} p_i ∧ ⋀_{i, σ⊨¬p_i} ¬p_i
```

Algorithm 3
Generalization with Init-Step Reachability

```
1: procedure BLOCK-PDR-1(δ, σ)
2:     d ← CUBE(σ)
3:     for l ∈ CUBE(σ) do
4:         t ← d \ {l}
5:         if (Init ⟹ ¬t) ∧ {Init}δ{¬t} then
6:             d ← t
       return d
```

2.2 All Generalizations Are Wrong

One simple generalization strategy Abby considers appears in Alg. 3, based on the standard ideas in IC3/PDR [Bradley 2011; Eén et al. 2011] and subsequent developments [e.g. Hoder and Bjørner 2012; Komuravelli et al. 2014]. It starts with the cube (as Alg. 2) and attempts to drop literals, resulting in a smaller conjunction, which many states satisfy; all these states are excluded from the candidate in line 6 of Alg. 1. Hence with this generalization Alg. 1 can exclude many states in each iteration, overcoming the problem with the naive algorithm above. Alg. 3 chooses to drop a literal from the conjunction if no state reachable in at most one step from *Init* satisfies the conjunction even when that literal is omitted (line 4 of Alg. 3); we refer to this algorithm as PDR-1, since it resembles PDR with a single frame.

For example, when in the example of Fig. 1 the algorithm attempts to block the state with $\mathbf{x} = 011\ldots1, \mathbf{c} = 000\ldots1$, Alg. 3 minimizes the cube to $d = x_1 \wedge c_1$, because no state reachable in at most one step satisfies d, but this is no longer true when another literal is omitted. Conjoining the invariant with $\neg d$ (in line 6 of Alg. 1) produces a clause of the invariant, $c_1 \rightarrow \neg x_1$. In fact, our results show that PDR-1 finds the aforementioned invariant in n^2 queries.

Yet there is a risk in over-generalization, that is, of dropping *too many* literals and excluding too many states. In Alg. 1, generalization must not return a formula that some reachable states satisfy, or the candidate I would exclude reachable states and would not be an inductive invariant. Alg. 3 chooses to take the strongest conjunction that does not exclude any state reachable in at most one step; it is of course possible (and plausible) that some states are reachable in two steps but not in one. Alg. 1 with the generalization in Alg. 3 might fail in such cases.

The necessity of generalization, on the one hand, and the problem of over-generalization on the other leads in practice to complex heuristic techniques. Instead of simple backward-reachability with generalization per Alg. 1, PDR never commits to a particular generalization [Eén et al. 2011] through a sequence of *frames*, which are (in some sense) a sequence of candidate invariants. The clauses resulting from generalization are used to strengthen frames according to a bounded reachability analysis; Alg. 3 corresponds to generalization in the first frame.

Overall, the study of backward-reachability and the PDR-1 generalization leaves us with the question: ***Is there a choice of generalization that can be used—in any way—to achieve an efficient invariant inference algorithm?***

In a non-interesting way, the answer is *yes*, there is a "good" way to generalize: Use Alg. 1, with the following generalization strategy: Upon blocking a pre-state σ, compute an inductive invariant of polynomial length, and return the clause of the invariant that excludes σ,[3] and this terminates in a polynomial number of steps.

Such generalization is clearly unattainable. It requires (1) perfect information of the transition system, and (2) solving a computationally hard problem, since we show that polynomial-length inference is Σ_2^P-hard (Thm. 4.2). What happens when generalization is computationally unbounded (an arbitrary function), but operates based on *partial information* of the transition system? Is there a ***generalization from partial information***, be it computationally intractable, that facilitates ***efficient inference***? If such a generalization exists we may wish to view invariant inference heuristics as *approximating* it in a computationally efficient way.

Similar questions arise in interpolation-based algorithms, only that generalization is performed not from a concrete state, but from a bounded unreachability proof. Still it is challenging to generalize enough to make progress but not too much as to exclude reachable states (or include states from which bad is reachable).

2.2.1 Our Results. Our first main result in this paper is that ***in general, there does not exist a generalization scheme from partial information*** leading to efficient inference based on Hoare queries. Technically, we prove that even a computationally unrestricted generalization from information gathered from Hoare queries requires an exponential number of queries. This result applies to any generalization strategy and any algorithm using it that can be modeled using Hoare queries, including Alg. 1 as well as more complex algorithms such as PDR. We also extend this lower bound to a model capturing interpolation-based algorithms (Thm. 6.6).

These results are surprising because a-priori it would seem possible, using unrestricted computational power, to devise queries that repeatedly halve the search space, yielding an invariant with a polynomial number of queries (the number of candidates is only exponential because we are interested in invariants up to polynomial length). We show that this is impossible to achieve using Hoare queries.

2.3 Inference Using Rich Queries

So far we have established strong impossibility results for invariant inference based on Hoare queries in the general case, even with computationally unrestricted generalization. We now turn

[3]Such a clause exists because σ is backward-reachable from bad states, and thus excluded from the invariant.

to shed some light on the techniques that inference algorithms such as PDR employ in practice. One of the fundamental principles of PDR is the incremental construction of invariants relying on rich Hoare queries. PDR-1 demonstrates a simplified realization of this principle. When PDR-1 considers a clause to strengthen the invariant, it checks the reachability of that individual clause from *Init*, rather than the invariant as a whole. This is the Hoare query $\{Init\}\delta\{\neg t\}$ in line 4 of Alg. 3, in which, crucially, the precondition is different from the postcondition. The full-fledged PDR is similar in this regard, strengthening a frame according to reachability from the previous frame via relative induction checks [Bradley 2011].

The algorithm in Alg. 2 is fundamentally different, and uses only inductiveness queries $\{I\}\delta\{I\}$, a specific form of Hoare queries where the precondition and postcondition are the same. Algorithms performing only inductiveness checks can in fact be very sophisticated, traversing the domain of candidates in clever ways. This approach was formulated in the *ICE learning* framework for learning inductive invariants [Garg et al. 2014, 2016] (later extended to general Constrained-Horn Clauses [Ezudheen et al. 2018]), in which algorithms present new candidates based on positive, negative, and implication examples returned by a "teacher" in response to incorrect candidate invariants.[4] The main point is that such algorithms do not perform queries other than inductiveness, and choose the next candidate invariant based solely on the counterexamples to induction showing the previous candidates were unsuitable.

The contrast between the two approaches raises the question: ***Is there a benefit to invariant inference in Hoare queries richer than inductiveness?*** For instance, to model PDR in the ICE framework, Vizel et al. [2017] extended the framework with relative inductiveness checks, but the question whether such an extension is necessary remained open.

2.3.1 Our Results. Our second significant result in this paper is showing an exponential gap between the general Hoare-query model and the more specific inductiveness-query model. To this end, we construct a class of transition systems, including the example of Fig. 1, for which (1) PDR-1, which is a Hoare-query algorithm, infers an invariant in a polynomial number of queries, but (2) *every* inductiveness-query algorithm requires an *exponential* number of queries, that is, an exponential number of candidates before it finds a correct inductive invariant. This demonstrates that analyzing the reachability of clauses separately can offer an exponential advantage in certain cases. This also proves that PDR cannot be cast in the ICE framework, and that the extension by Vizel et al. [2017] is necessary and strictly increases the power of inference with a polynomial number of queries. To the best of our knowledge, this is not only the first lower bound on ICE learning demonstrating such an exponential gap (also see the discussion in §9), but also the first polynomial upper bound on PDR for a class of systems.

We show this separation on a class of systems constructed using a technical notion of *maximal systems for monotone invariants*. These are systems for which there exists a monotone invariant (namely, an invariant propositional variables appear only negatively) with a linear number of clauses, and the transition relation includes *all* transitions allowed by this invariant. The system in Fig. 1 is an example of a maximal system: it allows every transition between states satisfying the invariant (namely, between all even $\mathbf{x}$'s with the same representation), and also every transition between states violating the invariant (namely, between all odd $\mathbf{x}$'s with the same representation).[5] The success of PDR-1 relies on the small diameter (every reachable state is reachable in one step) and the fact that the invariant is monotone. However, we show that for inductiveness-query algorithms this class is as hard as the class of *all* programs admitting monotone invariants, whose hardness is

[4]Our formulation focuses on implication examples—counterexamples to inductiveness queries—and strengthens the algorithm with full information about the set of initial and bad states instead of positive and negative examples (resp.).
[5]Transitions violating the $\mathbf{c}$ axiom or modifying it are excluded in this modeling.

established from the results of §2.2.1. For example, from the perspective of inductiveness-query algorithms, the example of Fig. 1, which is a maximal program as explained above, is as hard as *any* system that admits its invariant (and also respects the **c** axiom and leaves **c** unchanged). This is because an inductiveness-query algorithm can only benefit from having *fewer* transitions and hence fewer counterexamples to induction, whereas maximal programs include as many transitions as possible. If an inductiveness query algorithm is to infer an invariant for the example of Fig. 1, it must also be able to infer an invariant for all systems whose transitions are a *subset* of the transitions of this example. This includes systems with an exponential diameter, as well as systems admitting other invariants, potentially exponentially long. This program illustrates our lower bound construction, which takes *all* maximal programs for monotone-CNF invariants.

In our lower bound we follow the existing literature on the analysis of inductiveness-query algorithms, which focuses on the worst-case notion w.r.t. potential examples (strong convergence in Garg et al. [2014]). An interesting direction is to analyze inductiveness-query algorithms that exercise some control over the choice of counterexamples to induction, or under probabilistic assumptions on the distribution of examples.

2.4 A Different Perspective: Exact Learning of Invariants with Hoare Queries

This paper can be viewed as developing a theory of exact learning of inductive invariants with Hoare queries, akin to the classical theory of concept learning with queries [Angluin 1987]. The results outlined above are consequences of natural questions about this model: The impossibility of generalization from partial information (§2.2.1) stems from an exponential lower bound on the Hoare-query model. The power of rich Hoare queries (§2.3.1) is demonstrated by an exponential separation between the Hoare- and inductiveness-query models, in the spirit of the gap between concept learning using both equivalence and membership queries and concept learning using equivalence queries alone [Angluin 1990].

The similarity between invariant inference (and synthesis in general) and exact concept learning has been observed before [e.g. Alur et al. 2015; Bshouty et al. 2017; Garg et al. 2014; Jha et al. 2010; Jha and Seshia 2017]. Our work highlights some interesting differences and connections between invariant learning with Hoare, and concept learning with equivalence and membership queries. This comparison yields (im)possibility results for translating algorithms from concept learning with queries to invariant inference with queries. Another outcome is the third significant result of this paper: a proof that learning from counterexamples to induction is inherently harder than learning from examples labeled as positive or negative, formally corroborating the intuition advocated by Garg et al. [2014]. More broadly, the complexity difference between learning from labeled examples and learning from counterexamples to induction demonstrates that the convergence rate of learning in Counterexample-Guided Inductive Synthesis [e.g. Jha et al. 2010; Jha and Seshia 2017; Solar-Lezama et al. 2006] depends on the form of examples. The proof of this result builds on the lower bounds discussed earlier, and is discussed in §8.[6]

3 BACKGROUND

3.1 States, Transitions Systems, and Inductive Invariants

In this paper we consider safety problems defined via formulas in propositional logic. Given a propositional vocabulary Σ that consists of a finite set of Boolean variables, we denote by $\mathcal{F}(\Sigma)$ the set of well formed propositional formulas defined over Σ. A *state* is a valuation to Σ. For a

[6] It may also be interesting to note that one potential difference between classical learning and invariant inference, mentioned by Löding et al. [2016], does not seem to manifest in the results discussed in §2.2.1: the transition systems in the lower bound for inductiveness queries in Corollary 7.11 have a *unique* inductive invariant, and still the problem is hard.

