Polynomial Invariant Generation
for Non-deterministic Recursive Programs

Krishnendu Chatterjee¹, Hongfei Fu²
Amir Kafshdar Goharshady¹, Ehsan Kafshdar Goharshady³

¹IST Austria
²Shanghai Jiao Tong University
³Ferdowsi University of Mashhad
Invariants

- An assertion at a point of the program that holds whenever a *valid* execution reaches that point
- An over-approximation of the set of reachable states

```
Precondition: 100 - y^2 \geq 0
if x^2 - 100 \geq 0 then
   Invariant: -y^2 + 100 \geq 0
   x := y
else
   Invariant: -x^2 + 100 \geq 0
   skip
fi
Postcondition: 10 - x \geq 0
```
Inductive Invariants

Let $C$ be a set of program locations that is visited by every cycle. An *Inductive Invariant* is an assertion $A_l$ at every location $l \in C$, such that for each $l, l' \in C$:

- **Initiation**: $A_l$ holds in the first visit to $l$.
- **Consecution**: If $A_l$ holds at $l$, then every simple path from $l$ to $l'$ ensures that $A_{l'}$ holds at $l'$.

In the sequel, we assume $C = L$, i.e. every label is in $C$.

The primary method to show that an assertion is an invariant is to generate an inductive invariant that strengthens it.
Invariant Generation

- Safety
- Termination
- Complexity Bounds
- Cost Bounds
### Previous Works on Invariant Generation

<table>
<thead>
<tr>
<th>Approach</th>
<th>Assignments and Guards</th>
<th>Invariants</th>
<th>Nondet</th>
<th>Rec</th>
<th>Prob</th>
<th>Sound</th>
<th>Complete</th>
<th>Weak</th>
<th>Strong</th>
</tr>
</thead>
<tbody>
<tr>
<td>This Work</td>
<td>Polynomial</td>
<td>Polynomial</td>
<td>✓</td>
<td>✓</td>
<td>×</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>CAV'03</td>
<td>Linear&lt;sup&gt;c&lt;/sup&gt;</td>
<td>Linear</td>
<td>✓</td>
<td>×</td>
<td>×</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>ACA'04</td>
<td>Polynomial</td>
<td>Polynomial</td>
<td>✓</td>
<td>✓</td>
<td>×</td>
<td>✓</td>
<td>✓</td>
<td>✓‡</td>
<td>✓</td>
</tr>
<tr>
<td>OOPSLA'13</td>
<td>General</td>
<td>Linear (Presburger)</td>
<td>✓</td>
<td>✓</td>
<td>×</td>
<td>✓</td>
<td>×</td>
<td>×</td>
<td>×</td>
</tr>
<tr>
<td>ATVA'17</td>
<td>Polynomial</td>
<td>Polynomial</td>
<td>×</td>
<td>×</td>
<td>✓</td>
<td>✓</td>
<td>✓&lt;sup&gt;a&lt;/sup&gt;</td>
<td>✓</td>
<td>Poly</td>
</tr>
<tr>
<td>LICS'18</td>
<td>Linear&lt;sup&gt;d&lt;/sup&gt;</td>
<td>Polynomial</td>
<td>✓</td>
<td>×</td>
<td>×</td>
<td>✓</td>
<td>✓‡</td>
<td>✓‡</td>
<td>✓‡&lt;sup&gt;b&lt;/sup&gt;</td>
</tr>
<tr>
<td>POPL'18</td>
<td>Polynomial, Exponential, Logarithmic</td>
<td>Polynomial, Exponential, Logarithmic</td>
<td>✓</td>
<td>✓</td>
<td>×</td>
<td>✓</td>
<td>×</td>
<td>×</td>
<td>×</td>
</tr>
<tr>
<td>ISSAC'04</td>
<td>Polynomial, Exponential</td>
<td>Polynomial</td>
<td>✓</td>
<td>×</td>
<td>×</td>
<td>✓</td>
<td>✓</td>
<td>✓&lt;sup&gt;b&lt;/sup&gt;</td>
<td>✓&lt;sup&gt;b&lt;/sup&gt;</td>
</tr>
<tr>
<td>POPL'04</td>
<td>Polynomial&lt;sup&gt;d&lt;/sup&gt;</td>
<td>Polynomial</td>
<td>✓</td>
<td>×</td>
<td>×</td>
<td>✓</td>
<td>✓‡</td>
<td>✓‡</td>
<td>✓‡&lt;sup&gt;b&lt;/sup&gt;</td>
</tr>
<tr>
<td>FMCAD'15</td>
<td>General&lt;sup&gt;d&lt;/sup&gt;</td>
<td>General&lt;sup&gt;d&lt;/sup&gt;</td>
<td>✓</td>
<td>×&lt;sup&gt;e&lt;/sup&gt;</td>
<td>×</td>
<td>✔</td>
<td>×</td>
<td>□</td>
<td>□</td>
</tr>
<tr>
<td>PLDI'17</td>
<td>General</td>
<td>General</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>ATVA'16</td>
<td>Polynomial, Without Conditional Branching</td>
<td>Polynomial</td>
<td>✓</td>
<td>×</td>
<td>×</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>Poly</td>
</tr>
<tr>
<td>ISSAC'17</td>
<td>Polynomial&lt;sup&gt;f&lt;/sup&gt;</td>
<td>Polynomial</td>
<td>✓</td>
<td>×</td>
<td>×</td>
<td>✓</td>
<td>✓‡</td>
<td>✓‡</td>
<td>✓‡&lt;sup&gt;b&lt;/sup&gt;</td>
</tr>
<tr>
<td>SAS'15</td>
<td>Polynomial</td>
<td>Polynomial</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>✓</td>
<td>✓</td>
<td>×</td>
<td>×</td>
</tr>
</tbody>
</table>

