Polynomial Invariant Generation for Non-deterministic Recursive Program

Krishnendu Chatterjee et al.

January 6, 2021

Problem

Invariant generation:

- Program: programs with **polynomial assignments**.
- Invariant: conjunction of strict **polynomial inequalities**.

Find a sound and complete algorithm for the synthesis.

Contributions

- Non-recursive: A sound and complete method to generate polynomial invariants for polynomial programs (template-based).
- The method can be extended to handle recursion.
- Worst-case complexity: subexponential.
- Weak invariant: can be reduced to solving a quadratically-constrained linear program (QCLP).

Contribution: Comparison

Approach	Assignments and Guards	Invariants	Nondet	Rec	Prob	Sound	Complete	Weak	Strong
This Work	Polynomial	Polynomial	✓	✓	✗	✓	✓[♦]	✓ QCLP	✓ Subexp
[19] CAV'03	Linear[c]	Linear	✓	✗	✗	✓	✓	✓ Exp[†]	✓ Exp[†]
[50] ACA'04	Polynomial	Polynomial	✓	✓	✗	✓	✓	✓ 2Exp	✓ 2Exp
[29] OOPSLA'13	General	Linear (Presburger)	✓	✓	✗	✓	✗	✗	✗
[32] ATVA'17	Polynomial	Polynomial	✗	✗	✓	✓	✓[a]	✓ Poly	✗
[47] LICS'18	Linear[‡]	Polynomial *Equalities*	✓	✗	✗	✓[‡]	✓[‡]	✗	✓[‡,b]
[53] POPL'18	Polynomial, Exponential, Logarithmic	Polynomial, Exponential, Logarithmic	✓	✓	✗	✓	✗	✗	✗
[66]* ISSAC'04	Polynomial, Exponential	Polynomial *Equalities*	✓	✗	✗	✓	✓	✓[b]	✓[b]
[69] POPL'04	Polynomial[c]	Polynomial *Equalities*	✓	✗	✗	✓	✓[b]	✓[b]	✓[b]
[31] FMCAD'15	General[d]	General[d]	✓	✗[e]	✗	✓	✗	✗	✗
[52] PLDI'17	General	General	✓	✓	✗	✓	✗	✗	✗
[28] ATVA'16	Polynomial, *Without Conditional Branching*	Polynomial *Equalities*	✓	✗	✗	✓	✓	✓ Poly	✓ Poly
[49]* ISSAC'17	Polynomial[‡]	Polynomial *Equalities*	✓	✗	✗	✓	✓[‡]	✓[‡,b]	✓[‡,b]
[1]* SAS'15	Polynomial	Polynomial	✗	✗	✗	✓	✗	✗	✗

Overview

- A demo example.
- Algorithm.
  1. Computation model.
  2. Preliminaries: program, invariant and mathmatical tools.
  3. Algorithm for non-recursive programs.
  4. Soundness and semi-completeness.
  5. Algorithm for recursive program.
- Experimental Results.

Example: Basic Idea

Goal: synthesize a postcondition and invariants.

Precondition: $100 - y^2 \geq 0$
if $x^2 - 100 \geq 0$ **then**
  Invariant: $c_1 \cdot y^2 + c_2 \cdot y + c_3 \geq 0$
  x := y
else
  Invariant: $c_4 \cdot x^2 + c_5 \cdot x + c_6 \geq 0$
  skip
fi
Postcondition: $c_7 \cdot x + c_8 \geq 0$

$$100 - y^2 \geq 0 \land x^2 - 100 \geq 0 \Rightarrow c_1 y^2 + c_2 y + c_3 \geq 0$$

$$100 - y^2 \geq 0 \land x^2 - 100 \geq 0 \Rightarrow c_4 x^2 + c_5 x + c_6 < 0$$

$$c_1 y^2 + c_2 y + c_3 \geq 0 \Rightarrow c_7 y + c_8 \geq 0$$

$$c_4 x^2 + c_5 x + c_6 \geq 0 \Rightarrow c_7 x + c_8 \geq 0$$

Example: Basic Idea

Precondition: $100 - y^2 \geq 0$
if $x^2 - 100 \geq 0$ **then**
    Invariant: $c_1 \cdot y^2 + c_2 \cdot y + c_3 \geq 0$
    $x := y$
else
    Invariant: $c_4 \cdot x^2 + c_5 \cdot x + c_6 \geq 0$
    skip
fi
Postcondition: $c_7 \cdot x + c_8 \geq 0$

$$100 - y^2 \geq 0 \wedge x^2 - 100 \geq 0 \Rightarrow c_1 y^2 + c_2 y + c_3 \geq 0$$
$$100 - y^2 \geq 0 \wedge x^2 - 100 \geq 0 \Rightarrow c_4 x^2 + c_5 x + c_6 < 0$$

