Nano Steps and Baby Challenges within the Grand Challenge

Deepak Kapur
Department of Computer Science
University of New Mexico
Albuquerque, New Mexico, USA
Automating Induction Theorem Proving

- An extremely difficult problem

- Issues:
  - Induction variable(s)
  - Induction schema to be used
  - Determining intermediate lemmas Needed

- Baby Challenge: Can we characterize conjectures which can be decided **automatically** (without user interaction) using inductive methods?
Structural Conditions on Recursive Definitions

A recursive definition of $f$ is **theory-based** if all terms in the definition are from the theory except for occurrences of $f$.

\[
\begin{align*}
\text{append}(\text{nil}, y) &= y \\
\text{append}(\text{cons}(a, x), y) &= \text{cons}(a, \text{append}(x, y)) \\
0 \ast y &= 0 \\
s(x) \ast y &= (x \ast y) + x
\end{align*}
\]

Compatibility of function definitions:

When functions are composed and their arguments are instantiated in a subgoal of an induction proof attempt, it should be possible to simplify them so that the induction hypothesis is applicable and the simplified subgoal is a formula in a decidable theory.
Decidable Conjectures

\[
\begin{align*}
\exp(\log(x)) &= x \\
\log(\exp(x)) &= x \\
\bton(\pad0(\ntob(x))) &= x \\
\text{last}(\ntob(\double(x))) &= 0 \\
\log(\exp(x)) &= x \\
\bton(\pad0(\ntob(x))) &= x \\
\text{last}(\ntob(\double(x))) &= 0 \\
\double(u + v) &= u + \double(v) \\
\double(u + v) &= \double(u) + \double(v) \\
(u + v) + w &= u + (v + w) \\
\text{len}(\text{append}(u, v)) &= \text{len}(u) + \text{len}(v) \\
\text{min}(u + v, u + w) &= u + \text{min}(v, w) \\
\text{s}(\text{len}(\text{append}(u, v))) &= \text{len}(\text{append}(u, \text{cons}(n, v))) \\
\text{double}(x) &= x \Rightarrow x = 0 \\
\text{double}(\text{half}(x)) &= x \Rightarrow \text{even}(x) = \text{true}
\end{align*}
\]

Details can be found in

What Next?

- Extend classes of recursive definitions and relationship between them.
- Extend classes of conjectures that can be handled automatically.
- Bootstrapping: Extended decision procedures and multilevel induction proof attempts needed lemmas.
Automatic Generation of Polynomial Loop Invariants

1. Quantifier-Elimination: Eliminating Program Variables from Parameterized Formulas Hypothesized as Assertions

2. Ideal-Theoretic Methods:

   Polynomial Invariants Form an Ideal

   a) Intersection of Invariant Ideals Corresponding to All Paths of Execution of a Program
   b) Program Construct Semantics using Ideal Operations
Polynomial Invariants Form an Ideal

- **States** at a program point $\equiv$ set of values variables take
- Characterize states by a conjunction of polynomial equations

$$ (p_1 = 0 \land \cdots \land p_k = 0). $$

The set of values which make the above formula true can be characterized by the radical ideal of $\{p_1, \cdots, p_k\}$, denoted as $IV(p_1, \cdots, p_k)$.

- If $p = 0, q = 0$ are invariants, so are $s \cdot p = 0$ for any polynomial $s$ as well as $p + q = 0$.