<sup>a</sup> Semi-complete
<sup>b</sup> Uses Gröbner basis computations (super-exponential in worst-case).
<sup>c</sup> Considers general transition systems instead of programs.
<sup>d</sup> Handles non-linearity using linearization heuristics.
<sup>e</sup> Can be extended to handle recursion (see [50]).
<sup>f</sup> Generates a system of quadratic inequalities, but then applies quantifier elimination, leading to exponential runtime.
<sup>†</sup> Treats branching conditions as non-determinism.
* Does not support nested loops.

<sup>‡</sup> Semi-complete, assuming compactness (see Remark 4 and Lemma 5.8)
Polynomial Invariants

- Consider programs with polynomial guards/assignments
- Goal: Generate invariants that are conjunctions of polynomial inequalities

Pick Two:

- Automation
- Completeness
- Applicability
Polynomial Invariants

Pick Two:

- Automation
- Completeness
- Applicability

“Automatically Generating Loop Invariants using Quantifier Elimination”
(Kapur, ACA 2004)
Polynomial Invariants

Pick Two:

- Automation
- Completeness
- Applicability

“Non-linear Reasoning for Invariant Synthesis”
(Kincaid, Cyphert, Breck, and Reps, POPL 2018)
Polynomial Invariants

Pick Two:

- Automation
- Completeness
- Applicability

Interactive Theorem Provers
Not the subject of this talk!
Why not all three?
A Simple Example

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: $c_7 \cdot x + c_8 \geq 0$

\[
\begin{align*}
100 - y^2 & \geq 0 \land x^2 - 100 & \geq 0 & \Rightarrow c_1 \cdot y^2 + c_2 \cdot y + c_3 \geq 0 \\
100 - y^2 & \geq 0 \land 100 - x^2 & > 0 & \Rightarrow c_4 \cdot x^2 + c_5 \cdot x + c_6 \geq 0 \\
c_1 \cdot y^2 + c_2 \cdot y + c_3 & \geq 0 & \Rightarrow c_7 \cdot y + c_8 & \geq 0 \\
c_4 \cdot x^2 + c_5 \cdot x + c_6 & \geq 0 & \Rightarrow c_7 \cdot x + c_8 & \geq 0
\end{align*}
\]
A Simple Example

