Towards a Theory of Learning Inductive Invariants

Yotam Feldman  Neil Immerman  Mooly Sagiv

Sharon Shoham  James R. Wilcox

@yotamfe, @SagivMooly, @wilcoxjay
Init: \[ (x_1, \ldots, x_n) := 0 \ldots 0 \]

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

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

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

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

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

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

Not inductive:

\[
\begin{array}{c}
00 \ldots 1 \xrightarrow{\delta} 11 \ldots 1
\end{array}
\]

\[
\begin{array}{c}
I \quad \dashv
\end{array}
\]

\[
I: x_n \neq 1
\]

Inductive:

\[
\begin{array}{c}
x_n \neq 1 \xrightarrow{\delta} I
\end{array}
\]
Inductive Invariants

Goal:
Find inductive invariants automatically

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

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

Not inductive:
\[
\begin{array}{c}
\text{00 ... 1} \\
\text{I} \\
\delta \\
\rightarrow \\
\text{11 ... 1} \\
\neg \text{I}
\end{array}
\]

Inductive:
\[
\begin{array}{c}
x_n \neq 1 \\
\text{I} \\
\delta \\
\rightarrow \\
x_n \neq 1 \\
\text{I}
\end{array}
\]
This Work

Invariant Inference vs. Exact Concept Learning

- Query-based learning models for invariant inference
- Invariant inference is harder than concept learning
- Complexity results for invariant inference algorithms from classification algorithms
Boolean transition systems, $\Sigma = \{ p_1, \ldots, p_n \}$

Given a transition system from a class $\mathcal{P}$ (over $\Sigma$),
Find an inductive invariant

\[ I \in \text{DNF} \quad |I| \leq \text{poly}(n) \]

(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
Interpolation-Based Inference

\[ I = \text{Init} \]

\[ k + 1 \text{ times} \]

[CAV’03] Interpolation and SAT-Based Model Checking, McMillan
Interpolation-Based Inference

\[ I = \text{Init} \]

[CAV’03] Interpolation and SAT-Based Model Checking, McMillan
Interpolation-Based Inference

\[ I = \text{Init} \]

[CAV’03] Interpolation and SAT-Based Model Checking, McMillan
Interpolation-Based Inference

\[ I = \text{Init} \lor \text{Interpolant} \]

[CAV’03] Interpolation and SAT-Based Model Checking, McMillan
Interpolation-Based Inference

\[ I = \text{Init} \lor \text{Interpolant} \]

[CAV’03] Interpolation and SAT-Based Model Checking, McMillan
Interpolation-Based Inference

\[ I = \text{Init} \lor \text{Interpolant} \]

[CAV’03] Interpolation and SAT-Based Model Checking, McMillan
Model-Based Interpolation

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

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

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

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

\[\sigma_1 = 01 \ldots 10\]

\[k\text{ times}\]

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

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

Interpolant_1 = \(x_1 = 0 \land x_2 = 1 \land \ldots \land x_n = 1 \land x_n = 0\)

\[\sigma_1 = 01 \ldots 10\]

\[\delta(I)\]

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

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

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

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

\[
\sigma_1 = 01 \ldots 10
\]

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

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

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

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

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

\[
\sigma_1 = 01 \ldots 10
\]

Interpolant_1 = (x_1 = 0 \land x_2 = 1 \land \ldots \land x_n = 1 \land x_n = 0)

\[\text{Init} = I\]

\[\delta(I)\]

\[\sigma_1\]

\[\text{Bad}\]

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

\[\sigma_1 = 01 \ldots 10\]

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

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

\[\delta(I)\]

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

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

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

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

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

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

\[ \sigma_1 = 01 \ldots 10 \]

[\text{HVC’12}] Computing Interpolants without Proofs. Chockler, Ivrii, Matsliah

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

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

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

\[
I = \text{Init} \lor (x_n = 0)
\]

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

Inferring invariant in DNF:

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

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

\begin{align*}
I & := \text{false} \\
\text{while } (_, \sigma') \text{ counterexample to Inductive}(\delta, I): \\
\text{generalize}(\sigma') :& \quad \text{drop literals from } \sigma' \\
& \quad \text{while } \text{BMC}(\delta, \sigma', k) \cap \text{Bad} = \emptyset
\end{align*}
Model-Based Interpolation

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

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

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

\textbf{generalize}(\(\sigma'\)):

drop literals from \(\sigma'\)

while \(\text{BMC}(\delta, \sigma', k) \cap \text{Bad} = \emptyset\)
Understanding Invariant Inference

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

\text{generalize}(\sigma'):
drop literals from \sigma'
while \text{BMC}(\delta, \sigma', k) \cap \text{Bad} = \emptyset
Understanding Invariant Inference

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

\text{generalize}(\sigma'):
- drop literals from \(\sigma'\)
while \(\text{BMC}(\delta, \sigma', k) \cap \text{Bad} = \emptyset\)

Complexity bounds from exact classification algorithms
Rich SAT queries allow exponentially faster inference
Understanding Invariant Inference

Invariant Inference

Complexity bounds from exact classification algorithms

Exact Concept Learning

Rich SAT queries allow exponentially faster inference
Exact Concept Learning with Equivalence & Membership Queries

learning algorithm

is it $\psi_1$?

✓ / X + counterexample

is it $\psi_2$?

✓ / X + counterexample

does $\sigma_3 \models ?$

✓ / X

... Membership Equivalence

oracle

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

learning algorithm

is it $\psi_1$?