**Objective:** Invoking Hilbert's finite basis theorem, a finite basis of the invariant ideal corresponding to program states at a control point exists. How to compute this ideal?
# Table of Examples

<table>
<thead>
<tr>
<th>PROGRAM</th>
<th>COMPUTING</th>
<th>VARIABLES</th>
<th>BRANCHES</th>
<th>TIMING</th>
</tr>
</thead>
<tbody>
<tr>
<td>freire1</td>
<td>√</td>
<td>2</td>
<td>1</td>
<td>&lt; 3 s.</td>
</tr>
<tr>
<td>freire2</td>
<td>√</td>
<td>3</td>
<td>1</td>
<td>&lt; 5 s.</td>
</tr>
<tr>
<td>cohen cu</td>
<td>cube</td>
<td>4</td>
<td>1</td>
<td>&lt; 5 s.</td>
</tr>
<tr>
<td>cousot</td>
<td>toy</td>
<td>2</td>
<td>2</td>
<td>&lt; 4 s.</td>
</tr>
<tr>
<td>divbin</td>
<td>division</td>
<td>3</td>
<td>2</td>
<td>&lt; 5 s.</td>
</tr>
<tr>
<td>dijkstra</td>
<td>√</td>
<td>3</td>
<td>2</td>
<td>&lt; 6 s.</td>
</tr>
<tr>
<td>fermat2</td>
<td>factor</td>
<td>3</td>
<td>2</td>
<td>&lt; 4 s.</td>
</tr>
<tr>
<td>wensley2</td>
<td>division</td>
<td>4</td>
<td>2</td>
<td>&lt; 5 s.</td>
</tr>
<tr>
<td>euclidex2</td>
<td>gcd</td>
<td>6</td>
<td>2</td>
<td>&lt; 6 s.</td>
</tr>
<tr>
<td>lcm2</td>
<td>lcm</td>
<td>4</td>
<td>2</td>
<td>&lt; 5 s.</td>
</tr>
<tr>
<td>factor</td>
<td>factor</td>
<td>4</td>
<td>4</td>
<td>&lt; 20 s.</td>
</tr>
</tbody>
</table>

PC Linux Pentium 4 2.5 Ghz Details can be found in

### Table of Examples

<table>
<thead>
<tr>
<th>PROGRAM</th>
<th>COMPUTING</th>
<th>d</th>
<th>VARS</th>
<th>IF'S</th>
<th>LOOPS</th>
<th>DEPTH</th>
<th>TIME</th>
</tr>
</thead>
<tbody>
<tr>
<td>cohencu</td>
<td>cube</td>
<td>3</td>
<td>5</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>2.45</td>
</tr>
<tr>
<td>dershowitz</td>
<td>real division</td>
<td>2</td>
<td>7</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1.71</td>
</tr>
<tr>
<td>divbin</td>
<td>integer division</td>
<td>2</td>
<td>5</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>1.91</td>
</tr>
<tr>
<td>euclidex1</td>
<td>Bezout's coefs</td>
<td>2</td>
<td>10</td>
<td>0</td>
<td>2</td>
<td>2</td>
<td>7.15</td>
</tr>
<tr>
<td>euclidex2</td>
<td>Bezout's coefs</td>
<td>2</td>
<td>8</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>3.69</td>
</tr>
<tr>
<td>fermat</td>
<td>divisor</td>
<td>2</td>
<td>5</td>
<td>0</td>
<td>3</td>
<td>2</td>
<td>1.55</td>
</tr>
<tr>
<td>prod4br</td>
<td>product</td>
<td>3</td>
<td>6</td>
<td>3</td>
<td>1</td>
<td>1</td>
<td>8.49</td>
</tr>
<tr>
<td>freire1</td>
<td>integer sqrt</td>
<td>2</td>
<td>3</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0.75</td>
</tr>
<tr>
<td>hard</td>
<td>integer division</td>
<td>2</td>
<td>6</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>2.19</td>
</tr>
<tr>
<td>lcm2</td>
<td>lcm</td>
<td>2</td>
<td>6</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>2.03</td>
</tr>
<tr>
<td>readers</td>
<td>simulation</td>
<td>2</td>
<td>6</td>
<td>3</td>
<td>1</td>
<td>1</td>
<td>4.15</td>
</tr>
</tbody>
</table>