state σ, the *cube of* σ, denoted $cube(\sigma)$, is the conjunction of all literals that hold in 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. A class of transition systems, denoted $\mathcal{P}$, is a set of transition systems. 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 $Init \implies I$, $I \wedge \delta \implies I'$, and $I \implies \neg Bad$, where I' denotes the result of substituting each $x \in \Sigma$ for $x' \in \Sigma'$ in I, and $\varphi \implies \psi$ denotes the validity of the formula $\varphi \rightarrow \psi$. In the context of propositional logic, a transition system is safe if and only if it has an inductive invariant. When I is not inductive, a *counterexample to induction* 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').

The classes CNF_n, DNF_n and Mon-CNF_n. CNF_n is the set of propositional formulas in Conjunctive Normal Form (CNF) with at most n clauses (disjunction of literals). DNF_n is likewise for Disjunctive Normal Form (DNF), where n is the maximal number of cubes (conjunctions of literals). Mon-CNF_n is the subset of CNF_n in which all literals are negative.

3.2 Invariant Inference Algorithms

In this section we briefly provide background on inference algorithms that motivate our theoretical development in this paper. The main results of the paper do not depend on familiarity with these algorithms or their details; this (necessarily incomprehensive) "inference landscape" is presented here for context and motivation for defining the Hoare-query model (§5), studying its complexity and the feasibility of generalization (§6), and analyzing the power of Hoare queries compared to inductiveness queries (§7). We allude to specific algorithms in motivating each of these sections.

IC3/PDR. IC3/PDR maintains a sequence of formulas $F_0, F_1, \ldots$, called *frames*, each of which can be understood as a candidate inductive invariant. The sequence is gradually modified and extended throughout the algorithm's run. It is maintained as an approximate reachability sequence, meaning that (1) $Init \implies F_0$, (2) $F_j \implies F_{j+1}$, (3) $F_j \wedge \delta \implies (F_{j+1})'$, and (4) $F_j \implies \neg Bad$. These properties ensure that F_j overapproximates the set of states reachable in j steps, and that the approximations contain no bad states. (We emphasize that $F_j \implies \neg Bad$ does *not* imply that a bad state is unreachable in *any* number of states.) The algorithm terminates when one of the frames implies its preceding frame ($F_j \implies F_{j-1}$), in which case it constitutes an inductive invariant, or when a counterexample trace is found. In iteration N, a new frame F_N is added to the sequence. One way of doing so is by initializing F_N to *true*, and strengthening it until it excludes all bad states. Strengthening is done by *blocking* bad states: given a bad state $\sigma_b \models F_N \wedge Bad$, the algorithm strengthens F_{N-1} to exclude all σ_b's pre-states—states that satisfy $F_{N-1} \wedge \delta \wedge (cube(\sigma_b))'$—one by one (thereby demonstrating that σ_b is unreachable in N steps). Blocking a pre-state σ_a from frame $N-1$ is performed by a recursive call to block its own pre-states from frame $N-2$, and so on. If this process reaches a state from *Init*, the sequence of states from the recursive calls constitutes a trace reaching *Bad* from *Init*, which is a counterexample to safety. Alternatively, when a state σ is successfully found to be unreachable from F_{j-1} in one step, i.e., $F_{j-1} \wedge \delta \wedge (cube(\sigma_b))'$ is unsatisfiable, frame F_j is strengthened to reflect this fact. Aiming for efficient convergence (see §2.1), PDR chooses to *generalize*, and exclude more states. A basic form of generalization is performed by dropping literals from $cube(\sigma)$ as long as the result t is still unreachable from F_{j-1}, i.e., $F_{j-1} \wedge \delta \wedge t'$ is still unsatisfiable. This is very similar to PDR-1 above (§2.2), where F_{j-1} was always $F_0 = Init$. Often *inductive generalization* is used, dropping literals as long as $F_{j-1} \wedge \neg t \wedge \delta \wedge t'$, reading that $\neg t$ is inductive *relative* to F_{j-1}, which can drop more literals than basic generalization. A core optimization of PDR is *pushing*, in which a

frame F_j is "opportunistically" strengthened with a clause α from F_{j-1}, if F_{j-1} is already sufficiently strong to show that α is unreachable in F_j.

For a more complete presentation of PDR and its variants as a set of abstract rules that may be applied nondeterministically see e.g. Gurfinkel and Ivrii [2015]; Hoder and Bjørner [2012]. The key point from the perspective of this paper is that the algorithm and its variants access the transition relation δ in a very specific way, checking whether some α is unreachable in one step of δ from the set of states satisfying a formula F (or those satisfying $F \wedge \alpha$), and obtains a counterexample when it is reachable (see also Vizel et al. [2017]). Crucially, other operations (e.g., maintaining the frames, checking whether $F_j \implies F_{j-1}$, etc.) do not use δ. We will return to this point when discussing the Hoare-query model, which can capture IC3/PDR (§5).

ICE. The ICE framework [Garg et al. 2014, 2016] (later extended to general Constrained-Horn Clauses [Ezudheen et al. 2018]), is a learning framework for inferring invariants from positive, negative and implication counterexamples. We now review the framework using the original terminology and notation; later in the paper we will use a related formulation that emphasizes the choice of candidates (in §7.1).

In ICE learning, the teacher holds an unknown target (P, N, R), where $P, N \subseteq D, R \subseteq D \times D$ are sets of examples. The learner's goal is to find a hypothesis $H \in C$ s.t. $P \subseteq H, N \cap H = \emptyset$, and for each $(x, y) \in R$, $x \in H \implies y \in H$. The natural way to cast inference in this framework is, given a transition system $(\mathit{Init}, \delta, \mathit{Bad})$ and a set of candidate invariants $\mathcal{L}$, to take D as the set of program states, P a set of reachable states including Init, N a set of states including Bad from which a safety violation is reachable, R the set of transitions of δ, and $C = \mathcal{L}$. Iterative ICE learning operates in rounds. In each round, the learner is provided with a sample—(E, B, I) s.t. $E \subseteq P, B \subseteq N, I \subseteq R$—and outputs an hypothesis $H \in C$. The teacher returns that the hypothesis is correct, or extends the sample with an example showing that H is incorrect. The importance of implication counterexamples is that they allow implementing a teacher using a SAT/SMT solver without "guessing" what a counterexample to induction indicates [Garg et al. 2014; Löding et al. 2016]. Examples of ICE learning algorithms include Houdini [Flanagan and Leino 2001] and symbolic abstraction [Reps et al. 2004; Thakur et al. 2015], as well as designated algorithms [Garg et al. 2014, 2016]. Theoretically, the analysis of Garg et al. [2014] focuses on *strong convergence* of the learner, namely, that the learner can always reach a correct concept, no matter how the teacher chooses to extend samples between rounds. In this work, we will be interested in the *number of rounds* the learner performs. We will say that the learner is *strongly-convergent* with *round-complexity* r if for every ICE teacher, the learner finds a correct hypothesis in at most r rounds, provided that one exists. We extend this definition to a class of target descriptions in the natural way.

Interpolation. The idea of interpolation-based algorithms, first introduced by McMillan [2003], is to generalize proofs of bounded unreachability into elements of a proof of *un*bounded reachability, utilizing *Craig interpolation*. Briefly, this works as follows: encode a bounded reachability from a set of states F in k steps, and use a SAT solver to find that this cannot reach Bad. When efficient interpolation is supported in the logic and solver, the SAT solver can produce an interpolant C: a formula representing a set of states that (i) overapproximates the set of states reachable from F in k_1 steps, and still (ii) cannot reach Bad in k_2 steps (any choice $k_1 + k_2 = k$ is possible). Thus C overapproximates concrete reachability from F without reaching a bad state, although both these facts are known in only a bounded number of steps. The hope is that C would be a useful generalization to include as part of the invariant. The original algorithm [McMillan 2003] sets some k as the current unrolling bound, starts with $F = \mathit{Init}$, obtains an interpolant C with $k_1 = 1, k_2 = k-1$, sets $F \leftarrow F \vee C$ and continues in this fashion, until an inductive invariant is found, or Bad becomes reachable in k steps from F, in which case k is incremented and the algorithm is restarted. The use

of interpolation and generalization from bounded unreachability has been used in many works since [e.g. Henzinger et al. 2004; Jhala and McMillan 2007; McMillan 2006; Vizel and Grumberg 2009; Vizel et al. 2013]. Combining ideas from interpolation and PDR has also been studied [e.g. Vizel and Gurfinkel 2014]. The important point for this paper is that many interpolation-based algorithms only access the transition relation when checking bounded reachability (from some set of states α to some set of states β), and extracting interpolants when the result is unreachable. We will return to this point when discussing the interpolation-query model, which aims to capture interpolation-based algorithms (§6.2).

4 POLYNOMIAL-LENGTH INVARIANT INFERENCE

In this section we formally define the problem of *polynomial-length invariant inference* for CNF formulas, which is the focus of this paper. We then relate the problem to the problem of inferring DNF formulas with polynomially many cubes via duality (see the extended version [Feldman et al. 2020]), and focus on the case of CNF in the rest of the paper.

Our object of study is the problem of polynomial-length inference:

Definition 4.1 (Polynomial-Length Inductive Invariant Inference). The *polynomial-length inductive invariant inference problem* (*invariant inference* for short) for a class of transition systems $\mathcal{P}$ and a polynomial $p(n) = \Omega(n)$ is the problem: Given a transition system $TS \in \mathcal{P}$ over Σ, *decide* whether there exists an inductive invariant $I \in \mathrm{CNF}_{p(n)}$ for TS, where $n = |\Sigma|$.

Notation. In the sequel, when considering the polynomial-length inductive invariant inference problem of a transition system $TS = (Init, \delta, Bad) \in \mathcal{P}$, we denote by Σ the vocabulary of $Init$, Bad and δ. Further, we denote $n = |\Sigma|$.

Complexity. The *complexity* of polynomial-length inference is measured in $|TS| = |Init|+|\delta|+|Bad|$. Note that the invariants are required to be polynomial in $n = |\Sigma|$.

$\mathrm{CNF}_{p(n)}$ is a rich class of invariants. Inference in more restricted classes can be solved efficiently. For example, when only conjunctive candidate invariants are considered, and $\mathcal{P}$ is the set of all propositional transition systems, the problem can be decided in a polynomial number of SAT queries through the Houdini algorithm [Flanagan and Leino 2001; Lahiri and Qadeer 2009]. Similar results hold also for CNF formulas with a constant number of literals per clause (by defining a new predicate for each of the polynomially-many possible clauses and applying Houdini), and for CNF formulas with a constant number of clauses (by translating them to DNF formulas with a constant number of literals per cube and applying the dual procedure). However, a restricted class of invariants may miss invariants for some programs and reduces the generality of the verification procedure. Hence in this paper we are interested in the richer class of polynomially-long CNF invariants. In this case the problem is no longer tractable even with a SAT solver:

Theorem 4.2. *Let $\mathcal{P}$ be the set of all propositional transition systems. Then polynomial-length inference for $\mathcal{P}$ is Σ_2^P-complete, where $\Sigma_2^P = NP^{SAT}$ is the second level of the polynomial-time hierarchy.*

We defer the proof to §6.1.1.

We note that polynomial-length inference can be encoded as specific instances of template-based inference; the Σ_2^P-hardness proof of Lahiri and Qadeer [2009] uses more general templates and therefore does not directly imply the Σ_2^P-hardness of polynomial-length inference. Lower bounds on polynomial-length inference entail lower bounds for template-based inference.

Remark 4.1. *In the above formulation, an efficient procedure for deciding* safety *does not imply polynomial-length inference is tractable, since the program may be safe, but all inductive invariants*

may be too long. To overcome this technical quirk, we can consider a promise problem *[Goldreich 2006] variant of polynomial-length inference:*

Given a transition system $TS \in \mathcal{P}$,

- ***(Completeness)** If TS has an inductive invariant $I \in CNF_{p(n)}$, the algorithm must return* yes.
- ***(Soundness)** If TS is not safe the algorithm must return* no.