Precondition: $100 - y^2 \geq 0$

\[ \text{if } x^2 - 100 \geq 0 \text{ then} \]
   \[ \text{Invariant: } -1 \cdot y^2 + 0 \cdot y + 100 \geq 0 \]
   \[ x := y \]

\[ \text{else} \]
   \[ \text{Invariant: } c_4 \cdot x^2 + c_5 \cdot x + c_6 \geq 0 \]
   \[ \text{skip} \]

\[ \text{fi} \]

Postcondition: $c_7 \cdot x + c_8 \geq 0$

\[ \blacktriangleleft 100 - y^2 \geq 0 \land x^2 - 100 \geq 0 \Rightarrow -1 \cdot y^2 + 0 \cdot y + 100 \geq 0 \]

\[ \blacktriangleleft 100 - y^2 \geq 0 \land 100 - x^2 > 0 \Rightarrow c_4 \cdot x^2 + c_5 \cdot x + c_6 \geq 0 \]

\[ \blacktriangleleft -1 \cdot y^2 + 0 \cdot y + 100 \geq 0 \Rightarrow c_7 \cdot y + c_8 \geq 0 \]

\[ \blacktriangleleft c_4 \cdot x^2 + c_5 \cdot x + c_6 \geq 0 \Rightarrow c_7 \cdot x + c_8 \geq 0 \]
A Simple Example

Precondition: $100 - y^2 \geq 0$

if $x^2 - 100 \geq 0$ then

Invariant: $-1 \cdot y^2 + 0 \cdot y + 100 \geq 0$

$x := y$

else

Invariant: $-1 \cdot x^2 + 0 \cdot x + 100 \geq 0$

skip

fi

Postcondition: $c_7 \cdot x + c_8 \geq 0$

$\triangleright 100 - y^2 \geq 0 \land x^2 - 100 \geq 0 \Rightarrow -1 \cdot y^2 + 0 \cdot y + 100 \geq 0$

$\triangleright 100 - y^2 \geq 0 \land 100 - x^2 > 0 \Rightarrow -1 \cdot x^2 + 0 \cdot x + 100 \geq 0$

$\triangleright -1 \cdot y^2 + 0 \cdot y + 100 \geq 0 \Rightarrow c_7 \cdot y + c_8 \geq 0$

$\triangleright -1 \cdot x^2 + 0 \cdot x + 100 \geq 0 \Rightarrow c_7 \cdot x + c_8 \geq 0$
A Simple Example

► $100 - y^2 \geq 0 \Rightarrow c_7 \cdot y + c_8 \geq 0$
► $(a \cdot y - b)^2 \geq 0 \land 100 - y^2 \geq 0 \Rightarrow c_7 \cdot y + c_8 \geq 0$
► $c_7 \cdot y + c_8 = (a \cdot y - b)^2 + d \cdot (100 - y^2)$
► $c_7 \cdot y + c_8 = a^2 \cdot y^2 - 2 \cdot a \cdot b \cdot y + b^2 + 100 \cdot d - d \cdot y^2$

► 0 = $a^2 - d$
► $c_7 = -2 \cdot a \cdot b$
► $c_8 = b^2 + 100 \cdot d$
► One solution: $a = \frac{1}{2\sqrt{5}}, b = \sqrt{5}, d = \frac{1}{20}, c_7 = -1, c_8 = 10$

► In other words

$$10 - y = \left(\frac{1}{2\sqrt{5}} \cdot y - \sqrt{5}\right)^2 + \frac{1}{20}(100 - y^2)$$

So we can safely deduce:

$$100 - y^2 \geq 0 \Rightarrow 10 - y \geq 0$$
A Simple Example

Precondition: $100 - y^2 \geq 0$

if $x^2 - 100 \geq 0$ then

Invariant: $-1 \cdot y^2 + 0 \cdot y + 100 \geq 0$

$x := y$

else

Invariant: $-1 \cdot x^2 + 0 \cdot x + 100 \geq 0$

skip

fi

Postcondition: $-1 \cdot x + 10 \geq 0$

$100 - y^2 \geq 0 \land x^2 - 100 \geq 0 \Rightarrow -1 \cdot y^2 + 0 \cdot y + 100 \geq 0$

