Synthesis and Inductive Learning – Part 3

Sanjit A. Seshia

EECS Department
UC Berkeley

Acknowledgments:
Susmit Jha, Alexandre Donze, Edward Lee

NSF ExCAPE Summer School
June 23-25, 2015
Questions of Interest for this Tutorial

- How can inductive synthesis be used to solve other (non-synthesis) problems?
  - Reducing a Problem to Synthesis

- How does inductive synthesis compare with machine learning? What are the common themes amongst various inductive synthesis efforts?
  - Oracle-Guided Inductive Synthesis (OGIS) Framework

- Is there a complexity/computability theory for inductive synthesis?
  - Yes! A first step: Theoretical analysis of counterexample-guided inductive synthesis (CEGIS)
Outline for this Lecture Sequence

- Examples of Reduction to Synthesis
  - Specification
  - Verification

- Differences between Inductive Synthesis and Machine Learning

- Oracle-Guided Inductive Synthesis
  - Examples, CEGIS

- Theoretical Analysis of CEGIS
  - Properties of Learner
  - Properties of Verifier

- Demo: Requirement Mining for Cyber-Physical Systems
Theoretical Analysis of OGIS
Language Learning in the Limit

[E.M. Gold, 1967]

- Concept = Formal Language
- Class of languages identifiable in the limit if there is a learning procedure that, for each language in that class, given an infinite stream of strings, will eventually generate a representation of the language.

- Results:
  - Cannot learn regular languages, CFLs, CSLs using just positive witness queries
  - Can learn using both positive & negative witness queries (assuming all examples eventually enumerated)
Query-Based Learning

[Queries and Concept Learning, 1988]
[Queries Revisited, 2004]

- First work on learning based on querying an oracle
  - Supports witness, equivalence, membership, subsumption/subset queries
  - No separate correctness condition or formal specification
  - Focus on proving complexity results for specific concept classes

- Sample results
  - Can learn DFAs in poly time from membership and equivalence queries
  - Cannot learn DFAs or DNF formulas in poly time with just equivalence queries
Revisiting the Comparison

<table>
<thead>
<tr>
<th>Feature</th>
<th>Formal Inductive Synthesis</th>
<th>Machine Learning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Concept/Program Classes</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Learning Algorithms</td>
<td>General-Purpose Solvers</td>
<td>Specialized</td>
</tr>
<tr>
<td>Oracle-Guidance</td>
<td>Exact, w/ Formal Spec</td>
<td>Rare (black-box oracles)</td>
</tr>
<tr>
<td></td>
<td>Approximate, w/ Cost Function</td>
<td></td>
</tr>
</tbody>
</table>

What can we prove about convergence/complexity of formal inductive synthesis for:
- General concept classes (e.g., recursive languages)
- Different properties of “general-purpose” learners
- Different properties of (non black-box) oracles
CEGIS: Theoretical Analysis
Query Types for CEGIS

**LEARNER**
- Positive Witness:
  \[ x \in \phi, \text{ if one exists, else } \bot \]

**ORACLE**
- Equivalence: Is \( f = \phi \)?
  - Yes / No + \( x \in \phi \oplus f \)
- Subsumption: Is \( f \subseteq \phi \)?
  - Yes / No + \( x \in f \setminus \phi \)

- Finite memory vs Infinite memory

- Type of counter-example given

Concept class: Any set of recursive languages
Questions

- **Convergence:** How do properties of the learner and oracle impact convergence of CEGIS? (learning in the limit for infinite-sized concept classes)

- **Sample Complexity:** For finite-sized concept classes, what upper/lower bounds can we derive on the number of oracle queries, for various CEGIS variants?
Problem 1: Bounds on Sample Complexity
Teaching Dimension

[Goldman & Kearns, ‘90, ‘95]

- The minimum number of (labeled) examples a teacher must reveal to uniquely identify any concept from a concept class
Teaching a 2-dimensional Box

What about N dimensions?
Teaching Dimension

- The *minimum* number of (labeled) examples a teacher must reveal to *uniquely* identify any concept from a concept class

\[ TD(C) = \max_{c \in C} \min_{\sigma \in \Sigma(c)} |\sigma| \]

where

- \( C \) is a concept class
- \( c \) is a concept
- \( \sigma \) is a teaching sequence (uniquely identifies concept \( c \))
- \( \Sigma \) is the set of all teaching sequences
Theorem: \( TD(C) \) is lower bound on Sample Complexity