Other cases, including the case of safety with an invariant outside $CNF_{p(n)}$, are not constrained. An algorithm deciding safety thus solves also this problem. All the results of this paper apply both to the standard version above and the promise problem: upper bounds on the standard version trivially imply upper bounds on the promise problem, and in our lower bounds we use transition systems that are either (i) safe and *have an invariant in $CNF_{p(n)}$, or (ii) unsafe.*

5 INVARIANT INFERENCE WITH QUERIES AND THE HOARE QUERY MODEL

In this paper we study algorithms for polynomial-length inference through *black-box* models of *inference with queries*. In this setting, the algorithm accesses the transition relation through (rich) queries, but cannot read the transition relation directly. Our main model is of *Hoare-query algorithms*, which query the validity of a postcondition from a precondition in one step of the system. Hoare-query algorithms faithfully capture a large class of SAT-based invariant inference algorithms, including PDR and related methods.

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 computational complexity of SAT-based algorithms captured by the model. Such an information-based analysis is also necessary for questions involving unbounded computational power and restricted information, in the context of computationally-unrestricted bounded-reachability generalization (see §6.3).

In this section we define the basic notions of queries and query-based inference algorithms. We also define the primary query model we study in the paper: the Hoare-query model. In the subsequent sections we introduce and study additional query models: the interpolation-query model (§6.2), and the inductiveness-query model (§7.1).

Inference with queries. We model queries of the transition relation 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 $\boldsymbol{Q}$ is a set of query oracles with the same interface. We consider several different query oracles, representing different ways of obtaining information about the transition relation.

Definition 5.1 (Inference algorithm in the query model). An *inference algorithm from queries*, denoted $\mathcal{A}^{Q}(Init, Bad, [\delta])$, is defined w.r.t. a query oracle Q and is 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.

$\mathcal{A}^{Q}(Init, Bad, [\delta])$ solves the problem of polynomial-length invariant inference for $(Init, \delta, Bad)$.

The Hoare-query model. Our main object of study in this paper is the Hoare-query model of invariant inference algorithms. It captures SAT-based invariant inference algorithms querying the behavior of a single step of the transition relation at a time.

Definition 5.2 (Hoare-Query Model). For a transition relation δ and input formulas $\alpha, \beta \in \mathcal{F}(\Sigma)$, the *Hoare-query oracle*, $\mathcal{H}(\delta, \alpha, \beta)$, returns *false* if $(\alpha \wedge \delta \wedge \neg\beta') \in \text{SAT}$; otherwise it returns *true*.

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

Intuitively, a Hoare-query algorithm gains access to the transition relation, δ, exclusively by repeatedly choosing $\alpha, \beta \in \mathcal{F}(\Sigma)$, and calling $\mathcal{H}(\delta, \alpha, \beta)$.

If we are using a SAT solver to compute the Hoare-query, $\mathcal{H}(\delta, \alpha, \beta)$, then when the answer is *false*, the SAT solver will also produce a counterexample pair of states σ, σ' such that $\sigma, \sigma' \models \alpha \wedge \delta \wedge \neg\beta'$.

We observe that using binary search, a Hoare-query algorithm can do the same:

Lemma 5.3. *Whenever $\mathcal{H}(\delta, \alpha, \beta) = \text{false}$, a Hoare-query algorithm can find σ, σ' such that $\sigma, \sigma' \models \alpha \wedge \delta \wedge \neg\beta'$ using $n = |\Sigma|$ Hoare queries.*

Proof. For each $x_i \in \Sigma \uplus \Sigma'$, if $x_i \in \Sigma$, conjoin it to α, else to β, and check whether $\mathcal{H}(\delta, \alpha_i, \beta_i)$ is still *false*. If it is, continue to x_{i+1}; otherwise flip x_i and continue to x_{i+1}. □

Example: PDR as a Hoare-query algorithm. The Hoare-query model captures the prominent PDR algorithm, facilitating its theoretical analysis. As discussed in §3.2, 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 $\mathcal{H}(\delta, F, \alpha)$ or $\mathcal{H}(\delta, F \wedge \alpha, \alpha)$ (for the algorithm's choice of $F, \alpha \in \mathcal{F}(\Sigma)$), and obtaining a counterexample using a polynomial number of Hoare queries, if one exists (Lemma 5.3). Furthemore, 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.

The Hoare-query model is not specific to PDR. It also captures algorithms in the ICE learning model [Garg et al. 2014], as we discuss in §7.1, and as result can model algorithms captured by the ICE model (see §3.2). In §7.2 we show that the Hoare-query model is in fact strictly more powerful than the ICE model.

Remark 5.1. *Previous black-box models for invariant inference [Garg et al. 2014] encapsulated access also to Init, Bad. In our model we encapsulate only access to δ, since (1) it is technically simpler, (2) a simple transformation can make Init, Bad uniform across all programs, embedding the differences in the transition relation; indeed, our constructions of classes of transition systems in this paper are such that Init, Bad are the same in all transition systems that share a vocabulary, hence Init, Bad may be inferred from the vocabulary. (Unrestricted access to Init, Bad is stronger, thus lower bounds on our models apply also to models restricting access.)*

Complexity. Focusing on *information*, we do not impose computational restrictions on the algorithms, and only count the number of queries the algorithm performs to reveal information of the transition relation. In particular, when establishing lower bounds on the query complexity, we even consider algorithms that may compute non-computable functions. However, whenever we construct algorithms demonstrating upper bounds on query complexity, these algorithms in fact have polynomial *time* complexity, and we note this when relevant.

Given a query oracle and an inference algorithm that uses it, we analyze the number of queries the algorithm performs as a function of $n = |\Sigma|$, in a *worst-case* model w.r.t. to possible transition systems over Σ in the class of interest.

The definition is slightly more complicated by considering, as we do later in the paper, query-models in which more than one oracle exists, i.e., an algorithm may use any oracle from a *family* of query oracles. In this case, we analyze the query complexity of an algorithm in a *worst-case* model w.r.t. the possible query oracles in the family as well.

Formally, the query complexity is defined as follows:

Definition 5.4 (Query Complexity). For a class of transitions systems $\mathcal{P}$, the *query complexity* of (a possibly computationally unrestricted) $\mathcal{A}$ w.r.t. a query oracle family $\mathcal{Q}$ is defined as

$$q^{\mathcal{Q}}_{\mathcal{A}}(n) = \sup_{Q \in \mathcal{Q}} \sup_{(Init, \delta, Bad) \in \mathcal{P}, \atop |\Sigma| = n} \#\text{query}(\mathcal{A}^{Q}(Init, Bad, [\delta])) \tag{1}$$

where $\#\text{query}(\mathcal{A}^{Q}(Init, Bad, [\delta]))$ is the number of times the algorithm accesses Q given this oracle and the input. (These numbers might be infinite.)

The query complexity in the Hoare-query model is $q^{\{\mathcal{H}\}}_{\mathcal{A}}(n)$.

Remark 5.2. *In our definition, query complexity is a function of the size of the vocabulary $n = |\Sigma|$, but not of the size of the representation of the transition relation $|\delta|$. This reflects the fact that an algorithm in the black-box model does not access δ directly. In the extended version [Feldman et al. 2020] we discuss the complexity w.r.t. $|\delta|$ as well. The drawback of such a complexity measure is that learning δ itself becomes feasible, undermining the black-box model. Efficiently learning δ is possible when using unlimited computational power and exponentially-long queries. However, whether the same holds when using unlimited computational power with only polynomially-long queries is related to open problems in classical concept learning.*

6 THE INFORMATION COMPLEXITY OF HOARE-QUERY ALGORITHMS

In this section we prove an information-based lower bound on Hoare-query invariant inference algorithms, and also extend the results to algorithms using *interpolation*, another SAT-based operation. We then apply these results to study the role of information in generalization as part of inference algorithms.

6.1 Information Lower Bound for Hoare-Query Inference

We show that a Hoare-query inference algorithm requires $2^{\Omega(n)}$ Hoare-queries in the worst case to decide whether a CNF invariant of length polynomial in n exists. (Recall that n is a shorthand for $|\Sigma|$, the size of the vocabulary of the input transition system.) This result applies even when allowing the choice of queries to be inefficient, and when allowing the queries to use exponentially-long formulas. It provides a lower bound on the time complexity of actual algorithms, such as PDR, that are captured by the model. Formally:

Theorem 6.1. *Every Hoare-query inference algorithm $\mathcal{A}^{\mathcal{H}}$ deciding polynomial-length inference for the class of all propositional transition systems has query complexity of $2^{\Omega(n)}$.*

The rest of this section proves a strengthening of this theorem, for a specific class of transition systems (which we construct next), for any class of invariants that includes monotone CNF, and for computationally-unrestricted algorithms:

Theorem 6.2. *Every Hoare-query inference algorithm $\mathcal{A}^{\mathcal{H}}$, even computationally-unrestricted, deciding invariant inference for the class of transition systems $\mathcal{P}_{\Sigma^P_2}$ (§6.1.1) and for any class of target invariants $\mathcal{L}$ s.t. Mon-CNF$_n \subseteq \mathcal{L}$[7] has query complexity of $2^{\Omega(n)}$.*

[7]Here we extend the definition of polynomial-length invariant inference to $\mathcal{L}$ instead of $\text{CNF}_{p(n)}$.

(That classes containing Mon-CNF$_n$ are already hard becomes important in §7.)

6.1.1 A Hard Class of Transition Systems. In this section we construct a $\mathcal{P}_{\Sigma_2^P}$, a hard class of transition systems, on which we prove hardness results.

The QBF$_2$ *problem.* The construction of $\mathcal{P}_{\Sigma_2^P}$ follows the Σ_2^P-complete problem of QBF$_2$ from classical computational complexity theory. In this problem, the input is a quantified Boolean formula $\exists\overline{y}.\,\forall\overline{x}.\,\phi(\overline{x},\overline{y})$ where ϕ is a Boolean (quantifier-free) formula, and the problem of QBF$_2$ is to decide whether the quantified formula is *true*, namely, there exists a Boolean assignment to $\overline{y}$ s.t. $\phi(\overline{x},\overline{y})$ is true for every Boolean assignment to $\overline{x}$.

The class $\mathcal{P}_{\Sigma_2^P}$. For each $k \in \mathbb{N}$, we define $\mathcal{P}^k_{\Sigma_2^P}$. Finally, $\mathcal{P}_{\Sigma_2^P} = \bigcup_{k\in\mathbb{N}} \mathcal{P}^k_{\Sigma_2^P}$.

Let $k \in \mathbb{N}$. For each formula $\exists\overline{y}.\,\forall\overline{x}.\,\phi(\overline{y},\overline{x})$, where $\overline{y} = y_1,\ldots,y_k$, $\overline{x} = x_1,\ldots,x_k$ are variables and ϕ is a quantifier-free formula over the variables $\overline{x}\cup\overline{y}$, we define a transition system $TS^\phi = (Init_k, \delta^P_\phi, Bad_k)$. Intuitively, it iterates through $\overline{y}$ lexicographically, and for each $\overline{y}$ it iterates lexicographically through $\overline{x}$ and checks if all assignments to $\overline{x}$ satisfy $\phi(\overline{y},\overline{x})$. If no such $\overline{y}$ is found, this is an error. More formally,

(1) $\Sigma_k = \{y_1,\ldots,y_k,x_1,\ldots,x_k,a,b,e\}$.
(2) $Init_k = \overline{y} = \overline{0} \wedge \overline{x} = \overline{0} \wedge \neg a \wedge b \wedge \neg e$.
(3) $Bad_k = e$.
(4) δ^P_ϕ: evaluate $\phi(\overline{y},\overline{x})$, and perform the following changes (at a single step): If the result is false, set a to true. If $\overline{x} = \overline{1}$ and a is still false, set b to false. If in the pre-state $\overline{x} = \overline{1}$, increment $\overline{y}$ lexicographically, reset a to false, and set $\overline{x} = 0$; otherwise increment $\overline{x}$ lexicographically. If in the pre-state $\overline{y} = \overline{1}$, set e to b. (Intuitively, a is false as long as no falsifying assignment to $\overline{x}$ has been encountered for the current $\overline{y}$, b is true as long as we have not yet encountered a $\overline{y}$ for which there is no falsifying assignment.)

