From Concept Learning to SAT-Based Invariant Inference

Sharon Shoham

Supervised Verification of Infinite-State Systems

FSTTCS 2023
Safety Verification

No bad state is reachable, along any execution of the system

• No null dereference
• No mutual exclusion violation
• No consensus violation: at most one value is chosen

...
Safety Verification

Safety: no bad state is reachable from the initial states

\[ \text{Init:} \quad (x_1, \ldots, x_n) := 0 \ldots 0 \]
\[ \text{Bad:} \quad (x_1, \ldots, x_n) = 1 \ldots 1 \]

\[ \delta: \quad y_1, \ldots, y_n := * \]
\[ x_1, \ldots, x_n := (x_1, \ldots, x_n) + 2 \cdot (y_1, \ldots, y_n) \pmod{2^n} \]
Inductive Invariants

Safety: no bad state is reachable from the initial states

\[
\text{Init:} \quad (x_1, \ldots, x_n) := 0 \ldots 0
\]

\[
\text{Bad:} \quad (x_1, \ldots, x_n) = 1 \ldots 1
\]

\[
\delta: \quad y_1, \ldots, y_n := *
\]

\[
x_1, \ldots, x_n := (x_1, \ldots, x_n) + 2 \cdot (y_1, \ldots, y_n) \pmod{2^n}
\]

Initiation: \( \text{Init} \subseteq I \)
Safety: \( I \cap \text{Bad} = \emptyset \)
Consecution: \( \{I\} \delta \{I\} \)
Inductive Invariants

Safety: no bad state is reachable from the initial states

\[
\text{Init: } (x_1, \ldots, x_n) := 0 \ldots 0
\]

\[
\text{Bad: } (x_1, \ldots, x_n) = 1 \ldots 1
\]

\[
\delta: \\
y_1, \ldots, y_n := * \\
x_1, \ldots, x_n := (x_1, \ldots, x_n) + 2 \cdot (y_1, \ldots, y_n) \pmod{2^n}
\]

\[
\begin{align*}
I: & \quad x_n \neq 1 \\
\text{Inductive:} & \\
I: & \quad (x_1, \ldots, x_n) \neq 1 \ldots 1 \\
\text{Not inductive:}
\end{align*}
\]
Invariant Inference

Goal: Find inductive invariants automatically

Initiation:
\[ \text{Init} \subseteq \mathcal{I} \]
Safety:
\[ \mathcal{I} \cap \text{Bad} = \emptyset \]

Consecution:
\[ \delta(\mathcal{I}) \subseteq \mathcal{I} \]

Not inductive:
\[ \mathcal{I} : (x_1, \ldots, x_n) \neq 1 \ldots 1 \]

Inductive:
\[ \mathcal{I} : x_n \neq 1 \]

\[ \begin{align*}
(x_1, \ldots, x_n) & = 1 \ldots 1 \\
2 \cdot (y_1, \ldots, y_n) & \pmod{2^n}
\end{align*} \]
SAT-Based Invariant Inference

Goal: Find inductive invariants automatically

Means: Employ a SAT solver

\[ y_1, \ldots, y_n \equiv x_1, \ldots, x_n \equiv x_1, \ldots, x_n + 2 \cdot (y_1, \ldots, y_n) \pmod{2n} \]

\[ (x_1, \ldots, x_n) \equiv 0 \ldots 0 \]

\[ (x_1, \ldots, x_n) = 1 \ldots 1 \]

\( I \): \( x_n \neq 1 \)

Inductive:

\[ x_n \neq 1 \]

\( I \) \( \delta \)

Not inductive:

\[ (x_1, \ldots, x_n) \neq 1 \ldots 1 \]

\( I \) \( \delta \) \( \neg I \)

Init: Initial states

Bad: Bad states

Safety: no bad state is reachable from the initial states

SAT-Based Invariant Inference

Initiation: \( I \subseteq \text{Init} \)

Safety: \( I \cap \text{Bad} = \emptyset \)

Consecution: \( \delta(I) \subseteq I \)

\( I \): \( (x_1, \ldots, x_n) \neq 1 \ldots 1 \)

Means: Employ a SAT solver
SAT-Based Invariant Inference

Goal: Find inductive invariants automatically

Means: Employ a SAT solver

Init, Bad: formulas over $X$
\[\delta: \text{formula over } X, X'\]

SAT query Examples:

Initiation: \(\text{Init} \land \neg I \) unsat?
Safety: \(I \land \text{Bad} \) unsat?
Cons.: \(I \land \delta \land \neg I' \) unsat?

* \(I' = I[X \mapsto X']\)
SAT-Based Invariant Inference

- predicate abstraction [CAV’97, POPL’02]
- symbolic abstraction [VMCAI’04,’16]
- interpolation [CAV’03, TACAS’06]
- IC3/PDR [VMCAI’11, FMCAD’11]
- abduction [OOPSLA’13]
- SyGuS [FMCAD’13,...]
- ICE learning [CAV’14, POPL’15]
- ...