Directly: $c_1 = -1, c_2 = 0, c_3 = 100,\ c_4 = 1, c_5 = 0, c_6 = 100.$

$$c_1 y^2 + c_2 y + c_3 \geq 0 \Rightarrow c_7 y + c_8 \geq 0$$

Example: Basic Idea

$$c_1 y^2 + c_2 y + c_3 \geq 0 \Rightarrow c_7 y + c_8 \geq 0$$

1. wlog. We can add tautology to assumptions.

$$c_1 y^2 + c_2 y + c_3 \geq 0 \wedge (ay - b)^2 \geq 0 \Rightarrow c_7 y + c_8 \geq 0$$

2. Expanded:

$$c_7 y + c_8 = (ay - b)^2 + d(c_1 y^2 + c_2 y + c_3)$$

3. Parameters are equal:

$$0 = a^2 + c_1 d$$
$$c_7 = -2ab + c_2 d$$
$$c_8 = b^2 + c_3 d$$
.

With previous valuation for c_1, c_2, c_3 and put into we have $c_7 = -1, c_8 = 10,$ $a = \frac{1}{2\sqrt{5}}, b = \sqrt{5}, d = \frac{1}{20}$

Example: General

- **Soundness**: To prove

  $$g_1 \geq 0, g_2 \geq 0, \ldots, g_m \geq 0 \Rightarrow g \geq 0$$

  where $g, g_i's$ are polynomials.

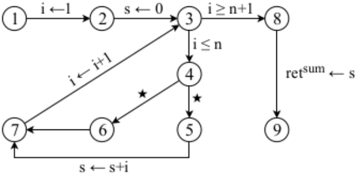
  $$g = h_0 + \sum_{i=1}^{m} h_i \cdot g_i$$

  where h_i is sum of squares.
- **Semi-completeness**: In real algebraic geometry: Putinar's Positivstellensatz Theorem.

Preliminaries: Program

Example

```
      sum(n)  {
1a:   i  :=  1;
2a:   s  :=  0;
3b:   while  i ≤ n  do
4d:          if  ⋆  then
5a:                 s  :=  s + i
             else
6a:                 skip
             fi;
7a:          i  :=  i + 1
      od;
8a:   return  s
9e:   }
```

- Sets of labels: $\mathbf{L}_a, \mathbf{L}_b, \mathbf{L}_c, \mathbf{L}_d, \mathbf{L}_e$
- Set of variables:
  $V_*^f = \{\mathtt{ret}^f, n, \bar{n}\}$
  $V^f = V_*^f \cup \{s, i\}$
- 

Preliminaries: Invariants

In the following $\mathfrak{e}$s' are all arithmetic expressions.

- Pre-conditions: $\mathtt{Pre}(l) = \bigwedge_{i=1}^{m} \mathfrak{e}_i \geq 0$ over V^f.
- Post-conditions: $\mathtt{Post}(f) = \bigwedge_{i=1}^{m} \mathfrak{e}_i > 0$ over input variables and return variable.
- Invariants: $\mathtt{Inv}(l) = \bigwedge_{i=1}^{m} \mathfrak{e}_i > 0$ over V^f.
- Inductive invariants: Initialization($\mathtt{Pre}(l_{in}) \Rightarrow \mathtt{Ind}(l_{in})$) and Consecution.
- Abstract Path: use pre- and post- conditions as a behavior of a function.
- Recursive inductive invariants: initiation, consecution and post-condition consecution.

Preliminaries: Theorems

Theorem (Putinar's Positivstellensatz)

$g_1, \ldots, g_m, g \in \mathbb{R}[V]$ are polynomials over V with real coefficients. Let $\Pi := \{x \in \mathbb{R}^V \mid \forall i. g_i(x) \geq 0\}$. If

- *$\{x \in \mathbb{R}^V \mid g_k(x) \geq 0\}$ is compact.*
- *$g(x) > 0$ for all $x \in \Pi$.*

then,

$$g = h_0 + \sum_{i=1}^{m} h_i \cdot g_i$$

Lemma

Given a $h \in \mathbb{R}[V]$, deciding whether $h = \Sigma_i f_i^2$ can be reduced in polynomial time to solving a system of quadratic equalities.