We denote the resulting class of transition systems $\mathcal{P}^k_{\Sigma_2^P} = \{TS^\phi \mid \phi = \phi(y_1,\ldots,y_k,x_1,\ldots,x_k)\}$.

The following lemma relates the QBF$_2$ problem for ϕ to the inference problem of TS^ϕ:

Lemma 6.3. *Let $TS^\phi \in \mathcal{P}^k_{\Sigma_2^P}$. Then TS^ϕ is safe iff it has an inductive invariant in Mon-CNF$_{2k+1}$ iff the formula $\exists\overline{y}.\,\forall\overline{x}.\,\phi(\overline{y},\overline{x})$ is true.*

Proof. There are two cases:

- If $\exists\overline{y}.\,\forall\overline{x}.\,\phi(\overline{y},\overline{x})$ is true, let $\overline{v}$ be the first valuation for $\overline{y}$ that realizes the existential quantifiers. Then the following is an inductive invariant for TS^ϕ:

$$I = \neg e \wedge (b \rightarrow \overline{y} \le \overline{v}) \wedge ((b \wedge a) \rightarrow \overline{y} < \overline{v}) \tag{2}$$

  where the lexicographic constraint is expressed by the following recursive definition on $\overline{y}_{[d]} = (y_1,\ldots,y_d), \overline{v}_{[d]} = (v_1,\ldots,v_d)$:

$$\overline{y}_{[d]} < \overline{v}_{[d]} \stackrel{\text{def}}{=} \begin{cases} \neg y_d \vee (\overline{y}_{[d-1]} < \overline{v}_{[d-1]}) & v_d = \mathit{true} \\ \neg y_d \wedge (\overline{y}_{[d-1]} < \overline{v}_{[d-1]}) & v_d = \mathit{false} \end{cases}$$

  and $\overline{y} \le \overline{v} \stackrel{\triangle}{=} \overline{y} < (\overline{v}+1)$ (or *true* if $\overline{v} = \overline{1}$).
  $I \in$ Mon-CNF$_{2k+1}$: Note that $\overline{y}_{[k]} < \overline{v}_{[k]}$ can be written in CNF with at most n clauses: in the first case a literal is added to each clause, and in the second another clause is added. Thus I can be written in CNF with at most $2k+1$ clauses. Further, the literals of $\overline{y}$ appear only negatively in $\overline{y}_{[k]} < \overline{v}_{[k]}$, and hence also in I. The other literals ($\neg e, \neg a, \neg b$) also appear only negatively in I. Hence, I is monotone.

The proof that I is indeed inductive appears in the extended version [Feldman et al. 2020].

- If $\exists \overline{y}.\, \forall \overline{x}.\, \phi(\overline{y}, \overline{x})$ is not true, then TS^{ϕ} is not safe (and thus does not have an inductive invariant of any length). This is because for every valuation of $\overline{y}$ a violating $\overline{x}$ is found, turning a to true, and b never turns to false, so after iterating through all possible $\overline{y}$'s e will become true.

□

Before we turn to prove Thm. 6.2 and establish a lower bound on the query complexity in the Hoare model, we note that this construction also yields the computational hardness mentioned in §4:

Proof of Thm. 4.2. The upper bound is straightforward: guess an invariant in $\mathrm{CNF}_{p(n)}$ and check it. For the lower bound, use the reduction outlined above: given $\phi(y_1, \ldots, y_k, x_1, \ldots, x_k)$, construct TS^{ϕ}. Note that the vocabulary size, n, is $2k + 3$, and the invariant, when exists, is of length at most $2k + 1 \leq n$.[8] The reduction is polynomial as the construction of TS^{ϕ} (and n) from ϕ is polynomial in k and $|\phi|$: note that lexicographic incrementation can be performed with a propositional formula of polynomial size. □

6.1.2 Lower Bound's Proof. We now turn to prove Thm. 6.2. Given an algorithm with polynomial query complexity, the proof constructs two transition system: one that has a polynomial-length invariant and one that does not, and yet all the queries the algorithm performs do not distinguish between them. The construction uses the path the algorithm takes when all Hoare queries return *false* as much as possible. Intuitively, such responses are less informative and rule out less transition relations, because they merely indicate the existence of a single counterexample to a Hoare triple, opposed to the result *true* which indicates that *all* transitions satisfy a property.

Proof of Thm. 6.2. Let $\mathcal{A}$ be a computationally unbounded Hoare-query algorithm. We show that the number of Hoare queries performed by $\mathcal{A}$ on transition systems from $\mathcal{P}_{\Sigma_2^P}$ with $|\Sigma| = n$ is at least $2^{\frac{n-1}{2}}$. To this end, we show that if $\mathcal{A}$ over $|\Sigma| = 2n + 3$ performs less than 2^n queries, then there exist two formulas ψ_1, ψ_2 over $y_1, \ldots, y_n, x_1, \ldots, x_n$ such that

- all the Hoare queries performed by $\mathcal{A}$ on $\delta^P_{\psi_1}$ and $\delta^P_{\psi_2}$ (the transition relations of TS^{ψ_1} and TS^{ψ_2}, respectively) return the same result, even though
- $\mathcal{A}$ should return different results when run on $TS^{\psi_1} \in \mathcal{P}^n_{\Sigma_2^P}$ and $TS^{\psi_2} \in \mathcal{P}^n_{\Sigma_2^P}$, since TS^{ψ_1} has an invariant in Mon-CNF_{2n+1} and TS^{ψ_2} does not have an invariant (of any length).

We begin with some notation. Running on input TS^{ϕ}, we abbreviate $\mathcal{H}(\delta^P_{\phi}, \cdot, \cdot)$ by $\mathcal{H}(\phi, \cdot, \cdot)$. Denote the queries $\mathcal{A}$ performs and their results by $\mathcal{H}(\phi, \alpha_1, \beta_1) = b_1, \ldots, \mathcal{H}(\phi, \alpha_m, \beta_m) = b_m$. We call an index i *sat* if $b_i = \mathit{false}$. We say that ψ query-agrees with ϕ if $\mathcal{H}(\psi, \alpha_i, \beta_i) = b_i$ for all i. We say that ψ sat-query-agrees with ϕ if for every i such that $b_i = \mathit{false}$ it holds that $\mathcal{H}(\psi, \alpha_i, \beta_i) = \mathit{false}$.

We first find a formula ϕ over $y_1, \ldots, y_n, x_1, \ldots, x_n$ such that the sequence of queries $\mathcal{A}$ performs when executing on TS^{ϕ} is *maximally satisfiable*: if ψ sat-query-agrees with ϕ, then ψ (completely) query-agrees with ϕ on the queries, that is,

$$\forall \psi.\ \ (\forall i.\ b_i = \mathit{false} \Rightarrow \mathcal{H}(\psi, \alpha_i, \beta_i) = b_i) \Longrightarrow (\forall i.\ \mathcal{H}(\psi, \alpha_i, \beta_i) = b_i) \tag{3}$$

We construct this sequence iteratively (and define ϕ accordingly) by always taking ϕ so that the result of the next query is *false* as long as this is possible while remaining consistent with the results of the previous queries: Initially, choose some arbitrary ϕ^0. At each point i, consider the first i queries $\mathcal{A}$ performs on ϕ^i, $\mathcal{H}(\phi^i, \alpha_1, \beta_1) = b_1, \ldots, \mathcal{H}(\phi^i, \alpha_i, \beta_i) = b_i$. If $\mathcal{A}$ terminates without

[8] For an arbitrary polynomial $p(n) = \Omega(n)$, e.g., $p(n) = c \cdot n$ with $0 < c < 1$, enlarge Σ, e.g., by adding to *Init* initialization of fresh variables that are not used elsewhere, to ensure existence of an invariant of length $\leq p(n)$.

performing another query, we are done: the desired ϕ is ϕ_i. Otherwise let $(\alpha_{i+1}, \beta_{i+1})$ be the next query. Amongst formulas ϕ^{i+1} that query-agree on the first i queries, namely, $\mathcal{H}(\phi^{i+1}, \alpha_j, \beta_j) = b_j$ for all $j \leq i$, choose one such that $\mathcal{H}(\phi^{i+1}, \alpha_{i+1}, \beta_{i+1}) = \mathit{false}$ if possible; if such ϕ^{i+1} does not exist take e.g. $\phi^{i+1} = \phi^i$. The dependency of $\mathcal{A}$ on ϕ^i is solely through the results of the queries to $\mathcal{H}(\delta^P_\phi, \cdot, \cdot)$, so $\mathcal{A}$ performs the same i initial queries when given ϕ^{i+1}. The result is a maximally satisfiable sequence, for if a formula ψ differs in query $i+1$ in which the result is *false* instead of *true* we would have taken such a ψ as ϕ^{i+1}.

Let ϕ be such a formula with a maximally satisfiable sequence of queries $\mathcal{A}$ performs on ϕ, $\mathcal{H}(\phi, \alpha_1, \beta_1) = b_1, \ldots, \mathcal{H}(\phi, \alpha_m, \beta_m) = b_m$. For every sat i, take a counterexample $\sigma_i, \sigma'_i \models \alpha_i \wedge \delta^P_\phi \wedge \neg\beta'_i$. The single transition (σ_i, σ'_i) of δ^P_ϕ depends on the value of ϕ on at most one assignment to $\overline{x}, \overline{y}$, so there exists a valuation $v_i : \overline{y} \cup \overline{x} \to \{\mathit{true}, \mathit{false}\}$ such that

$$\forall \psi.\ \ \psi(v_i) = \phi(v_i) \Longrightarrow \sigma_i, \sigma'_i \models \alpha_i \wedge \delta^P_\psi \wedge \neg\beta'_i \tag{4}$$

as well. It follows that

$$\forall \psi.\ \ \psi(v_i) = \phi(v_i) \Longrightarrow \mathcal{H}(\psi, \alpha_i, \beta_i) = \mathcal{H}(\phi, \alpha_i, \beta_i) = \mathit{false}. \tag{5}$$

Let $v_{i_1}, \ldots, v_{i_t}$ be the valuations derived from the sat queries (concerning indexing, $b_i = \mathit{false}$ iff $b_i = b_{i_j}$ for some j). We say that a formula ψ *valuation-agrees* with ϕ on $v_{i_1}, \ldots, v_{i_t}$ if $\psi(v_{i_j}) = \phi(v_{i_j})$ *for all* j's. Since the sequence of queries is maximally satisfiable, if ψ valuation-agrees with ϕ on $v_{i_1}, \ldots, v_{i_t}$ then ψ query-agrees with ϕ, namely, $\mathcal{H}(\psi, \alpha_i, \beta_i) = \mathcal{H}(\phi, \alpha_i, \beta_i)$ for *all* $i = 1, \ldots, m$. As the dependency of $\mathcal{A}$ on ϕ is solely through the results $b_1, \ldots, b_m$, it follows that $\mathcal{A}$ performs the same queries on ψ as it does on ϕ and returns the same answer.