Why do they succeed?
Why do they fail?
(How can we make them better?)
Goal

Understand SAT-based invariant inference from the perspective of exact learning with queries

You
Are SAT-based invariant inference and exact concept learning related?

ChatGPT
While these concepts are related in the sense that they both involve logical reasoning and learning, they operate in different domains and have different focuses. SAT-based invariant inference is more specific to program analysis and verification, while exact concept learning is a broader term that can be applied to various domains in machine learning and knowledge representation.
Goal

Understand **SAT-based invariant inference** from the perspective of **exact learning with queries**

**You**

Are SAT-based invariant inference and exact concept learning related?

- **POPL20**
  Complexity and information in invariant inference.
  Feldman, Immerman, Sagiv, Shoham

- **POPL21**
  Learning the boundary of inductive invariants.
  Feldman, Sagiv, Shoham, Wilcox

- **SAS22**
  Invariant Inference With Provable Complexity From the Monotone Theory.
  Feldman, Shoham
Exact Concept Learning with Equivalence & Membership Queries

Goal: learn an unknown concept $\varphi$

Learning algorithm

is it $\psi_1$?

✓ / ✘ + counterexample

is it $\psi_2$?

✓ / ✘ + counterexample

does $\sigma_3 \models ?$

✓ / ✗

Membership  Equivalence

Oracle

[ML’87] Queries and Concept Learning. Angluin

positive: $\sigma \models \varphi$, $\sigma \not\models \psi_1$

negative: $\sigma \not\models \varphi$, $\sigma \models \psi_1$
SAT-Based Invariant Inference as Inference with Queries

Goal: infer an unknown inductive invariant $I$

Algorithms cannot access the transition relation directly, only through SAT queries
This Talk

Invariant Inference

- Query-based learning models for invariant inference
- Complexity lower and upper bounds for each model
- Invariant inference is harder than concept learning
- Complexity results for invariant inference algorithms from concept learning algorithms

Exact Concept Learning

vs.
**Inductiveness-Query Model**

**ICE framework** - Learn from examples:

- **Positive**: \( \sigma \models I \) (e.g., initial)
- **Negative**: \( \sigma \not\models I \) (e.g., bad)
- **Implication**: \( \sigma \models I \) implies \( \sigma' \models I \) (CTI)

\[
\alpha_i \land \delta \land \neg \alpha'_i \text{ unsat?}
\]

**Cex to Induction (CTI):** Transition \((\sigma, \sigma')\) of \(\delta\) s.t.
\[
\sigma \models \alpha_i, \; \sigma' \models \neg \alpha_i
\]

*\(\alpha'_i = \alpha_i[X \mapsto X']\)*

ICE framework - Learn from examples:
Positive: $\sigma \models I$ (e.g., initial)
Negative: $\sigma \not\models I$ (e.g., bad)
Implication: $\sigma \models I$ implies $\sigma' \models I$ (CTI)

Is it sufficient to capture existing SAT-based algorithms?

*C* $\alpha'_i = \alpha_i[X \mapsto X']$

Interpolation-Based Inference

\[ I_0 = \text{Init} \quad \text{Inductive?} \quad \text{no – let’s weaken} \]

[CAV’03] Interpolation and SAT-Based Model Checking, McMillan
[HVC’12] Computing Interpolants without Proofs. Chockler, Ivrii, Matsliah
[LPAR’13] Instantiations, Zippers and EPR Interpolation. Bjørner, Gurfinkel, Korovin, Lahav
Model-Based Interpolation

\[ I_0 = Init \]

Inductive? \[ \text{no} \] – let’s weaken

\[ \sigma_1 \]

\[ k \text{ times} \]

\[ \text{BMC}^k (\sigma_1, \delta, \text{Bad}) \text{ unsat} \]

\[ (\_, \sigma_1) \text{ CTI to } I_0 \]

\[ \text{Otherwise: failure} \]

\[ * k \text{ is a parameter} \]

[HVC’12] Computing Interpolants without Proofs. Chockler, Ivrii, Matsliah
[LPAR’13] Instantiations, Zippers and EPR Interpolation. Bjørner, Gurfinkel, Korovin, Lahav
Model-Based Interpolation

\[ I_0 = Init \]

\[ \begin{align*}
I_0 &= Init \\
I &= \ldots \\
\sigma_1 &= \ldots \\
\text{Interpolant} &\xrightarrow{\delta} \ldots \\
\text{Bad} &\xrightarrow{\delta} \ldots \\
\text{BMC}^k (\text{Interpolant}, \delta, \text{Bad}) &\text{unsat} \\
\text{CTI to } I_0 &\xrightarrow{(_-, \sigma_1)} \\
* k \text{ is a parameter} &\end{align*} \]