$100 - y^2 \geq 0 \land 100 - x^2 > 0 \Rightarrow -1 \cdot x^2 + 0 \cdot x + 100 \geq 0$

$-1 \cdot y^2 + 0 \cdot y + 100 \geq 0 \Rightarrow -1 \cdot y + 10 \geq 0$

$-1 \cdot x^2 + 0 \cdot x + 100 \geq 0 \Rightarrow -1 \cdot x + 10 \geq 0$
Outline

1. Generate a Template
   - For example, \( A_1 := c_0 + c_1 \cdot x + c_2 \cdot y + c_3 \cdot x^2 + c_4 \cdot x \cdot y + c_5 \cdot y^2 \geq 0 \)

2. Compute Inductivity Conditions (Initiation and Consecution)
   - Each such condition is of this form:
     \[
g_1 \geq 0 \land g_2 \geq 0 \land \ldots \land g_m \geq 0 \Rightarrow g \geq 0
\]

3. Handle the condition by writing \( g \) as a combination of \( g_i \)'s:
   \[
g = h_0 + \sum_{i=1}^{m} h_i \cdot g_i
\]  
   where the \( h_i \)'s are polynomials whose coefficients are new unknowns.
   - For example, \( h_0 = a_0 + a_1 \cdot x + a_2 \cdot y + a_3 \cdot x^2 + a_4 \cdot x \cdot y + a_5 \cdot y^2 \)

4. Equate corresponding coefficients on the two sides of (1).
5. Add extra conditions on the coefficients of \( h_i \)'s, ensuring that they are SOS.
6. Solve the resulting quadratic system.
Soundness

Soundness is trivial.

Assuming every $g_i$ is non-negative, and given that every $h_i$ is a SOS, we can directly infer that

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

is also non-negative.
Completeness

Theorem (Putinar’s Positivstellensatz)

Let $V$ be a finite set of variables and $g, g_1, \ldots, g_m \in \mathbb{R}[V]$ polynomials over $V$ with real coefficients. We define $\Pi := \{x \in \mathbb{R}^V \mid \forall i \ g_i(x) \geq 0\}$ as the set of points in which every $g_i$ is non-negative. If (i) there exists some $g_k$ s.t. the set $\{x \in \mathbb{R}^V \mid g_k(x) \geq 0\}$ is compact, and (ii) $g(x) > 0$ for all $x \in \Pi$, then

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

where each polynomial $h_i$ is SOS.

Theorem (See details in the paper)

Under certain conditions, our approach is complete for invariants that are a conjunction of strict polynomial inequalities.
Complexity and Applicability

If we fix the degree of all polynomials and the length of invariant at every point of the program, then our approach is a polynomial-time reduction from Invariant Generation to QP.
### Experimental Results

Times are reported in seconds. Time limit was 12 hours per instance.