It remains to argue that if $m < 2^n$ then there exist two formulas ψ_1, ψ_2 that valuation-agree with ϕ on $v_{i_1}, \ldots, v_{i_t}$ but differ in the correct result $\mathcal{A}$ should return: $\exists\overline{y}.\ \forall\overline{x}.\ \psi_1(\overline{y}, \overline{x})$ is true, and so TS^{ψ_1} has an invariant in Mon-CNF$_{2n+1}$ (Lemma 6.3), whereas $\exists\overline{y}.\ \forall\overline{x}.\ \psi_2(\overline{y}, \overline{x})$ is not, and so TS^{ψ_2} does not have an invariant of any length or form (Lemma 6.3). This is possible because the number of constraints imposed by valuation-agreeing with ϕ on $v_{i_1}, \ldots, v_{i_t}$ is less than the number of possible valuations of $\overline{x}$ for every valuation of $\overline{y}$ and vice versa:

$$\psi_1(\overline{y}, \overline{x}) = \bigwedge_{\substack{i=1..t \\ \theta(v_{i_j})=\mathit{false}}} (\overline{y}, \overline{x}) \neq v_{i_j} \tag{6}$$

is true on all valuations except for some of $v_{i_1}, \ldots, v_{i_t}$, and since $t \leq m < 2^n$ there exists some $\overline{y}$ such that for all $\overline{x}$, $(\overline{y}, \overline{x})$ is not one of these valuations (recall that $|y| = n$ bits). Dually,

$$\psi_2(\overline{y}, \overline{x}) = \bigvee_{\substack{i=1..t \\ \theta(v_{i_j})=\mathit{true}}} (\overline{y}, \overline{x}) = v_{i_j} \tag{7}$$

is false on all valuations except for some of $v_{i_1}, \ldots, v_{i_t}$, and since $t \leq m < 2^n$ for every $\overline{y}$ there exists $\overline{x}$ such that $(\overline{y}, \overline{x})$ is not one of these valuations (recall that $|y| = n$ bits). This concludes the proof. □

6.2 Extension to Interpolation-Based Algorithms

We now consider inference algorithms based on *interpolation*, another operation supported by SAT solvers. Interpolation has been introduced to invariant inference by McMillan [2003], and since extended in many works (see §3.2).

Interpolation algorithms infer invariants from facts obtained with Bounded Model Checking (BMC), which we formalize as follows:

Definition 6.4 (Bounded Reachabilitiy Check). The *k-bounded reachability check* returns

$$\mathcal{H}^{(k)}(\delta, \alpha, \beta) \stackrel{\text{def}}{=} \alpha(\Sigma^0) \wedge \delta(\Sigma^0, \Sigma^1) \wedge \ldots \wedge \delta(\Sigma^{k-1}, \Sigma^k) \Longrightarrow \beta(\Sigma^k) \tag{8}$$

for $\alpha, \beta \in \mathcal{F}(\Sigma)$, where $\Sigma^0, \ldots, \Sigma^k$ are $k+1$ distinct copies of the vocabulary.

Definition 6.5 (Interpolation-Query Model). An *interpolation-query oracle* is a query oracle Q such that for every $\delta, \alpha, \beta \in \mathcal{F}(\Sigma)$, and $k_1, k_2 \in \mathbb{N}$,

- $Q^{(k_1,k_2)}(\delta, \alpha, \beta) = \bot$ if $\mathcal{H}^{(k_1+k_2)}(\delta, \alpha, \beta) = \mathit{false}$, and
- $Q^{(k_1,k_2)}(\delta, \alpha, \beta) = \rho$ for $\rho \in \mathcal{F}(\Sigma)$ such that $\mathcal{H}^{(k_1)}(\delta, \alpha, \rho) = \mathit{true}$ and $\mathcal{H}^{(k_2)}(\delta, \rho, \beta) = \mathit{true}$ otherwise.

We define ITP to be the *family* of all interpolation-query oracles.

An algorithm in the *interpolation-query model*, also called an *interpolation-query algorithm*, is an inference from queries algorithm expecting any interpolation query oracle, where k_1, k_2 are bounded by a *polynomial* in n in all queries. The query complexity in this model is $q^{\text{ITP}}_{\mathcal{A}}(n)$.

Interpolation-query oracles form a *family* of oracles since different oracles can choose different ρ for every $\delta, \alpha, \beta, k_1, k_2$. Note that ρ may be exponentially long.

6.2.1 Lower Bound on Interpolation-Query Algorithms. We show an exponential lower bound on query complexity for interpolation-query algorithms. To this end we prove the following adaptation of Thm. 6.2:

Theorem 6.6. *Every interpolation-query inference algorithm, even computationally-unrestricted, deciding polynomial-length inference for the class of transition systems $\mathcal{P}_{\Sigma_2^P}$ (§6.1.1) has query complexity of $2^{\Omega(n)}$.*

We remark that the lower bound on the interpolation-query model does not follow directly from the result for the Hoare-query model: an interpolant for $\mathcal{H}^{(k_1+k_2)}(\delta, \alpha, \beta) = \mathit{true}$ depends on all traces of length $k_1 + k_2$ starting from states satisfying α, which may be an exponential number, so it cannot be computed simply by performing a polynomial number of Hoare queries to find these traces and computing an interpolant based on them. In principle, then, an interpolant can convey information beyond a polynomial number of Hoare queries. Our proof argument is therefore more subtle: we show that there exists a choice of an interpolant that is not more informative than the existence of some interpolant (i.e., only reveals information on $\mathcal{H}^{(k_1+k_2)}(\cdot, \cdot, \cdot)$), in the special case of systems in $\mathcal{P}_{\Sigma_2^P}$, in the maximally satisfiable branch of an algorithm's execution as used in the proof of Thm. 6.2. The full proof appears in the extended version [Feldman et al. 2020].

6.3 Impossibility of Generalization from Partial Information

Algorithms such as PDR use generalization schemes to generalize from specific states to clauses (see §2.2 and §3.2). It is folklore that "good" generalization is the key to successful invariant inference. In this section, we apply the results of §6.1 to shed light on the question of generalization. Technically, this is a discussion of the results in §6.1.

Clearly, if the generalization procedure has *full information*, that is, has unrestricted access to the input—including the transition relation—then unrestricted computational power makes the problem of generalization trivial (as is every other problem!). For example, "efficient" inference can be achieved by a backward-reachability algorithm (see §2.1) that blocks counterexamples through a generalization that uses clauses from a target invariant it can compute. This setting of full-information, computationally-unrestricted generalization was used by Padon et al. [2016] in an interactive invariant inference scenario.

Our analysis in §6.1 implies that the situation is drastically different when generalization possesses *partial information*: the algorithm does not know the transition relation exactly, and only knows the results of a polynomial number of Hoare queries. By Thm. 6.2, no choice of generalization made on the basis on this information can in general achieve inference in a polynomial number of steps. This impossibility holds even when generalization uses unrestricted computational power, and thus it is a problem of *information*. To further illustrate the idea of partial information, we note that the problem remains hard even when generalization is equipped with information beyond the results of a polynomial number of Hoare queries, information of the reachability of the transition system from *Init* and backwards from *Bad* in a *polynomial* number of steps[9]; in contrast, information of the states reachable in *any* number of steps constitutes full information and the problem is again trivial with unrestricted computational power.

Finally, the same challenge of partial information is present in algorithms basing generalization on a polynomial number of interpolation queries, as follows from Thm. 6.6.

7 THE POWER OF HOARE-QUERIES

Hoare queries are rich in the sense that the algorithm can choose a precondition α and postcondition β and check $\mathcal{H}(\delta, \alpha, \beta)$, where α may be different from β. As such, algorithms in the Hoare-query model can utilize more flexible queries beyond querying for whether a candidate is inductive. In practice, this richer form of queries facilitates an incremental construction of invariants in complex syntactic forms. For example, PDR [Bradley 2011; Eén et al. 2011] incrementally learns clauses in different frames via relative inductiveness checks, and interpolation learns at each iteration a term of the invariant from an interpolant [McMillan 2003] (see §3.2). In this section we analyze this important aspect of the Hoare-query model and show that it can be strictly stronger than inference based solely on presenting whole candidate inductive invariants. We formalize the latter approach by the model of *inductiveness-query algorithms*, closely related to ICE learning [Garg et al. 2014], and construct 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.

7.1 Inductiveness-Query Algorithms

We define a more restricted model of invariant inference using only inductiveness queries.

Definition 7.1 (Inductiveness-Query Model). An *inductiveness-query oracle* is a query oracle Q such that for every δ and $\alpha \in \mathcal{F}(\Sigma)$,

- $Q(\delta, \alpha) = \mathit{true}$ if $\alpha \wedge \delta \implies \alpha'$, and
- $Q(\delta, \alpha) = (\sigma, \sigma')$ such that $(\sigma, \sigma') \models \alpha \wedge \delta \wedge \neg\alpha'$ otherwise.

We define $\mathcal{I}$ to be the *family* of all inductiveness-query oracles.

An algorithm in the *inductiveness-query model*, also called an *inductiveness-query algorithm*, is an inference from queries algorithm expecting any inductiveness query oracle. The query complexity in this model is $q^{\mathcal{I}}_{\mathcal{A}}(n)$.

Inductiveness-query oracles form a *family* of oracles since different oracles can choose different (σ, σ') for every δ, α. Accordingly, the query complexity of inductiveness-query algorithms is measured as a worst-case query complexity over all possible choices of an inductiveness-query oracle in the family.

[9] This can be shown by noting that in $\mathcal{P}^{\Sigma_2^P}$ (used to establish the exponential lower bound) such polynomial-reachability information can be obtained from a polynomial number of Hoare queries, reducing this scenario to the original setting.

ICE learning and inductiveness-queries. The inductiveness-query model is closely related to ICE learning [Garg et al. 2014], except here the learner is provided with full information on *Init*, *Bad* instead of positive and negative examples. This model captures several interesting algorithms (see §3.2). 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. [2014]. Hence, lower bounds on the query complexity in the inductiveness query model imply lower bounds for the strong convergence of ICE learning. We formalize this in the following lemma, using terminology borrowed from Garg et al. [2014] (see §3.2):

Lemma 7.2. *Let $\mathcal{P}$ be a class of transition systems, and $\mathcal{L}$ a class of candidate invariants. Assume that deciding the existence of an invariant in $\mathcal{L}$, given an instance from $\mathcal{P}$, requires at least r queries in the inductiveness-query model. Then every strongly-convergent ICE-learner for $(\mathcal{P}, \mathcal{L})$ has round complexity at least r.*

The proof appears in the extended version [Feldman et al. 2020].

Inductiveness queries vs. Hoare queries. Inductiveness queries are specific instances of Hoare queries, where the precondition and postcondition are the same. Since Hoare queries can also find a counterexample in a polynomial number of queries (Lemma 5.3), inductiveness-query algorithms can be simulated by Hoare-query algorithms. Our results in the rest of this section establish that the converse is not true.

7.2 Separating Inductiveness-Queries from Hoare-Queries

In this section we show that the Hoare query model (Def. 5.2) is strictly stronger than the inductiveness query model (Def. 7.1). We will prove the following main theorem:

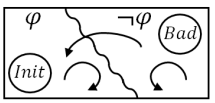
Theorem 7.3. *There exists a class of systems $\mathcal{M}_E$ for which*

- *polynomial-length invariant inference has* polynomial *query complexity in the* Hoare-query model *(in fact, also polynomial time complexity modulo the query oracle), but*
- *every algorithm in the* inductiveness-query model *requires an* exponential *number of queries.*

The upper bound is proved in Corollary 7.9, and the lower bound in Corollary 7.11.

7.2.1 Maximal Transition Systems for Monotone Invariants. We first define the transition systems with which we will prove Thm. 7.3. We start with a definition:

Definition 7.4 (Maximal System). Let *Init*, *Bad* $\not\equiv$ *false* and let φ be a formula such that $Init \implies \varphi$ and $\varphi \implies \neg Bad$. The *maximal transition system* w.r.t. φ is $(Init, \delta_\varphi^{\mathcal{M}}, Bad)$ where $\delta_\varphi^{\mathcal{M}} = \varphi \rightarrow \varphi'$. 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 the consecution of φ. Thus any transition relation $\tilde{\delta}$ for which φ satisfies consecution has $\tilde{\delta} \implies \delta_\varphi^{\mathcal{M}}$.

Lemma 7.5. *A maximal transition system $(Init, \delta_\varphi^{\mathcal{M}}, Bad)$ has a unique inductive invariant, φ.*

Proof. Let I be any invariant of $(Init, \delta_\varphi^{\mathcal{M}}, Bad)$. By the definition of $\delta_\varphi^{\mathcal{M}}$ and the fact that $Init \implies \varphi$, the set of states reachable from *Init* is exactly the set of states satisfying φ. Thus $\varphi \implies I$.

