Inferring polynomial invariants
with Polyinvar

Helmut Seidl and Michael Petter

TU-München

Chair Workshop, 2005
**Problem:**

**Question**
Is \(x = y\) valid at program point 3?

**Question**
What relation holds at program point 7?

\[\Rightarrow Polynomial\ invariants\]
Problem:

Is \( x = y \) valid at program point 3?

What relation holds at program point 7?

⇒ Polynomial invariants
**Problem:**

**Question**

Is $x = y$ valid at program point 3?

**Question**

What relation holds at program point 7?

$\Rightarrow$ Polynomial invariants
Problem:

Question
Is $x = y$ valid at program point 3?

Question
What relation holds at program point 7?

$\Rightarrow$ Polynomial invariants
Valid invariants:

Power sum

The example program calculates the square power sum \( x = \sum_{y=0}^{n} y^2 \), therefore

\[
x = \frac{2y^3 + 3y^2 + y}{6}
\]

holds at program point 7

Question

⇒ but how to automate this cognition?
Valid invariants:

1. \( x := 0 \)
2. \( x = 0 \)
3. \( y := 0 \)
   \( (y - n \neq 0) \)
4. \( y := y + 1 \)
5. \( x := y \cdot y + x \)
6. \( 2y^3 + 3y^2 + y - 6x = 0 \)
7. \( 2y^3 + 3y^2 + y - 6x = 0 \)

Power sum

The example program calculates the square power sum \( x = \sum_{y=0}^{n} y^2 \), therefore

\[
    x = \frac{2y^3 + 3y^2 + y}{6}
\]

holds at program point 7

Question

⇒ but how to automate this cognition?
## Related work

### Approaches with ideals

<table>
<thead>
<tr>
<th>Author(s)</th>
<th>Title</th>
<th>Year</th>
</tr>
</thead>
<tbody>
<tr>
<td>M.M.O, H.S.</td>
<td>Computing Polynomial Program Invariants</td>
<td>2004</td>
</tr>
<tr>
<td>S.S., H.B.S., Z.M.</td>
<td>Non-linear Loop Invariant Generation</td>
<td>2004</td>
</tr>
</tbody>
</table>

### Approach with modules

<table>
<thead>
<tr>
<th>Author(s)</th>
<th>Title</th>
<th>Year</th>
</tr>
</thead>
<tbody>
<tr>
<td>M.P.</td>
<td>Berechnung von polynomiellen Invarianten</td>
<td>2004</td>
</tr>
</tbody>
</table>

### Initial point

Interpret program states as ideals of polynomials; Store generators of the ideal as representation → M.M.O., H.S.
## Related work

### Approaches with ideals

<table>
<thead>
<tr>
<th>Authors</th>
<th>Title</th>
<th>Year</th>
</tr>
</thead>
<tbody>
<tr>
<td>E.RC, D.K.</td>
<td>Program Verification Using Automatic Generation Of Invariants</td>
<td>2004</td>
</tr>
<tr>
<td>M.MO, H.S.</td>
<td>Computing Polynomial Program Invariants</td>
<td>2004</td>
</tr>
<tr>
<td>S.S., H.B.S., Z.M.</td>
<td>Non-linear Loop Invariant Generation</td>
<td>2004</td>
</tr>
</tbody>
</table>

### Approach with modules

<table>
<thead>
<tr>
<th>Author</th>
<th>Title</th>
<th>Year</th>
</tr>
</thead>
<tbody>
<tr>
<td>M.P.</td>
<td>Berechnung von polynomiellen Invarianten</td>
<td>2004</td>
</tr>
</tbody>
</table>

### Initial point

Interpret program states as Ideals of polynomials; Store generators of the ideal as representation → M.MO., H.S.
Abstract Model

Polynomial programs...

- modelling control flow with (possibly annotated) edges
- assignments of multivariate polynomial expressions (without division) \( x := y \cdot y + x \)
- method calls \( x := f(y, z) \)
- unknown assignments \( x :=? \)

... with guards

- negative polynomial equality guards \((y - n) \neq 0\)
- positive polynomial equality guards \((y - n) = 0\)
- non deterministic choice for the rest \(skip\)

→ Goal: inferring all valid polynomial relations
Abstract Model

Polynomial programs...

- modelling control flow with (possibly annotated) edges
- assignments of multivariate polynomial expressions (without division) \( x := y \cdot y + x \)
- method calls \( x := f(y, z) \)
- unknown assignments \( x := ? \)

... with guards

- negative polynomial equality guards \((y - n) \neq 0\)
- positive polynomial equality guards \((y - n) = 0\)
- non deterministic choice for the rest \(skip\)

→ Goal: inferring all valid polynomial relations
Abstract Model

Polynomial programs...