- CEGIS: TD gives a lower bound on \#counterexamples needed to learn any concept
- Finite TD is necessary for termination
  - If \( C \) is finite, \( TD(C) \leq |C|-1 \)
- Finding Optimal Teaching Sequence is NP-hard (in size of concept class)
  - But heuristic approach works well ("learning from distinguishing inputs")
- Open Problems: Compute TD for common classes of SyGuS problems

[see Jha & Seshia, 2015]
Problem 2: Convergence of Counterexample-guided loop with positive witness and membership/subsumption queries
Learning $-1 \leq x \leq 1 \land -1 \leq y \leq 1$

($C = \text{Boxes around origin}$)

Arbitrary Counterexamples may not work for Arbitrary Learners
Learning $-1 \leq x, y \leq 1$ from Minimum Counterexamples (dist from origin)
Types of Counterexamples

Assume there is a function $\text{size}: D \rightarrow \mathbb{N}$
- Maps each example $x$ to a natural number
- Imposes total order amongst examples

- **CEGIS:** Arbitrary counterexamples
  - Any element of $f \oplus \phi$

- **MinCEGIS:** Minimal counterexamples
  - A least element of $f \oplus \phi$ according to $\text{size}$
  - Motivated by debugging methods that seek to find small counterexamples to explain errors & repair
Types of Counterexamples

Assume there is a function $size: D \rightarrow N$

- **CBCEGIS**: Constant-bounded counterexamples (bound $B$)
  - An element $x$ of $f \oplus \phi$ s.t. $size(x) < B$
  - Motivation: Bounded Model Checking, Input Bounding, Context bounded testing, etc.

- **PBCEGIS**: Positive-bounded counterexamples
  - An element $x$ of $f \oplus \phi$ s.t. $size(x)$ is no larger than that of any positive example seen so far
  - Motivation: bug-finding methods that mutate a correct execution in order to find buggy behaviors
Summary of Results

[Jha & Seshia, SYNT’14; TR’15]
Open Problems

- For Finite Domains: What is the impact of type of counterexample and buffer size to store counterexamples on the speed of termination of CEGIS?

- For Specific Infinite Domains (e.g., Boolean combinations of linear real arithmetic): Can we prove termination of CEGIS loop?
Demo: Requirement Mining for a Helicopter Control Model
An Example: Modeling Helicopter Dynamics (taken from textbook available at http://LeeSeshia.org, Chapter 2)
Modeling Physical Motion

Six degrees of freedom:
- Position: x, y, z
- Orientation: pitch, yaw, roll
Feedback Control Problem

A helicopter without a tail rotor, like the one below, will spin uncontrollably due to the torque induced by friction in the rotor shaft.

Control system problem: Apply torque using the tail rotor to counterbalance the torque of the top rotor.
Model of the helicopter

Input is the net torque of the tail rotor and the top rotor. Output is the angular velocity around the $y$ axis.

Parameters of the model are shown in the box. The input and output relation is given by the equation to the right.

\[
\dot{\theta}_y(t) = \dot{\theta}_y(0) + \frac{1}{I_{yy}} \int_{0}^{t} T_y(\tau) d\tau
\]
Proportional controller

\[ e(t) = \psi(t) - \dot{\theta}_y(t) \]
\[ T_y(t) = Ke(t) \]

\[ \dot{\theta}_y(t) = \dot{\theta}_y(0) + \frac{1}{I_{yy}} \int_{0}^{t} T_y(\tau) d\tau \]
\[ = \dot{\theta}_y(0) + \frac{K}{I_{yy}} \int_{0}^{t} (\psi(\tau) - \dot{\theta}_y(\tau)) d\tau \]

Note that the angular velocity appears on both sides, so this equation is not trivial to solve.
Summary of Lecture 3

- Theory: brief background
- Theoretical analysis of OGIS: properties of learner and verifier considered
- Teaching Dimension
- Impact of Counterexample type
- Requirement Mining demo: use of CEGIS on an infinite domain