Since $\delta_\varphi^{\mathcal{M}}$ allows transitions from any state satisfying $\neg\varphi$ to *Bad*, $I \implies \varphi$. □

The class of transition systems on which we focus, $\mathcal{M}_E$, is the class of maximal systems for monotone invariants, $\mathcal{M}$, together with certain unsafe systems.

Formally, for each $k \in \mathbb{N}$, we define $\mathcal{M}^k$ as the class of all transition systems $(Init_k, \delta_\varphi^{\mathcal{M}}, Bad_k)$ for $Init_k, Bad_k$ from $\mathcal{P}^k_{\Sigma_2^P}$ (§6.1.1) and $\varphi \in \text{Mon-CNF}_{2k+1}$ such that $Init_k \implies \varphi$ and $\varphi \implies \neg Bad_k$. We then define $\mathcal{M} = \bigcup_{k\in\mathbb{N}} \mathcal{M}^k$. Further, for each k we take the unsafe program $E_k = (Init_k, true, Bad_k)$, and define the class $\mathcal{M}_E = \mathcal{M} \cup \{E_k \mid k \in \mathbb{N}\}$. Below we abbreviate and refer to the class $\mathcal{M}_E$ as "monotone maximal systems".

Note that for each k, only a single transition system, E_k, in $\mathcal{M}_E$ does not have an invariant, and the others have a monotone invariant. Still, Corollary 7.11 establishes a lower bound on polynomial-length inference for $\mathcal{M}_E$ using inductiveness queries. This means that using inductiveness queries alone, it is hard to distinguish between monotone invariants (otherwise decision would have been feasible via search). On the other hand, with Hoare queries, search becomes feasible (establishing the upper bound).

7.2.2 Upper Bound for Hoare-Query Algorithms for Monotone Maximal Systems. A simple algorithm can find inductive invariants of monotone maximal systems with a polynomial number of queries. It is essentially PDR with a single frame. The ability to *find* invariants for $\mathcal{M}_E$ (and check invariants) shows that it is possible to *decide* polynomial-length inference for $\mathcal{M}_E$.

We now present the PDR-1 algorithm (which was also discussed in §2.2, and is cast here formally as a Hoare-query algorithm). This is a backward-reachability algorithm, operating by repeatedly checking for the existence of a counterexample to induction, and obtaining a concrete example by the method discussed in Lemma 5.3. The invariant is then strengthened by conjoining the candidate invariant with the negation of the formula Block returns. This formula is a subset of the cube of the pre-state. In PDR-1, Block performs generalization 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 the strongest 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 monotone maximal systems, since maximality ensures that their diameter is one.)

Algorithm 4 PDR-1 invariant inference in the Hoare-query model

```
1:  procedure PDR-1(Init, Bad, [δ])                  // Backward-reachability with PDR-1 generalization
2:      I ← ¬Bad
3:      while H(δ, I, I) = false do                  // I not inductive
4:          (σ, σ′) ← MODEL([δ], I, ¬I′)             // counterexample to induction of I. implemented using Lemma 5.3
5:          d ← BLOCK-PDR-1(Init, [δ], σ)
6:          I ← I ∧ ¬d
        return I
7:
8:  procedure BLOCK-PDR-1(Init, [δ], σ)              // Generalization according to one-step reachability
9:      d ← cube(σ)
10:     for l ∈ cube(σ) do
11:         t ← d \ {l}
12:         if Init ⟹ ¬t ∧ H(δ, Init, ¬t) then      // Init ⟹ t ∧ Init ∧ δ ⟹ ¬t′
13:             d ← t
        return d
```

The main property of monotone CNF formulas we exploit in the upper bound is the ability to reconstruct them from *prime consequences*.

Definition 7.6 (Prime Consequence). A clause c is a *consequence* of φ if $\varphi \implies c$. A *prime consequence*, c, of φ is a minimal consequence of φ, i.e., no proper subset of c is a consequence of φ.

Theorem 7.7 (Folklore). *If $\varphi \in$ Mon-CNF$_n$ and a clause c is a prime consequence of φ then c is a clause of φ.*

Thm. 7.7 is the dual of the folklore theorem on prime implicants of monotone DNF formulas as used e.g. by Valiant [1984]. For completeness we provide a proof in the extended version [Feldman et al. 2020].

We use this property to show that PDR-1 efficiently finds the invariants of the safe maximal monotone systems $\mathcal{M}$, as implied by the following, slightly more general, lemma:

Lemma 7.8. *Let $TS = (Init, \delta, Bad)$ be a transition system over Σ, $n = |\Sigma|$, and $m \in \mathbb{N}$ such that*

(i) TS is safe,
(ii) every reachable state in TS is reachable in at most one step from $Init$,
(iii) this set can be described by a formula $\varphi \in$ Mon-CNF$_m$.

Then PDR-1*$(Init, Bad, [\delta])$ returns the inductive invariant φ for TS with at most $n \cdot m$ Hoare queries.*

From this lemma and the uniqueness of the invariants (Lemma 7.5) the upper bound for $\mathcal{M}_E$ follows easily (the proofs appear in the extended version [Feldman et al. 2020]) :

Corollary 7.9. *Polynomial-length invariant inference of $\mathcal{M}_E$ can be decided in a polynomial number of Hoare queries.*

Remark 7.1. *The condition that the invariant is monotone in Lemma 7.8 can be relaxed to* pseudo-monotonicity*: A formula φ in CNF is* pseudo-monotone *if no propositional variable appears in φ both positively and negatively. Thus it can be made monotone by renaming variables. It still holds for a pseduo-monotone CNF φ that a prime consequence is a clause of φ, and therefore PDR-1 successfully finds an invariant in a polynomial number of Hoare queries also for the class of maximal systems for* pseudo-*monotone invariants.*

7.2.3 Lower Bound for Inductiveness-Query Algorithms for Monotone Maximal Systems. We now prove that every inductiveness-query algorithm for the class of monotone maximal systems requires exponential query complexity. The main idea of the proof is that inductiveness-query algorithms are oblivious to adding transitions:

Theorem 7.10. *Let X, Y be sets of transition systems, such that Y* covers *the transition relations of X, that is, for every $(Init, \delta, Bad) \in X$ there exists $(Init, \hat{\delta}, Bad) \in Y$ over the same vocabulary s.t. (1) $\delta \implies \hat{\delta}$, and (2) if $(Init, \delta, Bad)$ has an inductive invariant in CNF$_{p(n)}$, then so does $(Init, \hat{\delta}, Bad)$. Then if $\mathcal{A}$ is an inductiveness-query algorithm for Y with query complexity t, then $\mathcal{A}$ is also an inductiveness-query algorithm for X with query complexity t.*

Proof. Let $\mathcal{A}$ be an algorithm for Y as in the premise. We show that $\mathcal{A}$ also solves the problem for X. Let $(Init, \delta, Bad) \in X$ and analyze $\mathcal{A}^Q(Init, Bad, [\delta])$, where Q is some inductiveness-query oracle. Consider the first t candidates, $\alpha_1, \ldots, \alpha_t$. If one of them is an inductive invariant for $(Init, \delta, Bad)$, we are done. Otherwise, let $(Init, \hat{\delta}, Bad) \in Y$ as in the premise for the given $(Init, \delta, Bad)$. We show that in this case $\mathcal{A}^Q(Init, Bad, [\delta])$ simulates $\mathcal{A}^{Q'}(Init, Bad, [\hat{\delta}])$ where Q' is an inductiveness-query oracle derived from Q by $Q'(\hat{\delta}, \alpha_i) \;=\; Q(\delta, \alpha_i)$ for all $i = 1, \ldots, t$. Note that $Q'(\hat{\delta}, \cdot)$ is a valid inductiveness-query oracle: by the assumption that α_i is not inductive for δ, $Q(\delta, \alpha) = (\sigma, \sigma')$, that is, $\sigma, \sigma' \models \alpha \wedge \delta \wedge \neg\alpha'$. From condition 1, $\delta \implies \hat{\delta}$, and so we deduce that also $\sigma, \sigma' \models \alpha \wedge \hat{\delta} \wedge \neg\alpha'$. Therefore, after at most t queries, $\mathcal{A}^{Q'}(Init, Bad, [\hat{\delta}])$ terminates, returning either (i) an inductive

Table 1. Concept vs. invariant learning: query complexity of learning Mon-CNF$_n$

Invariant Inference			Concept Learning	
	Maximal Systems	General Systems		
Inductiveness	Exponential (Corollary 7.11)	Exponential (Thm. 6.2)	Equivalence	Subexponential[1] / Polynomial[2] [Angluin 1987; Hellerstein et al. 2012]
Hoare	Polynomial (Corollary 7.9)	Exponential (Thm. 6.2)	Equivalence + Membership	Polynomial [Angluin 1987]

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

invariant $\varphi \in \text{CNF}_{p(n)}$ for $(\mathit{Init}, \hat{\delta}, \mathit{Bad})$, which is also an inductive invariant for $(\mathit{Init}, \delta, \mathit{Bad})$, by condition 1; or (ii) no inductive invariant in $\text{CNF}_{p(n)}$ for $(\mathit{Init}, \hat{\delta}, \mathit{Bad})$, in which case this is also true for $(\mathit{Init}, \delta, \mathit{Bad})$, by condition 2. Either way $\mathcal{A}^{\mathcal{Q}}(\mathit{Init}, \mathit{Bad}, [\delta])$ is correct and uses $\leq t$ queries. □

The lower bound for monotone maximal systems results from Thm. 7.10 together with the hardness previously obtained in Thm. 6.2:

Corollary 7.11. *Every inductiveness-query algorithm, even computationally-unrestricted, deciding polynomial-length inference for $\mathcal{M}_E$ has query complexity of $2^{\Omega(n)}$.*

The proof applies Thm. 7.10 to $\mathcal{M}_E$, which covers $\mathcal{P}_{\Sigma_2^P}$, while the hardness of the latter was established in Thm. 6.2. The full proof appears in the extended version [Feldman et al. 2020].

We note that the transition relations in $\mathcal{M}_E$ are themselves polynomial in $|\Sigma|$. Hence the query complexity in this lower bound is exponential not only in $|\Sigma|$ but also in $|\delta|$ (see Remark 5.2).

Finally, it is interesting to notice that the safe systems in $\mathcal{M}_E$ have a *unique* inductive invariant, and still the problem is hard.

8 INVARIANT LEARNING & CONCEPT LEARNING WITH QUERIES

The theory of *exact concept learning* [Angluin 1987] asks a learner to identify an unknown formula[10] φ from a class $\mathcal{L}$ using queries posed to a teacher. Prominent types of queries include *membership*—given state σ, return whether $\sigma \models \varphi$—and *equivalence*—given θ, return true if $\theta \equiv \varphi$ or, otherwise, a counterexample, a σ s.t. $\sigma \not\models \theta, \sigma \models \varphi$ or vice versa.

What are the connections and differences between 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. Garg et al. 2014; Jha and Seshia 2017]. In this section we study these questions with the tool of query complexity and our aforementioned results.

The most significant outcome of this analysis is a new hardness result (Corollary 8.1) showing that ICE-learning is provably harder than classical learning: namely, that, as advocated by Garg et al. [2014], learning from counterexamples to induction is inherently harder than learning from examples labeled positive or negative. The proof of this result builds on the lower bound of Corollary 7.11. We also establish (im)possibility results for directly applying algorithms from concept learning to invariant inference.