**PC Linux Pentium 4 2.5 Ghz**

Details can be found in

Main Result

**THEOREM.** In a loop with assignments $\bar{x} := f_i(\bar{x})$, if tests are ignored and each $f_i$ is a solvable mapping with positive rational eigenvalues, the algorithm computes the strongest invariant in at most $2m + 1$ steps, where $m$ is the number of program variables in the loop.
Role of Algebraic Geometry

Soundness and Completeness of methods are proved using results from algebraic geometry:

- Hilbert's finite basis theorem for polynomial ideals,
- Dimensional analysis of ideals and how iterations of the loop give more and more information that reducing the dimension of ideals approximating the invariant ideal, and
- Finite dimensionality of vector spaces.
\[ a := 0; \quad s := 1; \quad t := 1; \]
\[ \textbf{while} \ (s \leq N) \ \textbf{do} \]
\[
\begin{align*}
a &:= a + 1; \
s &:= s + t + 2; \
t &:= t + 2;
\end{align*}
\] 
\[ \textbf{end while} \]
Quantifier-Elimination Method

\[ a := 0; \quad s := 1; \quad t := 1; \]
\[ \text{while } (s \leq N) \text{ do} \]
\[ \{ I(a, s, t) = (u_1 a^2 + u_2 s^2 + u_3 t^2 + u_4 a s + u_5 a t + u_6 st + u_7 a + u_8 s + u_9 t + u_{10} = 0) \} \]
\[ a := a + 1; \quad s := s + t + 2; \quad t := t + 2; \]
\[ \text{end while} \]
Example: Square Root Program

\[ a := 0; \quad s := 1; \quad t := 1; \]
\[ \text{while} \ (s \leq N) \ \text{do} \]
\[ \{ I(a, s, t) = (u_1 a^2 + u_2 s^2 + u_3 t^2 + u_4 as + u_5 at + u_6 st + u_7 a + u_8 s + u_9 t + u_{10} = 0) \} \]
\[ a := a + 1; \quad s := s + t + 2; \quad t := t + 2; \]
\[ \text{end while} \]

Quantifier elimination on the verification condition gives:

\[ u_1 = -u_5, \quad u_7 = -2u_3 - u_5 + 2u_{10}, \quad u_8 = -4u_3 - u_5, \quad u_9 = 3u_3 + u_5 - u_{10} \]
Example: Square Root Program

\[ a := 0; \quad s := 1; \quad t := 1; \]
\[ \textbf{while} (s \leq N) \textbf{do} \]
\[ \{I(a, s, t) = (u_1 a^2 + u_2 s^2 + u_3 t^2 + u_4 as + u_5 at + u_6 st + u_7 a + u_8 s + u_9 t + u_{10} = 0)\} \]
\[ a := a + 1; \quad s := s + t + 2; \quad t := t + 2; \]
\[ \textbf{end while} \]

Quantifier elimination on the verification condition gives:

\[ u_1 = -u_5, \quad u_7 = -2u_3 - u_5 + 2u_{10}, \quad u_8 = -4u_3 - u_5, \quad u_9 = 3u_3 + u_5 - u_{10} \]

Making exactly one of \( u_5, u_3, u_{10} \) to be 1, and other parameters to be 0, the following independent invariants are generated:

\[ 2a - t + 1 = 0, \quad a^2 - at + a + s - t = 0, \quad t^2 - 2a - 4s + 3t = 0 \]
Quantifier-Elimination Methods

- Generalized Presburger Arithmetic (for invariants expressed using linear inequalities)
- Parametric Gröbner Basis Algorithm (Kapur, 1994)
  - (similar to Weispfenning’s Comprehensive Gröbner Basis Algorithms)
- Quantifier Elimination Techniques for Real Closed Fields (REDLOG, QEPCAD)
What Next?

- **Algebraic Geometry** is a powerful theory about polynomials (built from numbers, variables and operations including $+, \times$) and it is very useful for automatically generating invariants for a small class of programs.

- How can similar theories be developed for other data structures - arrays, records, sequences, lists, objects?