<table>
<thead>
<tr>
<th>Benchmark</th>
<th>Ours</th>
<th>ICRA</th>
<th>SeaHorn</th>
<th>Humenberger et al, ISSNAC 2017</th>
<th>UAutomizer</th>
<th>Kapur, ACA 2004</th>
</tr>
</thead>
<tbody>
<tr>
<td>cohendiv</td>
<td>15.2</td>
<td>0.7</td>
<td>0.1</td>
<td>Not Applicable</td>
<td>3.3</td>
<td>Timed Out</td>
</tr>
<tr>
<td>divbin</td>
<td>5.4</td>
<td>Failed</td>
<td>Timed Out</td>
<td>0.2</td>
<td>Failed</td>
<td>Timed Out</td>
</tr>
<tr>
<td>hard</td>
<td>28.0</td>
<td>Failed</td>
<td>Failed</td>
<td>0.4</td>
<td>Failed</td>
<td>Timed Out</td>
</tr>
<tr>
<td>mannadiv</td>
<td>18.2</td>
<td>Failed</td>
<td>0.1</td>
<td>0.1</td>
<td>Timed Out</td>
<td>Timed Out</td>
</tr>
<tr>
<td>wensely</td>
<td>20.1</td>
<td>Failed</td>
<td>Failed</td>
<td>0.1</td>
<td>Failed</td>
<td>Timed Out</td>
</tr>
<tr>
<td>sqrt</td>
<td>5.8</td>
<td>0.8</td>
<td>Failed</td>
<td>0.1</td>
<td>Timed Out</td>
<td>Timed Out</td>
</tr>
<tr>
<td>dijkstra</td>
<td>12.8</td>
<td>Failed</td>
<td>Failed</td>
<td>Not Applicable</td>
<td>Failed</td>
<td>Timed Out</td>
</tr>
<tr>
<td>z3sqrt</td>
<td>12.9</td>
<td>0.5</td>
<td>0.1</td>
<td>Not Applicable</td>
<td>Failed</td>
<td>Timed Out</td>
</tr>
<tr>
<td>freire1</td>
<td>26.5</td>
<td>0.6</td>
<td>Failed</td>
<td>0.1</td>
<td>Failed</td>
<td>Timed Out</td>
</tr>
<tr>
<td>freire2</td>
<td>10.7</td>
<td>1.1</td>
<td>Failed</td>
<td>0.1</td>
<td>Failed</td>
<td>Timed Out</td>
</tr>
<tr>
<td>euclidex1</td>
<td>97.5</td>
<td>Failed</td>
<td>Failed</td>
<td>Not Applicable</td>
<td>Timed Out</td>
<td>Timed Out</td>
</tr>
<tr>
<td>euclidex2</td>
<td>39.3</td>
<td>Failed</td>
<td>Failed</td>
<td>0.4</td>
<td>Timed Out</td>
<td>Timed Out</td>
</tr>
<tr>
<td>euclidex3</td>
<td>203.1</td>
<td>Failed</td>
<td>Failed</td>
<td>Not Applicable</td>
<td>Timed Out</td>
<td>Timed Out</td>
</tr>
<tr>
<td>lcm1</td>
<td>17.9</td>
<td>0.8</td>
<td>0.1</td>
<td>Not Applicable</td>
<td>3.7</td>
<td>Timed Out</td>
</tr>
<tr>
<td>lcm2</td>
<td>18.7</td>
<td>0.8</td>
<td>0.1</td>
<td>0.1</td>
<td>3.2</td>
<td>Timed Out</td>
</tr>
<tr>
<td>prodbin</td>
<td>12.1</td>
<td>Failed</td>
<td>Failed</td>
<td>Not Applicable</td>
<td>Timed Out</td>
<td>Timed Out</td>
</tr>
<tr>
<td>prod4br</td>
<td>43.2</td>
<td>Failed</td>
<td>Failed</td>
<td>Not Applicable</td>
<td>Timed Out</td>
<td>Timed Out</td>
</tr>
<tr>
<td>cohencu</td>
<td>11.8</td>
<td>0.6</td>
<td>Failed</td>
<td>0.1</td>
<td>Timed Out</td>
<td>Timed Out</td>
</tr>
<tr>
<td>petter</td>
<td>20.4</td>
<td>0.5</td>
<td>0.1</td>
<td>0.1</td>
<td>2.7</td>
<td>Timed Out</td>
</tr>
</tbody>
</table>
Summary

For Polynomial Invariant Generation, we can have all three:

- **Automation**: Push-button approach
- **Completeness**: Through Putinar’s Positivstellensatz
- **Applicability**: Reduction to QP

See the paper for proofs and extension to recursive programs!

The research was partially supported by Austrian Science Fund (FWF) Grant No. NFN S11407-N23 (RiSE/SHiNE), Vienna Science and Technology Fund (WWTF) Project ICT15-003, National Natural Science foundation of China (NSFC) Grant No. 61802254, the Facebook PhD Fellowship Program, and DOC Fellowship No. 24956 of the Austrian Academy of Sciences (ÖAW).