Complexity: the easy, the complex, and the even-more-complex. In this paper we have studied the complexity of inferring $\mathcal{L} = \text{Mon-CNF}_n$ invariants using Hoare/inductiveness queries in two settings: for *general systems* (in §6.1), and for *maximal systems* in §7. Table 1 summarizes our results and contrasts them with known complexity results in classical concept learning for the same class of formulas. For the sake of the comparison, the table maps inductiveness queries to equivalence

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

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

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

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

Starting with similarity, the gap in the complexity between Hoare- and inductiveness-queries in learning invariants for maximal systems parallels the gap between equivalence and equivalence + membership queries in concept learning. Our proof for the upper bound for Hoare queries is related to the upper bound in concept learning and simulations of concept learning algorithms (see below), but the lower bound for inductiveness queries uses very different ideas, and establishes stronger lower bounds than possible in concept learning, as we describe below.

The similarity ends here. First, *general systems are harder*, and inferring $\mathcal{L} = \text{Mon-CNF}_n$ invariants for them is harder than concept learning with the same $\mathcal{L}$, even with the full power of Hoare queries. This, unsurprisingly, illustrates the challenges stemming from transition systems with complex reachability patterns, such as a large diameter. Second, even the hard cases for concept learning have *lower* complexity than the hard invariant inference problems: learning concepts in $\mathcal{L} = \text{Mon-CNF}_n$ has *subexponential* query complexity (or even polynomial complexity when exponentially-long candidates are allowed), whereas we prove *exponential* lower bounds for inference. One important instance of this discrepancy shows that inductiveness queries are inherently *weaker* than equivalence queries, as learning Mon-CNF_n invariants in the *inductiveness* model is *harder* than learning Mon-CNF_n formulas using *equivalence* queries. Put differently, this is a hardness result for concept learning with *ICE-equivalence queries*, which are like equivalence queries, only when the given θ is not equivalent to the target concept φ the teacher responds with an *implication* counterexample [Garg et al. 2014]: a pair σ, σ' s.t. $\sigma \models \theta$ and $\sigma' \not\models \theta$, but $\sigma \not\models \varphi$ or $\sigma' \models \varphi$. Our results thus imply:

Corollary 8.1. *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. [2014].

The higher complexity of inferring invariants has consequences for the feasibility of simulating queries (and algorithms) from concept learning in invariant inference, as we discuss next.

Queries: some unimplementable algorithms. Table 2 summarizes our results for the possibility and impossibility of simulating concept learning algorithms in invariant learning. This table depicts implementability (✓) or unimplementability (✗) of membership and equivalence queries used in concept learning a class of formulas $\mathcal{L}$ through inductiveness and Hoare queries used in learning invariants for maximal systems over $\mathcal{L}$, and for general systems with candidate invariants in $\mathcal{L}$. The proofs of impossibilities are based on the differences in complexity described above: that neither equivalence nor membership queries can be simulated over general systems using even Hoare queries is implied by the hardness of general systems; that neither equivalence nor membership can be simulated even over maximal systems using inductiveness queries is implied by the higher complexity of these compared to concept learning. The only possibility result is of simulating inductiveness and membership queries using Hoare queries over maximal systems; the idea is that a Hoare query $\mathcal{H}(\delta_\varphi^{\mathcal{M}}, \mathit{Init}, \neg \mathit{cube}(\sigma)) \stackrel{?}{=} \mathit{false}$ implements a membership query on σ, thanks

to fact that the inductive invariant is exactly the set of states reachable in one step, and that a membership query can disambiguate a counterexample to induction into a labeled example, so it is possible to simulate an equivalence query by an inductiveness query. Interestingly, the algorithm we use to show the polynomial upper bound on Hoare queries for maximal systems, PDR-1, can be obtained as such a translation of an algorithm from Angluin [1987] performing concept learning of Mon-CNF$_n$ using equivalence and membership queries.

9 RELATED WORK

Complexity of invariant inference. The fundamental question of the complexity of invariant inference in propositional logic has been studied by Lahiri and Qadeer [2009]. They show that deciding whether an invariant exists is PSPACE-complete. This includes systems with only exponentially-long invariants, which are inherently beyond reach for algorithms aiming to *construct* an invariant. In this paper we focus on the search for polynomially-long invariants. Lahiri and Qadeer [2009] study the related problem of template-based inference, and show it is Σ_2^P-complete. Polynomial-length inference for CNF formulas can be encoded as specific instances of template-based inference; the Σ_2^P-hardness proof of Lahiri and Qadeer [2009] uses more general templates and therefore does not directly imply the same hardness for polynomial-length inference. The same work also shows that inference is only Π_1^P = **coNP**-complete when candidates are only conjunctions (or, dually, disjunctions). In this paper we focus on the richer class of CNF invariants.

Black-box invariant inference. Black-box access to the program in its analysis is widespread in research on testing [e.g. Nidhra and Dondeti 2012]. In invariant inference, Daikon [Ernst et al. 2001] initiated the black-box learning of *likely* program invariants [see e.g. Csallner et al. 2008; Sankaranarayanan et al. 2008]. In this paper we are interested in inferring necessarily correct inductive invariants. The ICE learning model, introduced by Garg et al. [2014, 2016], and extended to general Constrained Horn Clauses in later work [Ezudheen et al. 2018], pioneered a black-box view of inference algorithms such as Houdini [Flanagan and Leino 2001] and symbolic abstraction [Reps et al. 2004; Thakur et al. 2015]. The inductiveness model in our work is inspired by this work, focusing on black-box access to the transition relation while providing the learner with full knowledge of the set of initial and bad states. Capturing PDR in a black-box model was achieved by extending ICE with relative-inductiveness queries [Vizel et al. 2017]. Our work shows that an extension is necessary, and applies to any Hoare-query algorithm.

Lower bounds for black-box inference. To the best of our knowledge, our work provides the first unconditional exponential lower bound for rich black-box inference models such as the Hoare-query model. An impossibility result for ICE learning in polynomial time in the setting of quantified invariants was obtained by Garg et al. [2014], based on the lower bound of Angluin [1990] for concept learning DFAs with equivalence queries. Our lower bound for monotone maximal systems (i) demonstrates an exponential gap between ICE learning and Hoare-query algorithms such as PDR (§7), and (ii) separates ICE learning from concept learning (§8); in particular, it holds even when candidates may be exponentially long (see Corollary 7.11 and Garg et al. [2013, Appendix B]).

Learning and synthesis with queries. Connections with exact learning with queries [Angluin 1987] are discussed in §8. The lens of synthesis has inspired many works applying ideas from machine learning to invariant inference [e.g. Garg et al. 2014; Jha et al. 2010; Sharma and Aiken 2016; Sharma et al. 2013b,a, 2012]. The role of learning with queries is recognized in prominent synthesis approaches such as Counterexample-Guided Inductive Synthesis (CEGIS) [Solar-Lezama et al. 2006] and synthesizer-driven approaches [e.g. Gulwani 2012; Jha et al. 2010; Le et al. 2017], which learn from equivalence and membership queries [Alur et al. 2015; Bshouty et al. 2017; Drachsler-Cohen et al. 2017; Jha and Seshia 2017]. The theory of oracle-guided inductive synthesis [Jha and Seshia

2017] theoretically studies the convergence of CEGIS in infinite concept classes using different types of counterexamples-oracles, and relates the finite case to the teaching dimension [Goldman and Kearns 1995]. In this work we study inference based on a different form of queries, and prove lower bounds on the convergence rate in finite classes.

Proof complexity. Proof complexity studies the power of polynomially-long proofs in different proof systems. A seminal result is that a propositional encoding of the pigeonhole principle has no polynomial resolution proofs [Haken 1985]. Ideas and tools from proof complexity have been applied to study SAT solvers [e.g. Pipatsrisawat and Darwiche 2011] and recently also SMT [Robere et al. 2018]. Proof complexity is an alternative technical approach to study the complexity of proof search algorithms, by showing that some instances do not have a short proof, showing a lower bound regardless of how search is conducted. Our work, inspired by learning theory, provides exponential lower bounds on query-based search even when the proof system is sufficiently strong to admit short proofs: in our setting, there is always a short derivation of an inductive invariant by generalization in backward-reachability, blocking counterexamples with the optimal choice, using clauses from a target invariant (see §6.3). We expect that proof complexity methods would prove valuable in further study of inference.

10 CONCLUSION

Motivated by the rise of SAT-based invariant inference algorithms, we have attempted to elucidate some of the principles on which they are based by a theoretical complexity analysis of algorithms attempting to infer invariants of polynomial size. We have developed information-based analysis tools, inspired by machine learning theory, to investigate two focal points in SAT-based inference design: (1) *Generalization*, which we have shown to be impossible from a polynomial number of Hoare queries in the general case; (2) *Rich Hoare queries*, beyond presenting candidate invariants, which we have shown to be pivotal in some cases. Our upper bound for PDR on the class of monotone maximal systems is a first step towards theoretical conditions guaranteeing polynomial running time for such algorithms. One lesson from our results is the importance of characteristics of the transition relations (rather than of candidate invariants), which make the difference between the lower bound for general systems (Thm. 6.2) and the upper bound for maximal systems (Corollary 7.9), both for the same class of candidate invariants. We believe that theoretical guarantees of efficient inference would involve special classes of transitions systems and algorithms using repeated generalization employing rich Hoare queries.

At the heart of our analysis lies the observation that many interesting SAT-based algorithms can be cast in a black-box model. This work focuses on the limits and opportunities in black-box inference and shows interesting information-theoretic lower bounds. One avenue for further research is an information-based analysis of black-box models extended with white-box capabilities, e.g. by investigating syntactical conditions on the transition relation that simplify generalization.

ACKNOWLEDGMENTS

We thank our shepherd and the anonymous referees for comments that improved the paper. We thank Kalev Alpernas, Nikolaj Bjørner, P. Madhusudan, Yishay Mansour, Oded Padon, Hila Peleg, Muli Safra, and James R. Wilcox for insightful discussions and suggestions, and Gil Buchbinder for saving a day. 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 National Science Foundation (NSF) grant no. CCF-1617498, by Len Blavatnik and the Blavatnik Family foundation, the Blavatnik Interdisciplinary Cyber Research Center, Tel Aviv University, the United States-Israel Binational Science Foundation (BSF) grant No. 2016260, and the Israeli Science Foundation (ISF) grant No. 1810/18.

REFERENCES

Rajeev Alur, Rastislav Bodík, Eric Dallal, Dana Fisman, Pranav Garg, Garvit Juniwal, Hadas Kress-Gazit, P. Madhusudan, Milo M. K. Martin, Mukund Raghothaman, Shambwaditya Saha, Sanjit A. Seshia, Rishabh Singh, Armando Solar-Lezama, Emina Torlak, and Abhishek Udupa. 2015. Syntax-Guided Synthesis. In *Dependable Software Systems Engineering*. 1–25.

Dana Angluin. 1987. Queries and Concept Learning. *Machine Learning* 2, 4 (1987), 319–342.

Dana Angluin. 1990. Negative Results for Equivalence Queries. *Machine Learning* 5 (1990), 121–150.

Nikolaj Bjørner and Arie Gurfinkel. 2015. Property Directed Polyhedral Abstraction. In *Verification, Model Checking, and Abstract Interpretation - 16th International Conference, VMCAI 2015, Mumbai, India, January 12-14, 2015. Proceedings*. 263–281. https://doi.org/10.1007/978-3-662-46081-8_15

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*. 70–87. https://doi.org/10.1007/978-3-642-18275-4_7

Nader H. Bshouty, Dana Drachsler-Cohen, Martin T. Vechev, and Eran Yahav. 2017. Learning Disjunctions of Predicates. In *Proceedings of the 30th Conference on Learning Theory, COLT 2017, Amsterdam, The Netherlands, 7-10 July 2017*. 346–369.

Alessandro Cimatti, Alberto Griggio, Sergio Mover, and Stefano Tonetta. 2014. IC3 Modulo Theories via Implicit Predicate Abstraction. In *Tools and Algorithms for the Construction and Analysis of Systems - 20th International Conference, TACAS 2014, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2014, Grenoble, France, April 5-13, 2014. Proceedings*. 46–61. https://doi.org/10.1007/978-3-642-54862-8_4

Michael Colón, Sriram Sankaranarayanan, and Henny Sipma. 2003. Linear Invariant Generation Using Non-linear Constraint Solving. In *Computer Aided Verification, 15th International Conference, CAV 2003, Boulder, CO, USA, July 8-12, 2003, Proceedings*. 420–432.