✓ / X + counterexample

is it $\psi_2$?

✓ / X + counterexample

does $\sigma_3 \models ?$

✓ / X

...
Inductiveness-Query Model

Algorithms cannot access the transition relation directly, only perform inductiveness queries

Complexity: # inductiveness queries worst case amongst possible counterexamples

Inductiveness-Query Model

inference algorithm

\[ \alpha_1 \text{ inductive?} \]
\[ \checkmark / \times + \text{counterexample} \]

\[ \cdots \]

\[ \alpha_m \text{ inductive?} \]
\[ \checkmark / \times + \text{counterexample} \]

inductiveness-query oracle

\( \delta \)
\( \{ \alpha_i \} \)

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

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

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

\text{generalize}(\sigma'):

drop literals from \( \sigma' \)

while \( \text{BMC}(\delta, \sigma', k) \cap \text{Bad} = \emptyset \)

\[ I \text{ inductive?} \]
\[ \checkmark / \times \]

?
Hoare-Query Model

Algorithms cannot access the transition relation directly, only perform Hoare queries.
Capable of modeling several interesting algorithms

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

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

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

\text{generalize}(\sigma'):

- drop literals from \( \sigma' \)

\[ \text{while } \text{BMC}(\delta, \sigma', 1) \cap \text{Bad} = \emptyset \]
Hoare > Inductiveness

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

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

A simple case of IC3/PDR

$\Rightarrow$ ICE cannot model 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.
Hoare > Inductiveness

Thm: There exists a class of transition systems \( \mathcal{P} \), so that for solving polynomial-length inference:

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

\[
\begin{align*}
I & := \text{false} \\
\text{while (}, \sigma', \text{) counterexample to Inductive}(\delta, I): & \\
& \quad I := I \lor \text{generalize}(\sigma') \\
\text{generalize}(\sigma'): & \\
& \quad \text{drop literals from } \sigma' \\
& \quad \text{while } \text{BMC}(\delta, \sigma', 1) \cap \text{Bad} = \emptyset
\end{align*}
\]

\{\sigma'}{\text{Bad}}? \quad \checkmark / \ x \quad \ldots
Hoare > Inductiveness

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

1. $\exists$ Hoare-query algorithm with $\text{poly}(n)$ queries
2. $\forall$ inductiveness-query algorithm requires $2^{\Omega(n)}$ queries
Learning from Counterexamples to Equivalence Queries

**Thm:** Learning from counterexamples to induction is **harder** than learning from labeled examples.

Positive/negative examples:

\[ \sigma^+ \models \varphi, \sigma^- \models \neg \varphi \]

Counterexamples to induction:

\[ \sigma \models \neg \varphi \text{ or } \sigma' \models \varphi \]

Learning monotone DNF: subexponential

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

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

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

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

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

\text{generalize}(\sigma'):

- drop literals from \sigma'
- while \text{BMC}(\delta, \sigma', k) \cap \text{Bad} = \emptyset

Complexity bounds from exact classification algorithms

Rich SAT queries allow exponentially faster inference
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

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

Exact learning DNF formulas

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

\text{generalize}(\sigma'):
\begin{enumerate}
  \item drop literals from \( \sigma' \)
  \item while \( \text{Membership}(\sigma') = \checkmark \)
\end{enumerate}

[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
From Learning to Inference

Exact learning
DNF formulas

\[ \psi := \text{false} \]
\[ \text{while } \sigma' \text{ counterexample to } \text{Equivalence}(\psi): \]
\[ \psi := \psi \lor \text{generalize}(\sigma') \]
\[ \text{generalize}(\sigma'): \]
\[ \text{drop literals from } \sigma' \]
\[ \text{while } \text{Membership}(\sigma') = \checkmark \]
\[ \text{BMC}(\sigma', k) \cap \text{Bad} = \emptyset \]

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

Exact learning DNF formulas ⟹ 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

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

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

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

\text{generalize}(\sigma'):

drop literals from \( \sigma' \)

while \text{BMC}(\sigma', k) \cap \text{Bad} = \emptyset

[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

Efficiently

Exact learning DNF formulas

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

while \( \sigma' \) counterexample to Equivalent(\( \psi \))

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

The invariant is \( k \)-fenced

Efficiently

Inferring DNF invariants

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

drop literals from \( \sigma' \)

while Membership(\( \sigma' \)) = \( \checkmark \)

while \( \text{BMC}(\sigma', k) \cap \text{Bad} = \emptyset \)

[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
$k$-Fenced Invariants

\[ (1,1,1) \]

\[ (0,0,0) \]

\[ I^* \]

\[ \neg I^* \]
$k$-Fenced Invariants

$I^*$

$(0,0,0)$

$(1,1,1)$
$k$-Fenced Invariants

\[ \partial^-(I^*) \]
$k$-Fenced Invariants

All the states in $\partial^- (I^*)$ can get to a bad state in at most $k$ steps
Thm. Interpolation-based inference finds an invariant in a polynomial number of SAT queries when

$$\exists I^*.\qquad$$

Fence condition: the Hamming boundary of $I^*$ reaches bad states in $k$ steps

$$+$$

$I^*$ is a short monotone DNF (via Angluin) or
$I^*$ is a short almost-monotone DNF (via Bshouty)

$O(1)$ terms with negated variables

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

Invariant Inference

- Query-based learning models for invariant inference
- Invariant inference is harder than concept learning
- Complexity results for invariant inference algorithms from classification algorithms

Exact Concept Learning

vs.