Combining Induction, Deduction and Structure for Synthesis

Sanjit A. Seshia

Associate Professor
EECS Department
UC Berkeley

Students: S. Jha, B. Brady, J. Kotker, W. Li
Collaborators: R. Bryant, S. Gulwani, A. Rakhlin, A. Tiwari

NSF ExCAPE Kick-off Meeting
June 4, 2012
Main Points

- **Induction + Deduction**
  - Induction: specific examples $\rightarrow$ general rules
    - Learning from examples
  - Deduction: general rules $\rightarrow$ specific conclusions
    - Logical inference and constraint solving

- **Three Synthesis Stories**
  - Purely deductive approach is hard

- **Future Directions**
Reverse Engineering Malware

Obfuscated code:
Input: y  Output: modified value of y

```c
{ a=1; b=0; z=1; c=0;
  while(1) {
    if (a == 0) {
      if (b == 0) { y=z+y; b=~b; c=~c; if (~c) break; }
      else if (b == 0) {z=y << 2; a=~a;}
      else { z=y << 3; a=~a; b=~b;}
    }
  } }
```

What it does:

```plaintext
y = y * 45
```

**CHALLENGE:**

**SPECIFICATION CONTROLLED BY ‘ATTACKER’**

From Conficker Worm

• SAFETY: Room Temperature $x$ must lie between 20 and 22 C.
• OPTIMALITY: Minimize switching between modes to save energy
Reactive Synthesis from LTL

Environment Assumptions

System Requirements

Synthesis

\( \varphi_e \rightarrow \varphi_s \)

CHALLENGE:
ENVIRONMENT SPECIFICATION

Often due to missing environment assumptions!
Common Approach: Sciduction

Structure-Constrained Induction and Deduction

**Inductive** Reasoning
(Active Learning: Generalizing from Examples)

+ 

**Deductive** Reasoning
(“Lightweight” Logical inference & Constraint solving)

+ 

**Structure** Hypotheses
(on artifacts to be synthesized)

+ SKETCH, TEMPLATE, Etc.
Demonstration 1: Reverse Engineering Malware

Obfuscated code:

```c
{ a=1; b=0; z=1; c=0;

while (1) {
    if (a == 0) {
        if (b == 0) { y=z+y; a =~a; 
    
        while ii {1} {
            if (~c) break; }
    
        else if(b == 0) {z=y << 2; a=~a;}
    
        else {
            z=y << 3; a=~a; b=~b;
    
        } }
```}

What it does:
Multiplies y by 45

Loop-free compositions of “components” (+, -, <<, >>, *, if-then-else,...) + Learning from Distinguishing I/O Examples + SMT Solving (bit-vector arithmetic)

FROM CONFICKER WORM

Demonstration 2: Synthesizing Switching Logic for Hybrid Systems

Guards are Hyperboxes

Hyperbox Learning from +/- Examples (safe/unsafe switching states)

Numerical Simulation (constraint solving)

• SAFETY: Room Temperature $x$ must lie between 20 and 22 C.
• OPTIMALITY: Minimize switching between modes to save energy

Papers: S. Jha et al., ICCPS 2010 and EMSOFT 2011.
Demonstration 3: Generating Environment Assumptions for Synthesis from LTL

**Env Assumptions are Restricted GR(1)**

+ **Version Space Learning**
  (from Counterstrategies)

+ **(finite-state) Model Checking**

Often due to missing environment assumptions!

Inductive Strategy at the Top-Level

Structure Hypothesis defines Concept (Program) Class

<table>
<thead>
<tr>
<th>INDUCTIVE ENGINE</th>
<th>Queries</th>
<th>DEDUCTIVE ENGINE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Learning from Examples</td>
<td>← Examples</td>
<td>Logical Inference &amp; Constraint Solving</td>
</tr>
<tr>
<td>Labels for Examples</td>
<td>← Examples</td>
<td></td>
</tr>
<tr>
<td>Verified Program or counter-example</td>
<td>←</td>
<td></td>
</tr>
</tbody>
</table>
Deductive Strategy at the Top-Level

Structure Hypothesis defines Logic & Domain Theory

INDUCTIVE ENGINE

Learning from Examples

Background
Facts

Conjectured Lemmas

DEDUCTIVE ENGINE

Logical Inference & Constraint Solving
Other Demonstrations

- **Synthesis**
  - Optimal Fixed-point code from Floating-point “model” [Jha PhD 2011, in submission]

- **Verification**
  - Timing analysis of embedded software [ICCAD 2008, ACM TECS]
  - Generating abstractions for SMT-based model checking of RTL hardware [FMCAD 2011]
Methodology

Procedure P: Induction + Deduction

\[
\begin{align*}
\dot{x} &= -0.002(x - 16) \\
T &= 0 \\
\end{align*}
\]

\[
\begin{align*}
\dot{x} &= -0.002(x - T) \\
\dot{T} &= 0.1 \\
\end{align*}
\]

\[
\begin{align*}
\dot{x} &= -0.002(x - T) \\
T &= -0.1 \\
\end{align*}
\]

\[
\begin{align*}
\dot{x} &= -0.002(x - T) \\
T &= 0 \\
\end{align*}
\]
Methodology

Procedure P’:
Induction + Deduction

Must allow user to formulate and check multiple hypotheses
Future Directions

- Portfolio of computational engines
  - Inductive, Deductive, combinations
  - Induction + Deduction even “inside” (SMT) solvers

- Choice of engine determined by
  - problem definition
  - structure hypotheses (human input)
  - application domain
  - [way to organize the synthesis competition?]

• Also appearing at DAC 2012