[HVC’12] Computing Interpolants without Proofs. Chockler, Ivrii, Matsliah
[LPAR’13] Instantiations, Zippers and EPR Interpolation. Bjørner, Gurfinkel, Korovin, Lahav
$I_1 = \text{Init} \lor \text{Interpolant}$

$I_0 = \text{Init}$

$\sigma_1$ $\delta$ $\text{Interpolant}$ $\delta$ $\text{Bad}$

$\text{BMC}^k (\text{Interpolant}, \delta, \text{Bad}) \text{ unsat}$

$(-, \sigma_1) \text{ CTI to } I_0$

* $k$ is a parameter

[HVC’12] Computing Interpolants without Proofs. Chockler, Ivrii, Matsliah
[LPAR’13] Instantiations, Zippers and EPR Interpolation. Bjørner, Gurfinkel, Korovin, Lahav
Model-Based Interpolation

**Init:**
\[(x_1, ..., x_n) := 0 \ldots 0\]

**Bad:**
\[(x_1, ..., x_n) = 1 \ldots 1\]

\[\delta:\]
\[y_1, ..., y_n := \ast\]
\[x_1, ..., x_n := (x_1, ..., x_n) + 2 \cdot (y_1, ..., y_n) \pmod{2^n}\]

\[\text{Interpolant}_1 = (x_1 = 0 \land x_2 = 1 \land \ldots \land x_{n-1} = 1 \land x_n = 0)\]

\[\sigma_1 = 01 \ldots 10\]

\[k \text{ times}\]

**I** \(\xrightarrow{\delta} \) \(\sigma_1\) \(\xrightarrow{\delta} \ldots \xrightarrow{\delta} \text{Bad}\)

\[I_0 = \text{Init}\]

\[(_, \sigma_1) \text{ CTI to } I_0\]

[HVC’12] Computing Interpolants without Proofs. Chockler, Ivrii, Matsliah

[LPAR’13] Instantiations, Zippers and EPR Interpolation. Bjørner, Gurfinkel, Korovin, Lahav
Model-Based Interpolation

**Init:**
\[(x_1, \ldots, x_n) := 0 \ldots 0\]

**Bad:**
\[(x_1, \ldots, x_n) = 1 \ldots 1\]

**δ:**
\[y_1, \ldots, y_n := *\]
\[x_1, \ldots, x_n := (x_1, \ldots, x_n) + 2 \cdot (y_1, \ldots, y_n) \pmod{2^n}\]

**σ_1:**
\[01 \ldots 10\]

**Interpolant_1:**
\[(x_1 = 0 \land x_2 = 1 \land \ldots \land x_{n-1} = 1 \land x_n = 0)\]

[HVC’12] Computing Interpolants without Proofs. Chockler, Ivrii, Matsliah
[LPAR’13] Instantiations, Zippers and EPR Interpolation. Bjørner, Gurfinkel, Korovin, Lahav
Model-Based Interpolation

\[
\begin{align*}
\text{Init:} & \quad (x_1, \ldots, x_n) := 0 \ldots 0 \\
\text{Bad:} & \quad (x_1, \ldots, x_n) = 1 \ldots 1
\end{align*}
\]

\[
\begin{align*}
\delta: & \quad y_1, \ldots, y_n := * \\
& \quad x_1, \ldots, x_n := (x_1, \ldots, x_n) + 2 \cdot (y_1, \ldots, y_n) \pmod{2^n}
\end{align*}
\]

Interpolant_{1} = (x_1 = 0 \land x_2 = 1 \land \ldots \land x_{n-1} = 1 \land x_n = 0)

\[
\sigma_1 = 01 \ldots 10
\]

\[
I_0 = \text{Init}
\]

\[
\delta
\]

[HVC’12] Computing Interpolants without Proofs. Chockler, Ivrii, Matsliah

[LPAR’13] Instantiations, Zippers and EPR Interpolation. Bjørner, Gurfinkel, Korovin, Lahav
Model-Based Interpolation

\[ \text{Init:} \quad (x_1, \ldots, x_n) := 0 \ldots 0 \]

\[ \text{Bad:} \quad (x_1, \ldots, x_n) = 1 \ldots 1 \]

\[ \delta: \quad \begin{align*}
y_1, \ldots, y_n & := * \\
x_1, \ldots, x_n & := (x_1, \ldots, x_n) + 2 \cdot (y_1, \ldots, y_n) \pmod{2^n} \end{align*} \]

\[ \text{Interpolant}_1 = (x_1 = 0 \land x_2 = 1 \land \ldots \land x_{n-1} = 1 \land x_n = 0) \]

\[ \sigma_1 = 01 \ldots 10 \]

\[ I_0 = \text{Init} \]