Patrick Cousot and Radhia Cousot. 1977. Abstract Interpretation: A Unified Lattice Model for Static Analysis of Programs by Construction or Approximation of Fixpoints. In *Conference Record of the Fourth ACM Symposium on Principles of Programming Languages, Los Angeles, California, USA, January 1977*. 238–252. https://doi.org/10.1145/512950.512973

Christoph Csallner, Nikolai Tillmann, and Yannis Smaragdakis. 2008. DySy: dynamic symbolic execution for invariant inference. In *30th International Conference on Software Engineering (ICSE 2008), Leipzig, Germany, May 10-18, 2008*. 281–290. https://doi.org/10.1145/1368088.1368127

Isil Dillig, Thomas Dillig, Boyang Li, and Kenneth L. McMillan. 2013. Inductive invariant generation via abductive inference. In *Proceedings of the 2013 ACM SIGPLAN International Conference on Object Oriented Programming Systems Languages & Applications, OOPSLA 2013, part of SPLASH 2013, Indianapolis, IN, USA, October 26-31, 2013*. 443–456.

Dana Drachsler-Cohen, Sharon Shoham, and Eran Yahav. 2017. Synthesis with Abstract Examples. In *Computer Aided Verification - 29th International Conference, CAV 2017, Heidelberg, Germany, July 24-28, 2017, Proceedings, Part I*. 254–278. https://doi.org/10.1007/978-3-319-63387-9_13

Niklas Eén, Alan Mishchenko, and Robert K. Brayton. 2011. 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*. 125–134. http://dl.acm.org/citation.cfm?id=2157675

Michael D. Ernst, Jake Cockrell, William G. Griswold, and David Notkin. 2001. Dynamically Discovering Likely Program Invariants to Support Program Evolution. *IEEE Trans. Software Eng.* 27, 2 (2001), 99–123. https://doi.org/10.1109/32.908957

P. Ezudheen, Daniel Neider, Deepak D'Souza, Pranav Garg, and P. Madhusudan. 2018. Horn-ICE learning for synthesizing invariants and contracts. *PACMPL* 2, OOPSLA (2018), 131:1–131:25.

Grigory Fedyukovich and Rastislav Bodík. 2018. Accelerating Syntax-Guided Invariant Synthesis. In *Tools and Algorithms for the Construction and Analysis of Systems - 24th International Conference, TACAS 2018, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2018, Thessaloniki, Greece, April 14-20, 2018, Proceedings, Part I*. 251–269. https://doi.org/10.1007/978-3-319-89960-2_14

Yotam M. Y. Feldman, Neil Immerman, Mooly Sagiv, and Sharon Shoham. 2020. *Complexity and Information in Invariant Inference*. Technical Report. arXiv:1910.12256 https://arxiv.org/abs/1910.12256

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*. 500–517.

Cormac Flanagan and Shaz Qadeer. 2002. Predicate abstraction for software verification. In *Conference Record of POPL 2002: The 29th SIGPLAN-SIGACT Symposium on Principles of Programming Languages, Portland, OR, USA, January 16-18, 2002*. 191–202. https://doi.org/10.1145/503272.503291

Pranav Garg, Christof Löding, P Madhusudan, and Daniel Neider. 2013. *ICE: A robust framework for learning invariants*. Technical Report. 69–87 pages. http://hdl.handle.net/2142/45973

Pranav Garg, Christof Löding, P Madhusudan, and Daniel Neider. 2014. ICE: A robust framework for learning invariants. In *Computer Aided Verification*. Springer, 69–87.

Pranav Garg, Daniel Neider, P. Madhusudan, and Dan Roth. 2016. 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*. 499–512. https://doi.org/10.1145/2837614.2837664

Sally A. Goldman and Michael J. Kearns. 1995. On the Complexity of Teaching. *J. Comput. Syst. Sci.* 50, 1 (1995), 20–31. https://doi.org/10.1006/jcss.1995.1003

Oded Goldreich. 2006. On Promise Problems: A Survey. In *Theoretical Computer Science, Essays in Memory of Shimon Even*. 254–290.

Susanne Graf and Hassen Saïdi. 1997. Construction of Abstract State Graphs with PVS. In *Computer Aided Verification, 9th International Conference, CAV '97, Haifa, Israel, June 22-25, 1997, Proceedings*. 72–83. https://doi.org/10.1007/3-540-63166-6_10

Sumit Gulwani. 2012. Synthesis from Examples: Interaction Models and Algorithms. In *14th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing, SYNASC 2012, Timisoara, Romania, September 26-29, 2012*. 8–14. https://doi.org/10.1109/SYNASC.2012.69

Arie Gurfinkel and Alexander Ivrii. 2015. Pushing to the Top. In *Formal Methods in Computer-Aided Design, FMCAD 2015, Austin, Texas, USA, September 27-30, 2015*. 65–72.

Armin Haken. 1985. The Intractability of Resolution. *Theor. Comput. Sci.* 39 (1985), 297–308. https://doi.org/10.1016/0304-3975(85)90144-6

Lisa Hellerstein, Devorah Kletenik, Linda Sellie, and Rocco A. Servedio. 2012. 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*. 31.1–31.18. http://proceedings.mlr.press/v23/hellerstein12/hellerstein12.pdf

Thomas A. Henzinger, Ranjit Jhala, Rupak Majumdar, and Kenneth L. McMillan. 2004. 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*. 232–244. https://doi.org/10.1145/964001.964021

Krystof Hoder and Nikolaj Bjørner. 2012. Generalized Property Directed Reachability. In *Theory and Applications of Satisfiability Testing - SAT 2012 - 15th International Conference, Trento, Italy, June 17-20, 2012. Proceedings*. 157–171.

Bertrand Jeannet, Peter Schrammel, and Sriram Sankaranarayanan. 2014. Abstract acceleration of general linear loops. In *The 41st Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL '14, San Diego, CA, USA, January 20-21, 2014*. 529–540. https://doi.org/10.1145/2535838.2535843

Susmit Jha, Sumit Gulwani, Sanjit A. Seshia, and Ashish Tiwari. 2010. 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*. 215–224. https://doi.org/10.1145/1806799.1806833

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

Ranjit Jhala and Kenneth L. McMillan. 2007. Interpolant-Based Transition Relation Approximation. *Logical Methods in Computer Science* 3, 4 (2007). https://doi.org/10.2168/LMCS-3(4:1)2007

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

Anvesh Komuravelli, Arie Gurfinkel, and Sagar Chaki. 2014. SMT-Based Model Checking for Recursive Programs. 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*. 17–34.

Shuvendu K. Lahiri and Shaz Qadeer. 2009. 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*. 214–229.

Vu Le, Daniel Perelman, Oleksandr Polozov, Mohammad Raza, Abhishek Udupa, and Sumit Gulwani. 2017. Interactive Program Synthesis. *CoRR* (2017). arXiv:1703.03539 http://arxiv.org/abs/1703.03539

Christof Löding, P. Madhusudan, and Daniel Neider. 2016. Abstract Learning Frameworks for Synthesis. In *Tools and Algorithms for the Construction and Analysis of Systems - 22nd International Conference, TACAS 2016, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2016, Eindhoven, The Netherlands, April 2-8, 2016, Proceedings*. 167–185. https://doi.org/10.1007/978-3-662-49674-9_10

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*. 1–13.

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

Srinivas Nidhra and Jagruthi Dondeti. 2012. Black Box and White Box Testing Techniques - A Literature Review. *International Journal of Embedded Systems and Applications* 2 (06 2012), 29–50. https://doi.org/10.5121/ijesa.2012.2204

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. 614–630.

Knot Pipatsrisawat and Adnan Darwiche. 2011. On the power of clause-learning SAT solvers as resolution engines. *Artif. Intell.* 175, 2 (2011), 512–525. https://doi.org/10.1016/j.artint.2010.10.002

Thomas W. Reps, Shmuel Sagiv, and Greta Yorsh. 2004. 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*. 252–266. https://doi.org/10.1007/978-3-540-24622-0_21

Robert Robere, Antonina Kolokolova, and Vijay Ganesh. 2018. The Proof Complexity of SMT Solvers. In *Computer Aided Verification - 30th International Conference, CAV 2018, Held as Part of the Federated Logic Conference, FloC 2018, Oxford, UK, July 14-17, 2018, Proceedings, Part II*. 275–293. https://doi.org/10.1007/978-3-319-96142-2_18

Sriram Sankaranarayanan, Swarat Chaudhuri, Franjo Ivancic, and Aarti Gupta. 2008. Dynamic inference of likely data preconditions over predicates by tree learning. In *Proceedings of the ACM/SIGSOFT International Symposium on Software Testing and Analysis, ISSTA 2008, Seattle, WA, USA, July 20-24, 2008*. 295–306. https://doi.org/10.1145/1390630.1390666

Sriram Sankaranarayanan, Henny B. Sipma, and Zohar Manna. 2004. Constraint-Based Linear-Relations Analysis. In *Static Analysis, 11th International Symposium, SAS 2004, Verona, Italy, August 26-28, 2004, Proceedings*. 53–68.

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

Rahul Sharma, Saurabh Gupta, Bharath Hariharan, Alex Aiken, Percy Liang, and Aditya V. Nori. 2013b. 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*. 574–592. https://doi.org/10.1007/978-3-642-37036-6_31

Rahul Sharma, Saurabh Gupta, Bharath Hariharan, Alex Aiken, and Aditya V. Nori. 2013a. Verification as Learning Geometric Concepts. In *Static Analysis - 20th International Symposium, SAS 2013, Seattle, WA, USA, June 20-22, 2013. Proceedings*. 388–411.

Rahul Sharma, Aditya V. Nori, and Alex Aiken. 2012. Interpolants as Classifiers. In *Computer Aided Verification - 24th International Conference, CAV 2012, Berkeley, CA, USA, July 7-13, 2012 Proceedings*. 71–87. https://doi.org/10.1007/978-3-642-31424-7_11

Armando Solar-Lezama, Liviu Tancau, Rastislav Bodík, Sanjit A. Seshia, and Vijay A. Saraswat. 2006. Combinatorial sketching for finite programs. In *Proceedings of the 12th International Conference on Architectural Support for Programming Languages and Operating Systems, ASPLOS 2006, San Jose, CA, USA, October 21-25, 2006*. 404–415. https://doi.org/10.1145/1168857.1168907

Saurabh Srivastava and Sumit Gulwani. 2009. Program verification using templates over predicate abstraction. In *Proceedings of the 2009 ACM SIGPLAN Conference on Programming Language Design and Implementation, PLDI 2009, Dublin, Ireland, June 15-21, 2009*. 223–234.

Saurabh Srivastava, Sumit Gulwani, and Jeffrey S. Foster. 2013. Template-based program verification and program synthesis. *STTT* 15, 5-6 (2013), 497–518.

Aditya V. Thakur, Akash Lal, Junghee Lim, and Thomas W. Reps. 2015. PostHat and All That: Automating Abstract Interpretation. *Electr. Notes Theor. Comput. Sci.* 311 (2015), 15–32. https://doi.org/10.1016/j.entcs.2015.02.003

Leslie G. Valiant. 1984. A Theory of the Learnable. *Commun. ACM* 27, 11 (1984), 1134–1142. https://doi.org/10.1145/1968.1972

Yakir Vizel and Orna Grumberg. 2009. 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*. 1–8. https://doi.org/10.1109/FMCAD.2009.5351148

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

Yakir Vizel and Arie Gurfinkel. 2014. Interpolating Property Directed Reachability. 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*. 260–276. https://doi.org/10.1007/978-3-319-08867-9_17

Yakir Vizel, Arie Gurfinkel, Sharon Shoham, and Sharad Malik. 2017. 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*. 521–538.