Algorithm: Non-recursive

Input: Program P, polynomial precondition `Pre`, max polynomial degree d, max size for invariant n and a parameter γ

Step 1: Setting up templates.

$$\eta(\ell) := s_{\ell,1,1} + s_{\ell,1,2} \cdot n + s_{\ell,1,3} \cdot \bar{n} + s_{\ell,1,4} \cdot i + s_{\ell,1,5} \cdot s + s_{\ell,1,6} \cdot r + \\ s_{\ell,1,7} \cdot n^2 + s_{\ell,1,8} \cdot n \cdot \bar{n} + s_{\ell,1,9} \cdot n \cdot i + s_{\ell,1,10} \cdot n \cdot s + s_{\ell,1,11} \cdot n \cdot r + \\ s_{\ell,1,12} \cdot \bar{n}^2 + s_{\ell,1,13} \cdot \bar{n} \cdot i + s_{\ell,1,14} \cdot \bar{n} \cdot s + s_{\ell,1,15} \cdot \bar{n} \cdot r + s_{\ell,1,16} \cdot i^2 + \\ s_{\ell,1,17} \cdot i \cdot s + s_{\ell,1,18} \cdot i \cdot r + s_{\ell,1,19} \cdot s^2 + s_{\ell,1,20} \cdot s \cdot r + s_{\ell,1,21} \cdot r^2 > 0.$$

Step 2: Setting up constraint pairs for each transition in CFG. (Γ, g). where $\Gamma = \bigwedge_{i=1}^{m} g_i \geq 0$.

Step 3: Translating constraint pairs to quadratic equalities. For each pair above, compute the equation of the form

$$g = \epsilon + h_0 + \textstyle\sum_{i=1}^{m} h_i \cdot g_i$$

where $h_i = \Sigma_{j=1}^{r} h_i \cdot m'_j$, and m'_j is monomial of degree at most λ.

Step 4: Finding representative solutions for quadratic equalities from above coefficients equal.

Algorithm: Complexity

- During the algorithm above, the system's size is polynomially dependent on number of lines of the program (for each transition in CFG).
- Solving the quadratic inequalities system is in subexponential time.

Theorem (Strong Invariant Synthesis)

Given a non-recursive program P and a precondition that satisfy the compactness condition, above algorithm solves the strong invariant synthesis problem in subexponential time. The solution is sound and semi-complete.

Recursive Program

Differences:

- **Step 1**: add setting up templates for post-conditions of functions.
- **Step 2**:
  setting up constraints pairs for function-call statements and post-condition consecution.

Implementation and Experimental Results

Implemented in Java. QCLP Solver: LOQO.
Parameter γ: highest degree of the input program.
Non-recursive:

Benchmark	n	d	|V|	|S|	Ours	ICRA	SeaHorn	[49]	UAutomizer	[50] using Z3
cohendiv	3	2	6	17391	15.2	0.7	0.1	Not Applicable	3.3	Timed Out
divbin	3	2	5	18351	5.4	Failed	Timed Out	0.2	Failed	Timed Out
hard	3	2	6	24975	28.0	Failed	Failed	0.4	Failed	Timed Out
mannadiv	3	2	5	16245	18.2	Failed	0.1	0.1	Timed Out	Timed Out
wensely	2	2	7	18874	20.1	Failed	Failed	0.1	Failed	Timed Out
sqrt	2	2	4	4072	5.8	0.8	Failed	0.1	Timed Out	Timed Out
dijkstra	2	2	5	10156	12.8	Failed	Failed	Not Applicable	Failed	Timed Out
z3sqrt	2	2	6	9404	12.9	0.5	0.1	Not Applicable	Failed	Timed Out
freire1	2	2	3	2432	26.5	0.6	Failed	0.1	Failed	Timed Out
freire2	2	3	4	9708	10.7	1.1	Failed	0.1	Failed	Timed Out
euclidex1	2	2	11	45756	97.5	Failed	Failed	Not Applicable	Timed Out	Timed Out
euclidex2	2	2	8	22468	39.3	Failed	Failed	0.4	Timed Out	Timed Out
euclidex3	2	2	13	72762	203.1	Failed	Failed	Not Applicable	Timed Out	Timed Out
lcm1	2	2	6	13361	17.9	0.8	0.1	Not Applicable	3.7	Timed Out
lcm2	2	2	6	12517	18.7	0.8	0.1	0.1	3.2	Timed Out
prodbin	2	2	5	10096	12.1	Failed	Failed	Not Applicable	Timed Out	Timed Out
prod4br	2	2	6	21064	43.2	Failed	Failed	Not Applicable	Timed Out	Timed Out
cohencu	2	3	5	16664	11.8	0.6	Failed	0.1	Timed Out	Timed Out
petter	1	2	3	1080	20.4	0.5	0.1	0.1	2.7	Timed Out

Recursive: 8 cases, only 1 can be solved by ICRA and SeaHorn.

Conclusion

- A sound and semi-complete synthesis of polynomial invariants.
- Progress. Weakness.
- Future work.