[HVC’12] Computing Interpolants without Proofs. Chockler, Ivrii, Matsliah
[LPAR’13] Instantiations, Zippers and EPR Interpolation. Bjørner, Gurfinkel, Korovin, Lahav
Model-Based Interpolation

\[\text{Init:} \quad (x_1, \ldots, x_n) := 0 \ldots 0\]

\[\text{Bad:} \quad (x_1, \ldots, x_n) = 1 \ldots 1\]

\[\sigma_1 = 01 \ldots 10\]

\[\delta: \quad y_1, \ldots, y_n := * \]
\[x_1, \ldots, x_n := (x_1, \ldots, x_n) + 2 \cdot (y_1, \ldots, y_n) \pmod{2^n}\]

\[\text{Interpolant}_1 = (x_1 = 0 \land x_2 = 1 \land \cdots \land x_{n-1} = 1 \land x_n = 0)\]

\[I_0 = \text{Init}\]

[HV'C'12] Computing Interpolants without Proofs. Chockler, Ivrii, Matsliah

[LPAR’13] Instantiations, Zippers and EPR Interpolation. Bjørner, Gurfinkel, Korovin, Lahav
Model-Based Interpolation

\textbf{Init:}\hspace{1em}(x_1,\ldots,x_n) := 0 \ldots 0

\textbf{Bad:}\hspace{1em}(x_1,\ldots,x_n) = 1 \ldots 1

\[ y_1,\ldots,y_n := * \]

\[ x_1,\ldots,x_n := (x_1,\ldots,x_n) + 2 \cdot (y_1,\ldots,y_n) \pmod{2^n} \]

\text{Interpolant}_1 = (x_1 = 0 \land x_2 = 1 \land \ldots \land x_{n-1} = 1 \land x_n = 0)

\sigma_1 = 01 \ldots 10

\[ I_0 = \text{Init} \]

[HVC'12] Computing Interpolants without Proofs. Chockler, Ivrii, Matsliah

[LPAR'13] Instantiations, Zippers and EPR Interpolation. Bjørner, Gurfinkel, Korovin, Lahav
Model-Based Interpolation

Init:
\((x_1, ..., x_n) := 0 \ldots 0\)

Bad:
\((x_1, ..., x_n) = 1 \ldots 1\)

\[ I = Init \lor (x_n = 0) \]

\[ \begin{align*}
\delta: & \quad y_1, ..., y_n := * \\
& \quad x_1, ..., x_n := (x_1, ..., x_n) + 2 \cdot (y_1, ..., y_n) \pmod{2^n}
\end{align*} \]

\[ I_0 = Init \]

\[ I \xrightarrow{\delta} \sigma_1 \xrightarrow{\delta} \ldots \xrightarrow{\delta} Bad \]

[HVC’12] Computing Interpolants without Proofs. Chockler, Ivrii, Matsliah
[LPAR’13] Instantiations, Zippers and EPR Interpolation. Bjørner, Gurfinkel, Korovin, Lahav
Model-Based Interpolation

Inferring invariant in DNF:

\[
\left( \ell_1^1 \land \cdots \land \ell_{k_1}^1 \right) \lor \cdots \lor \left( \ell_1^m \land \cdots \land \ell_{k_m}^m \right)
\]

\[
\text{gen}(\sigma_1) \quad \text{gen}(\sigma_m)
\]

ITP-k:

\[
I := \text{false}
\]

while (\_, \sigma') counterexample to \text{Inductive}(\delta, I):

\[
I := I \lor \text{generalize}(\sigma')
\]

generalize(\sigma'):

drop literals from \sigma'

while \text{BMC}^k(\sigma', \delta, \text{Bad}) unsat
Inductiveness-Query Model

Inference algorithm

\( \alpha_1 \) inductive?\
\( \checkmark / \times + \text{counterexample} \)

\( \alpha_m \) inductive?\
\( \checkmark / \times + \text{counterexample} \)

Inductiveness-query oracle

\( \{ \alpha_i \} \)

\( \delta \)

\{ \alpha_i \}

\( I \) := false

while \((\_ , \sigma')\) counterexample to \text{Inductive}(\delta, I):

\( I := I \lor \text{generalize}(\sigma') \)

\text{generalize}(\sigma'):

drop literals from \( \sigma' \)

while \( \text{BMC}^k(\sigma', \delta, \text{Bad}) \) unsat

\( I \) inductive?\
\( \checkmark / \times \)

?
Hoare-Query Model

inference algorithm

\{\alpha_1\} \delta^k \{\beta_1\}? 

✓ / X + counterexample

\ldots

\{\alpha_m\} \delta^k \{\beta_m\}? 

✓ / X + counterexample

Hoare-query oracle

\{\alpha_i\} \delta^k \{\beta_i\}

BMC^k (\alpha_i, \delta, \neg \beta_i) unsat?

Trace (\sigma_0, \ldots, \sigma_k) of \delta s.t.
\sigma_0 \models \alpha_i, \quad \sigma_k \models \neg \beta_i

Capable of modeling several interesting algorithms
Hoare-Query Model

ITP-\(k\): 

\( I := false \)

while \((\_ , \sigma')\) counterexample to Inductive(\(\delta, I\)):

\( I := I \lor \text{generalize}(\sigma') \)

generalize(\(\sigma'\)):

- drop literals from \(\sigma'\)
- while \(\text{BMC}^k(\sigma', \delta, \text{Bad})\) unsat

\(\{I\} \delta \{I\}\)?

