BLACK BOX INVARIANT SYNTHESIS

PRANAV GARG
UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN

JOINT WORK WITH:
P. MADHUSUDAN (UIUC),
CHRISTOF LODING (RWTH AACHEN)
DANIEL NEIDER (RWTH AACHEN)

IN CONSULTATION WITH:
DAN ROTH (UIUC)
AUTOMATED PROGRAM VERIFICATION

- Does a program $P$ meet its safety specification $\varphi$?
- Floyd-Hoare style Deductive Verification

```java
foo(int x) {
    y := x;
    while (x != 0) {
        x := x - 1;
        y := y - 1;
    }
    return y;
}
```

**pre:** true

**post:** $y = 0$

Invariant: $x = y$

To alleviate this burden of annotation $\rightarrow$ Invariant synthesis
Reach – post-closed, contains Init avoids Bad

Adequate Invariant: any post-closed set, contains Init, avoids Bad

--- Many choices; which one to pick?

Learn the “simplest” invariant!
SYNTAX GUIDED SYNTHESIS

Synthesize Inv such that

$$\forall \vec{x}.((pre(\vec{x}) \Rightarrow Inv(\vec{x})) \land (Inv(\vec{x}) \land T(\vec{x}, \vec{x'}) \Rightarrow Inv(\vec{x'})) \land (Inv(\vec{x}) \Rightarrow post(\vec{x})))$$

Can be reduced to Syntax-guided synthesis (SyGus)
BLACK-BOX LEARNING OF INVARIANTS

Program

Constraint Solver
check hypothesis?

Teacher

concrete data-points

H
(hypothesis)

Learner
ACTIVE LEARNING OF INVARIANTS

Program

Constraint Solver
check hypothesis?

Teacher

concrete data-points

Learner

H
(hypothesis)

Init

Reach^k

Bad
ACTIVE LEARNING OF INVARIANTS

Program

Constraint Solver
check hypothesis?

Teacher

Learner

concrete data-points

H
(hypothesis)

Init

Reach^k

Bad
ACTIVE LEARNING OF INVARIANTS

Program
Constraint Solver
check hypothesis?

Teacher

concrete data-points

Learner

$H \ (\text{hypothesis})$

$H \in H \implies H'$

Init $\xrightarrow{\text{Reach}^k} \text{Bad}$

$p \rightarrow p'$

$p \in H \implies p' \in H$
ICE: LEARNING USING EXAMPLES, COUNTER-EXAMPLES AND IMPLICATIONS

- To refute non-inductiveness of $H$, the teacher communicates $(p, p')$
  - if $p \in H$, then $p' \in H$ -- learner’s choice depends on simplicity, etc.

- Robust framework
  - Ensures progress and honest teacher

- Strong convergence
  - Can the learner eventually learn an invariant irrespective of the teacher’s answers?
ICE-LEARNING NUMERICAL INVARIANTS

\[ H = \bigvee_i \bigwedge_j \left( \pm v_1^{ij} \pm v_2^{ij} \leq c^{ij} \right) \]

Learning algorithm is strongly convergent

Implemented as a tool over Boogie (from Microsoft Research)
## ICE-LEARNING NUMERICAL INVARIANTS

<table>
<thead>
<tr>
<th>Program</th>
<th>Invariant</th>
<th>InvGen (sec)</th>
<th>Sharma et al (sec)</th>
<th>ICE (sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>w2</td>
<td>$x \leq n - 1$</td>
<td>0.1</td>
<td>0.1</td>
<td>0.0</td>
</tr>
<tr>
<td>ex7</td>
<td>$0 \leq i \land y \leq \text{len}$</td>
<td>X</td>
<td>0.4</td>
<td>0.0</td>
</tr>
<tr>
<td>array</td>
<td>$j \leq 0 \lor m \leq a[0]$</td>
<td>X</td>
<td>0.2</td>
<td>0.3</td>
</tr>
<tr>
<td>fig1</td>
<td>$x \leq -1 \lor y \geq 1$</td>
<td>X</td>
<td>X</td>
<td>0.1</td>
</tr>
<tr>
<td>fig3</td>
<td>$\text{lock} = 1 \lor x \leq y - 1$</td>
<td>0.1</td>
<td>0.1</td>
<td>0.0</td>
</tr>
<tr>
<td>fig9</td>
<td>$x = 0 \land y \geq 0$</td>
<td>0.1</td>
<td>0.2</td>
<td>0.0</td>
</tr>
<tr>
<td>ex23</td>
<td>$0 \leq y \leq z \land z \leq c + 4572$</td>
<td>X</td>
<td>X</td>
<td>14.2</td>
</tr>
<tr>
<td>cgr2</td>
<td>$(N \leq 0) \lor (0 \leq x \land 0 \leq m \leq N - 1)$</td>
<td>X</td>
<td>X</td>
<td>7.3</td>
</tr>
<tr>
<td>trex3</td>
<td>$0 \leq x1 \land 0 \leq x2 \land 0 \leq x3 \land d1 = 1 \land d2 = 1 \land d3 = 1$</td>
<td>0.5</td>
<td>X</td>
<td>2.2</td>
</tr>
<tr>
<td>tcs</td>
<td>$i \leq j - 1 \lor i \geq j + 1 \lor x = y$</td>
<td>0.1</td>
<td>0.5</td>
<td>1.4</td>
</tr>
<tr>
<td>sum1</td>
<td>$(sn = i - 1) \land (sn = 0 \lor sn \leq n)$</td>
<td>X</td>
<td>X</td>
<td>1.8</td>
</tr>
</tbody>
</table>
LEARNING DATA STRUCTURE INVARIANTS

• Learn invariants over linear data structures (arrays and lists)

• Unbounded size of the heap
  - properties are universally quantified

\[
\forall y_1, y_2, (\text{head} \rightarrow_{\text{next}}^* y_1 \rightarrow_{\text{next}}^* y_2 \Rightarrow \text{data}(y_1) \leq \text{data}(y_2))
\]

• General form: \( \land_i \forall y. (\text{Guard}_i(p, y) \Rightarrow \text{Data}_i(p, y)) \)

• Strongly convergent algorithms for learning quantified invariants.

• Quantified Data Automata (QDA) \( \Rightarrow \) normal form for such invariants
  - adapt passive Regular Positive Negative Inference (RPNI) for learning QDAs
ONGOING WORK

• Applying machine learning algorithms for synthesizing invariants

• Applying machine learning algorithms to program synthesis

• Synthesizing invariants for separation logic.