- modelling control flow with (possibly annotated) edges
- assignments of multivariate polynomial expressions (without division)\[ x := y \cdot y + x \]
- method calls \[ x := f(y, z) \]
- unknown assignments \[ x := ? \]

... with guards

- negative polynomial equality guards \[ (y - n) \neq 0 \]
- positive polynomial equality guards \[ (y - n) = 0 \]
- non deterministic choice for the rest \[ skip \]

→ Goal: inferring all valid polynomial relations
Abstract Model

Polynomial programs...

- modelling control flow with (possibly annotated) edges
- assignments of multivariate polynomial expressions (without division) $x := y \cdot y + x$
- method calls $x := f(y, z)$
- unknown assignments $x := ?$

... with guards

- negative polynomial equality guards $(y - n) \neq 0$
- positive polynomial equality guards $(y - n) = 0$
- non deterministic choice for the rest $skip$

→ Goal: inferring all valid polynomial relations
Intraprocedural example

```
squarepowsum (n ∈ N) ∈ N {
    x, y ∈ N;
    x ← 0, y ← 0;
    while (y ≠ n){
        y ← y + 1;
        x ← y · y + x;
    }
    return x;
}
```

State abstraction
Still, we have to find an abstraction for program states that serves our analysis...
Intraprocedural example

\[ \text{squarepowsum} \ (n \in \mathbb{N}) \in \mathbb{N} \{ \]
\[ x, y \in \mathbb{N}; \]
\[ x \leftarrow 0; y \leftarrow 0; \]
\[ \text{while } (y \neq n) \{ \]
\[ \quad y \leftarrow y + 1; \]
\[ \quad x \leftarrow y \cdot y + x; \]
\[ \} \]
\[ \text{return } x; \]
\[ \}

State abstraction
Still, we have to find an abstraction for program states that serves our analysis...
State abstraction

Polynomials

Polynomials are expressed by equations from the set $\mathbb{R}[X]$, polynomials over $\mathbb{R}$ and the variables from $X$, for example $x - y^2 + 25 = 0$.

Polynomial relations

1. $\forall_{PR[s] \subseteq \mathbb{R}[X]} \exists_{p \in PR[s]} \ \forall_{c \in \mathbb{R} \cup \{x\}} \Rightarrow c \cdot p \in PR[s]

2. $\forall_{PR[s] \subseteq \mathbb{R}[X]} \exists_{p \in PR[s]} \ \forall_{q \in PR[s], \circ \in \{+,-,\cdot\}} \Rightarrow q \circ p \in PR[s]

Polynomial ideals – finitely generated

Polynomial ideals are infinite sets of polynomials, with the upper properties. All ideals can be represented by a minimal number of generating polynomials. For example $\langle \{x - y^2 + 25, x^2 - z\} \rangle$
Verifying polynomial relations

1. $x := 0$
2. $y := 0$
3. $(y - n \neq 0)$
4. $y := y + 1$
5. $x := y \cdot y + x$
6. $(y - n = 0)$
7. $\langle 2y^3 + 3y^2 + y - 6x \rangle$

Fixpoint analysis

- Associating program states with polynomial ideals.
- Verifying polynomials
  - Computing the weakest precondition for a polynomial invariant ideal
- Weakest precondition
  - The only valid precondition can only be the relation $0 = 0$. 
Incremental fixpoint iteration: semantics

\[
\begin{align*}
S & \xrightarrow{PR'} \quad x_j := \text{?} \\
S & \xrightarrow{PR} \quad \{ q_i \mid q_i = \text{coeffs}(q, x_j^i) \} \\
S & \xrightarrow{PR'} \quad (p = 0) \\
S & \xrightarrow{PR} \quad q + a_e \cdot p \\
S & \xrightarrow{PR'} \quad (p \neq 0) \\
S & \xrightarrow{PR} \quad p \cdot q \\
S & \xrightarrow{PR'} \quad x = f(x) \\
S & \xrightarrow{PR} \quad ? \\
\end{align*}
\]
Incremental fixpoint iteration: semantics

```
incremental iteration: semantics

\[
x_j := p
\]

Recalculation of ideals at each iteration step is expensive
⇒ Only new generators \( q \) have to be propagated via edges.
```
Verifying polynomial relations

Fixpoint analysis
Associating program states with polynomial ideals.

Verifying polynomials
Computing the weakest precondition for a polynomial invariant ideal

Weakest precondition
The only valid precondition can only be the relation $0 = 0$. 