\(\checkmark / \times\)

\(\{\sigma'\} \delta^k \{\neg \text{Bad}\}\)?

\(\checkmark / \times\)

Also captures IC3/PDR
Outline

Invariant Inference

- Query-based learning models for invariant inference
- Complexity lower and upper bounds for each model
- Invariant inference is harder than concept learning
- Complexity results for invariant inference algorithms from concept learning algorithms

Exact Concept Learning
Hoare-Query Complexity

**Thm:** Every Hoare-query algorithm requires $2^{\Omega(n)}$ queries in the worst case for inferring $I \in \text{DNF}$ s.t. $|I| \leq \text{poly}(n)$

$n$ is the vocabulary size, $k = \text{poly}(n)$

- even with *unlimited* computational power
- *unconditional* lower bound

(Decision problem is $\Sigma_2^P$-complete.)

[CADE’09] Complexity and Algorithms for Monomial and Clausal Predicate Abstraction. Lahiri, Qadeer

[POPL’20] Complexity and Information in Invariant Inference. Feldman, Immerman, Shoham, Sagiv
Hoare-Query Complexity

**Thm**: Every Hoare-query algorithm requires $2^{\Omega(n)}$ queries in the worst case for inferring $I \in \text{DNF}$ s.t. $|I| \leq \text{poly}(n)$

$n$ is the vocabulary size, $k = \text{poly}(n)$

- even with **unlimited** computational power
- **unconditional** lower bound

Lower bound holds also for inferring **monotone** DNF invariants $I \in \text{MonDNF}$ s.t. $|I| \leq \text{poly}(n)$

variables appear only positively

[POPL’20] Complexity and Information in Invariant Inference. Feldman, Immerman, Shoham, Sagiv
Hoare > Inductiveness

Thm: There exists a class of transition systems $\mathcal{P}$, so that for solving inference:
1. $\exists$ Hoare-query algorithm (with $k=1$) with $\text{poly}(n)$ queries
2. $\forall$ inductiveness-query algorithm requires $2^{\Omega(n)}$ queries

[POPL’20] Complexity and Information in Invariant Inference. Feldman, Immerman, Shoham, Sagiv
Thm: There exists a class of transition systems $\mathcal{P}$, so that for solving inference:

1. \exists Hoare-query algorithm (with $k=1$) with $\text{poly}(n)$ queries
2. \forall inductiveness-query algorithm requires $2^{\Omega(n)}$ queries

Proof:

$\mathcal{P} = $ maximal transition systems for monotone DNF with $n$ terms

Maximal system for $\varphi$:
Hoare > Inductiveness

Upper bound:
\[ \exists \text{ Hoare-query algorithm (with } k=1 \text{) with } \text{poly}(n) \text{ queries} \]

Proof: \textbf{ITP-1} takes \( O(n^2) \) queries

\[
I := \text{false}
\]

while (\_, \sigma') counterexample to \textbf{Inductive}(\delta, I):
\[
I := I \lor \text{generalize}(\sigma')
\]

\text{generalize}(\sigma'):
\[
\sigma' \Rightarrow \varphi
\]

while \( \text{BMC}^1(\sigma', \delta, \text{Bad}) \) unsat

\[ \varphi = x_1 \lor (x_2 \land x_3) \]

\( \varphi \) is monotone

1 iteration 1 iteration
Hoare > Inductiveness

Lower bound:
∀ inductiveness-query algorithm requires $2^{\Omega(n)}$ queries

Proof:
inference algorithm

\[ \alpha_1 \text{ inductive?} \]

\[ X + \text{counterexample} \]

\[ \ldots \]

\[ \alpha_m \text{ inductive?} \]

\[ X + \text{counterexample} \]

\[ \{\alpha_i\} \]

\[ \delta \]

\[ \{\alpha_i\} \]

\[ \varphi \]

\[ \neg \varphi \]

\[ \text{Bad} \]

\[ \leq \]

\[ \varphi \]

\[ \neg \varphi \]

\[ \text{Bad} \]

\[ \text{Init} \]

\[ \alpha_i \]
Hoare > Inductiveness

Lower bound:
\[ \forall \text{ inductiveness-query algorithm requires } 2^{\Omega(n)} \text{ queries} \]

Proof:

\[ 2^{\Omega(n)} \leq \]

previous theorem

general systems
monotone DNF invariants

\[ \phi \]
\[ \neg \phi \]

Bad

\[ \leq \]

maximal systems
monotone DNF invariants

\[ \phi \]
\[ \neg \phi \]

Bad

\[ \leq \]

Init
Thm: There exists a class of transition systems $\mathcal{P}$, so that for solving inference:

