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.

$$
\begin{aligned}
append(nil, y) &= y \\
append(cons(a, x), y) &= cons(a, append(x, y))
\end{aligned}
$$

$$
\begin{aligned}
0 * y &= 0 \\
s(x) * y &= (x * y) + x
\end{aligned}
$$

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

$$
\begin{array}{rlcl}
 & exp2(log(x)) & = & x \\
 & log(exp2(x)) & = & x \\
 & bton(pad0(ntob(x))) & = & x \\
 & last(ntob(double(x))) & = & 0 \\
 & log(exp2(x)) & = & x \\
 & bton(pad0(ntob(x))) & = & x \\
 & last(ntob(double(x))) & = & 0 \\
 & double(u+v) & = & u + double(v) \\
 & double(u+v) & = & double(u) + double(v) \\
 & (u+v)+w & = & u+(v+w) \\
 & len(append(u,v)) & = & len(u) + \mathsf{len}(v) \\
 & min(u+v, u+w) & = & u + min(v,w) \\
 & s(len(append(u,v))) & = & len(append(u, cons(n,v))) \\
 & u*(v+w) & = & u*v+u*w \\
double(x) = x & \Rightarrow x & = & 0 \\
double(half(x)) = x & \Rightarrow even(x) & = & true
\end{array}
$$

Details can be found in

Kapur and Subramaniam (CADE-2000), Kapur and Giesl (IJCAR-2001) and Kapur and Giesl (CADE-2003)

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

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

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 $\mathbf{IV}(p_1, \cdots, p_k)$.

- If $p = 0, q = 0$ are invariants, so are $s\ p = 0$ 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

PROGRAM	COMPUTING	VARIABLES	BRANCHES	TIMING
freire1	$\sqrt[2]{}$	2	1	$<$ 3 s.
freire2	$\sqrt[3]{}$	3	1	$<$ 5 s.
cohencu	cube	4	1	$<$ 5 s.
cousot	toy	2	2	$<$ 4 s.
divbin	division	3	2	$<$ 5 s.
dijkstra	$\sqrt[2]{}$	3	2	$<$ 6 s.
fermat2	factor	3	2	$<$ 4 s.
wensley2	division	4	2	$<$ 5 s.
euclidex2	gcd	6	2	$<$ 6 s.
lcm2	lcm	4	2	$<$ 5 s.
factor	factor	4	4	$<$ 20 s.

PC Linux Pentium 4 2.5 Ghz Details can be found in

E. Rodríguez-Carbonell and D. Kapur, "Automatic Generation of Polynomial Loop Invariants: Algebraic Foundations," Proc. *International Symposium on Symbolic and Algebraic Computation (ISSAC-2004)*, July 2004, Santander Spain

Table of Examples

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

PC Linux Pentium 4 2.5 Ghz

Details can be found in

E. Rodríguez-Carbonell and D. Kapur, "An Abstract Interpretation Approach for Automatic Generation of Polynomial Invariants," Proc. *11th Static Analysis Symposium (SAS-2004)*, September 2004, Verona, Italy.

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;   s := 1;   t := 1;
while (s ≤ N) do

    a := a + 1;      s := s + t + 2;      t := t + 2;
end while
```

Quantifier-Elimination Method

$a := 0; \quad s := 1; \quad t := 1;$

while $(s \le N)$ **do**

$\{I(a, s, t) = (u_1 \; a^2 + u_2 \; s^2 + u_3 \; t^2 + u_4 \; as + u_5 \; at + u_6 \; st + u_7 \; a + u_8 \; s + u_9 \; t + u_{10} = 0)\}$

$\quad a := a + 1; \qquad s := s + t + 2; \qquad t := t + 2;$

end while

Example: Square Root Program

$a := 0; \quad s := 1; \quad t := 1;$

while $(s \leq N)$ **do**

$\{I(a, s, t) = (u_1\ a^2 + u_2\ s^2 + u_3\ t^2 + u_4\ as + u_5\ at + u_6\ st + u_7\ a + u_8\ s + u_9\ t + u_{10} = 0)\}$

$\quad a := a + 1; \qquad s := s + t + 2; \qquad t := t + 2;$

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

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