1. $\exists$ Hoare-query algorithm (with $k=1$) with $\text{poly}(n)$ queries
2. $\forall$ inductiveness-query algorithm requires $2^{\Omega(n)}$ queries

Similar proof works with a simple case of IC3/PDR

$\Rightarrow$ ICE cannot model IC3/PDR,

and the extension of [VMCAI’17] is necessary

[POPL’20] Complexity and Information in Invariant Inference. Feldman, Immerman, Shoham, Sagiv

[VMCAI’17] IC3 - Flipping the E in ICE. Vizel, Gurfinkel, Shoham, Malik.
Outline

Invariant Inference

- Query-based learning models for invariant inference
- Complexity lower and upper bounds for each model
- Invariant inference is harder than concept learning
- Complexity results for invariant inference algorithms from concept learning algorithms

Exact Concept Learning
Inferring Monotone DNF

Invariant Inference vs. Exact Concept Learning

Maximal

<table>
<thead>
<tr>
<th>Inductive</th>
<th>General</th>
</tr>
</thead>
<tbody>
<tr>
<td>$2^{\Omega(n)}$</td>
<td>$2^{\Omega(n)}$</td>
</tr>
<tr>
<td>Hoare</td>
<td>poly</td>
</tr>
</tbody>
</table>

General

<table>
<thead>
<tr>
<th>Equiv</th>
<th>sub-exponential</th>
</tr>
</thead>
<tbody>
<tr>
<td>Equiv + mem</td>
<td>poly</td>
</tr>
</tbody>
</table>

[ML’87] Queries and Concept Learning, Angluin
Inductiveness vs. Equivalence Queries

Invariant Inference (learning algorithm)

Exact Concept Learning (oracle)

Counterexamples to induction:
\( \sigma \models \neg \varphi \text{ or } \sigma' \models \varphi \)

Positive/negative examples:
\( \sigma^+ \models \varphi, \sigma^- \models \neg \varphi \)

<table>
<thead>
<tr>
<th></th>
<th>Maximal</th>
<th>General</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inductive</td>
<td>(2^{\Omega(n)})</td>
<td>(2^{\Omega(n)})</td>
</tr>
<tr>
<td>Hoare</td>
<td>poly</td>
<td>(2^{\Omega(n)})</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>Equiv</th>
<th>sub-exponential</th>
</tr>
</thead>
<tbody>
<tr>
<td>Equiv + mem</td>
<td>poly</td>
<td>poly</td>
</tr>
</tbody>
</table>

[ML’87] Queries and Concept Learning, Angluin
Inductiveness vs. Equivalence Queries

**Thm:** Learning from counterexamples to induction is **harder** than learning from positive/negative examples.

Counterexamples to induction:  
\( \sigma \models \lnot \varphi \) or \( \sigma' \models \varphi \)

Positive/negative examples:  
\( \sigma^+ \models \varphi \), \( \sigma^- \models \lnot \varphi \)

<table>
<thead>
<tr>
<th>Inductive</th>
<th>Maximal</th>
<th>General</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>( 2^\Omega(n) )</td>
<td>( 2^\Omega(n) )</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Hoare</th>
<th>Maximal</th>
<th>General</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>( \text{poly} )</td>
<td>( 2^\Omega(n) )</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>Equiv</th>
<th>sub-exponential</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Equiv + mem</td>
<td>poly</td>
</tr>
</tbody>
</table>

[ML’87] Queries and Concept Learning, Angluin  
Inductiveness vs. Equivalence Queries

**Thm:** Learning from counterexamples to induction is **harder** than learning from positive/negative examples.

<table>
<thead>
<tr>
<th>Counterexamples to induction:</th>
<th>Positive/negative examples:</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\sigma \models \neg \varphi$ or $\sigma' \models \varphi$</td>
<td>$\sigma^+ \models \varphi$, $\sigma^- \models \neg \varphi$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>Maximal</th>
<th>General</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inductive</td>
<td>$2\Omega(n)$</td>
<td>$2\Omega(n)$</td>
</tr>
<tr>
<td>Hoare</td>
<td>poly</td>
<td>$2\Omega(n)$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>Equiv</th>
<th>sub-exponential</th>
</tr>
</thead>
<tbody>
<tr>
<td>Equiv + mem</td>
<td>poly</td>
<td></td>
</tr>
</tbody>
</table>

[ML’87] Queries and Concept Learning, Angluin
Invariant Inference with Equivalence & Membership Queries

Invariant Inference

<table>
<thead>
<tr>
<th></th>
<th>Maximal</th>
<th>General</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inductive</td>
<td>$2\Omega(n)$</td>
<td>$2\Omega(n)$</td>
</tr>
<tr>
<td>Hoare</td>
<td>poly</td>
<td>$2\Omega(n)$</td>
</tr>
</tbody>
</table>

Exact Concept Learning

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Equiv</td>
<td>sub-exponential</td>
<td></td>
</tr>
<tr>
<td>Equiv + mem</td>
<td>poly</td>
<td></td>
</tr>
</tbody>
</table>

[ML’87] Queries and Concept Learning, Angluin
**Invariant Inference with Equivalence & Membership Queries**

**Thm.** In general, in the Hoare-query model, **no efficient way** to implement a teacher for equivalence and membership queries.

---

### Invariant Inference

<table>
<thead>
<tr>
<th></th>
<th>Maximal</th>
<th>General</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inductive</td>
<td>$2^{\Omega(n)}$</td>
<td>$2^{\Omega(n)}$</td>
</tr>
<tr>
<td>Hoare</td>
<td>poly</td>
<td>$2^{\Omega(n)}$</td>
</tr>
</tbody>
</table>

---

### Exact Concept Learning

<table>
<thead>
<tr>
<th></th>
<th>Equiv</th>
<th>sub-exponential</th>
</tr>
</thead>
<tbody>
<tr>
<td>Equiv + mem</td>
<td>poly</td>
<td></td>
</tr>
</tbody>
</table>

---

[ML’87] Queries and Concept Learning, Angluin
[POPL’20] Complexity and Information in Invariant Inference. Feldman, Immerman, Shoham, Sagiv
Outline

Invariant Inference

- Query-based learning models for invariant inference
- Complexity lower and upper bounds for each model
- Invariant inference is harder than concept learning

Exact Concept Learning

vs.

- Complexity results for invariant inference algorithms from concept learning algorithms
From Learning to Inference

learning algorithm

is it $\psi_1$?

✓ / X + counterexample

is it $\psi_2$?

✓ / X + counterexample

does $\sigma_3 \models \_?\$

✓ / X

Membership

Equivalence

oracle

Need to implement this
From Learning to Inference

\[ I := \text{false} \]

\text{while } (\_ , \sigma') \text{ counterexample to } \text{Inductive}(I) : \quad
\[ I := I \lor \text{generalize}(\sigma') \]

\text{generalize}(\sigma') : \quad
\text{drop literals from } \sigma' \quad
\text{while } \text{BMC}^k(\sigma', \delta, \text{Bad}) \text{ unsat}

[CAV’03] Interpolation and SAT-Based Model Checking, McMillan
[HVC’12] Computing Interpolants without Proofs. Chockler, Ivrii, Matsliah
From Learning to Inference

Exact learning DNF formulas

\[
\psi := \text{false} \\
\text{while } \sigma' \text{ counterexample to } \text{Equivalence}(\psi): \\
\psi := \psi \lor \text{generalize}(\sigma')
\]

\text{generalize}(\sigma'):
\begin{align*}
\text{drop literals from } \sigma' \\
\text{while } \text{Membership}(\sigma') = \text{✓}
\end{align*}

Inferring DNF invariants

\[
I := \text{false} \\
\text{while } (\_, \sigma') \text{ counterexample to } \text{Inductive}(I): \\
I := I \lor \text{generalize}(\sigma')
\]

\text{generalize}(\sigma'):
\begin{align*}
\text{drop literals from } \sigma' \\
\text{while } \text{BMC}^k(\sigma', \delta, \text{Bad}) \text{ unsat}
\end{align*}

[CACM’84] A Theory of the Learnable. Valiant
[ML’87] Queries and Concept Learning. Angluin
[ML’95] On the Learnability of Disjunctive Normal Form Formulas. Aizenstein and Pitt

[CAV’03] Interpolation and SAT-Based Model Checking, McMillan
[HVC’12] Computing Interpolants without Proofs. Chockler, Ivrii, Matsliah
From Learning to Inference

Exact learning DNF formulas $\Rightarrow$ Inferring DNF invariants

$$\psi := \text{false}$$
while $\sigma'$ counterexample to $\text{Equivalence}(\psi)$:
$$\psi := \psi \lor \text{generalize}(\sigma')$$

$\text{generalize}(\sigma')$:
drop literals from $\sigma'$
while $\text{Membership}(\sigma') = \checkmark$

$\text{generalize}(\sigma')$:
drop literals from $\sigma'$
while $\text{BMC}^k(\sigma', \delta, \text{Bad})$ unsat

$\text{Inductive}(I)$:

$\psi$ $\Rightarrow$ $\text{BMC}^k(\sigma', \delta, \text{Bad})$ unsat

References:

[ML'87] Queries and Concept Learning. Angluin
[ML'95] On the Learnability of Disjunctive Normal Form Formulas. Aizenstein and Pitt
[CAV'03] Interpolation and SAT-Based Model Checking, McMillan
[HVC’12] Computing Interpolants without Proofs. Chockler, Ivrii, Matsliah
From Learning to Inference

Efficiently

Exact learning DNF formulas

Inferring DNF invariants

ψ := false
while σ' counterexample to Equivalence(ψ, I) do
ψ := ψ ∨ generalize(σ')
I := I ∨ generalize(σ')
generalize(σ'):
  drop literals from σ'
while Membership(σ') = ✓

Efficiently

When and how is it possible to implement queries?

generalize(σ'):
  drop literals from σ'
while BMC^k(σ', δ, Bad) unsat

[ML'87] Queries and Concept Learning. Angluin
[ML'95] On the Learnability of Disjunctive Normal Form Formulas. Aizenstein and Pitt

[CAV’03] Interpolation and SAT-Based Model Checking, McMillan
[HVC’12] Computing Interpolants without Proofs. Chockler, Ivrii, Matsliah
From Learning to Inference

Exact learning formulas ⟹ Inferring invariants

Thm: can implement mem & equiv queries when
• the invariant is $k$-fenced, and
• the algorithm’s queries are one-sided

One-Sided Equivalence($\psi$): $\psi \Rightarrow \varphi$
One-Sided Membership($\sigma$): $\sigma \in \varphi \cup \partial^-(\varphi)$

[POPL’21] Learning the Boundary of Inductive Invariants. Feldman, Sagiv, Shoham, Wilcox
From Learning to Inference

Exact **learning** formulas $\implies$ Inferring invariants

**Thm:** can implement mem & equiv queries when
• the invariant is *k*-fenced, and
• the algorithm’s queries are one-sided

[POPL’21] Learning the Boundary of Inductive Invariants. Feldman, Sagiv, Shoham, Wilcox
$k$-Fenced Invariants

$I^*$ is $k$-fenced if all the states in $\partial^-(I^*)$ can reach a bad state in at most $k$ steps

Outer boundary: all states $\sigma \notin I^*$ that differ from some $\sigma' \in I^*$ in one bit
$k$-Fenced Invariants

$I^*$ is $k$-fenced if all the states in $\partial^- (I^*)$ can reach a bad state in at most $k$ steps.

Outer boundary: all states $\sigma \not\in I^*$ that differ from some $\sigma' \in I^*$ in one bit.
$k$-Fenced Invariants

$I^*$ is $k$-fenced if all the states in $\partial^- (I^*)$ can reach a bad state in at most $k$ steps.

Outer boundary
all states $\sigma \notin I^*$ that differ from some $\sigma' \in I^*$ in one bit
**Init:**

\[(x_1, \ldots, x_n) := 0 \ldots 0\]

**Bad:**

\[(x_1, \ldots, x_n) = 1 \ldots 1\]

\[\delta:\]

\[y_1, \ldots, y_n := *\]

\[x_1, \ldots, x_n := (x_1, \ldots, x_n) + 2 \cdot (y_1, \ldots, y_n) \pmod{2^n}\]

**Outer boundary**

all the states in \(I^*\) that differ from some \(\sigma' \in I^*\) in one bit

**\(I^*\):**

\[x_n \neq 1\]

1-fenced:

all the states in \(\partial^{-}(I^*) = \{x_n = 1\}\)

can reach a bad state in at most 1 steps

\(I^*\) can reach a bad state in at most \(k\) steps
**Efficient Inference**

**Corollary:** Efficient inference of $k$-fenced invariants in:
- **Monotone DNF**, via Angluin’87
- **Almost-monotone DNF**, via Bshouty’95

$O(1)$ terms with negated variables

[ML’87] Queries and Concept Learning. Angluin
Corollary: Efficient inference of \(k\)-fenced invariants in:
- Monotone DNF, via Angluin’87
- Almost-monotone DNF, via Bshouty’95

What about Bshouty’s CDNF algorithm?
- short DNF and short CNF representation
e.g., expressible as a short decision tree

Efficient Inference

**Corollary:** Efficient inference of $k$-fenced invariants in:
- Monotone DNF, via Angluin’87
- Almost-monotone DNF, via Bshouty’95

What about Bshouty’s CDNF algorithm?
- short DNF and short CNF representation
  - e.g., expressible as a short decision tree

[POPL’21] Learning the Boundary of Inductive Invariants. Feldman, Sagiv, Shoham, Wilcox
Efficient Inference

Corollary: Efficient inference of two-sided $k$-fenced invariants with a short \textit{DNF and} short \textit{CNF} representation e.g., expressible as a short decision tree, via Bshouty’95

[POPL’21] Learning the Boundary of Inductive Invariants. Feldman, Sagiv, Shoham, Wilcox

Efficient Inference

Corollary: Efficient inference of two-sided $k$-fenced invariants with a short DNF and short CNF representation, e.g., expressible as a short decision tree, via Bshouty’95

Thm: efficient inference of CDNF invariants also when they are one-sided $k$-fenced
not by transformation from learning

[SAS ‘22] Invariant Inference With Provable Complexity From the Monotone Theory. Feldman, Shoham
Conclusion

Invariant Inference vs. Exact Concept Learning

- Query-based learning models for invariant inference
- Complexity lower and upper bounds for each model
- Invariant inference is harder than concept learning
- Complexity results for invariant inference algorithms from concept learning algorithms