Motivation

Model

Intraprocedural analysis

Interprocedural analysis

Conclusion
Verifying polynomial relations

Fixpoint analysis
Associating program states with polynomial ideals.

Verifying polynomials
Computing the weakest precondition for a polynomial invariant ideal

Weakest precondition
The only valid precondition can only be the relation $0 = 0$. 

Verifying polynomial relations

<table>
<thead>
<tr>
<th>1</th>
<th>$x := 0$</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>$y := 0$</td>
</tr>
<tr>
<td>3</td>
<td>$(y - n \neq 0)$</td>
</tr>
<tr>
<td>4</td>
<td>$y := y + 1$</td>
</tr>
<tr>
<td>5</td>
<td>$x := y \cdot y + x$</td>
</tr>
<tr>
<td>6</td>
<td>$y - n = 0$</td>
</tr>
<tr>
<td>7</td>
<td>$\langle 2y^3 + 3y^2 + y - 6x \rangle$</td>
</tr>
</tbody>
</table>
\textbf{Verifying polynomial relations}

\begin{itemize}
\item \textbf{Intraprocedural analysis}
  \item \textbf{Interprocedural analysis}
\item \textbf{Conclusion}
\end{itemize}

\textbf{Fixpoint analysis}

Associating program states with polynomial ideals.

\textbf{Verifying polynomials}

Computing the weakest precondition for a polynomial invariant ideal

\textbf{Weakest precondition}

The only valid precondition can only be the relation $0 = 0$. 

Verifying polynomial relations

Fixpoint analysis
Associating program states with polynomial ideals.

Verifying polynomials
Computing the weakest precondition for a polynomial invariant ideal

Weakest precondition
The only valid precondition can only be the relation $0 = 0$. 

Motivation
Model
Intraprocedural analysis
Interprocedural analysis
Conclusion
Verifying polynomial relations

**Fixpoint analysis**
Associating program states with polynomial ideals.

**Verifying polynomials**
Computing the weakest precondition for a polynomial invariant ideal

**Weakest precondition**
The only valid precondition can only be the relation $0 = 0$. 

```
x := 0
y := 0
(y - n \neq 0)
y := y + 1
x := y \cdot y + x
```

```
\langle 0 \rangle
\langle x \rangle
\langle 2y^3 + 3y^2 + y - 6x \rangle
\langle 2y^3 + 3y^2 + y - 6x \rangle
\langle 2y^3 - 3y^2 + y - 6x \rangle
\langle 2y^3 + 3y^2 + y - 6x \rangle
\langle 2y^3 + 3y^2 + y - 6x \rangle
```
Verifying polynomial relations

Fixpoint analysis
Associating program states with polynomial ideals.

Verifying polynomials
Computing the weakest precondition for a polynomial invariant ideal

Weakest precondition
The only valid precondition can only be the relation $0 = 0$. 

Motivation
Model
Intraprocedural analysis
Interprocedural analysis
Conclusion
Verifying polynomial relations

1. $x := 0$
   - $\langle 0 \rangle$
2. $y := 0$
   - $\langle x \rangle$
3. $(y - n \neq 0)$
   - $\langle 2y^3 + 3y^2 + y - 6x \rangle$
4. $y := y + 1$
   - $\langle 2y^3 + 3y^2 + y - 6x \rangle$
   - skip
5. $x := y \cdot y + x$
   - $\langle 2y^3 - 3y^2 + y - 6x \rangle$
6. $\langle 2y^3 + 3y^2 + y - 6x \rangle$
7. $(y - n = 0)$
   - $\langle 2y^3 + 3y^2 + y - 6x \rangle$

Fixpoint analysis
- Associating program states with polynomial ideals.

Verifying polynomials
- Computing the weakest precondition for a polynomial invariant ideal.

Weakest precondition
- The only valid precondition can only be the relation $0 = 0$. 

Motivation

Model

Intraprocedural analysis

Interprocedural analysis

Conclusion
Infering polynomial relations

Weakest precondition
Evaluating the WP provides values for the generic parameters

Inferring relations
The weakest precondition for a generic polynomial of degree $n$. E.g:

$$\sum_{0 \leq i_1 + \ldots + i_k \leq d} a_{i_1, \ldots, i_k} \cdot x_1^{i_1} \cdot \ldots \cdot x_k^{i_k}$$

$$\langle ax^2 + by^2 + cxy + dx + ey + f \rangle$$
Infering polynomial relations

Weakest precondition
Evaluating the WP provides values for the generic parameters

Inferring relations
The weakest precondition for a generic polynomial of degree $n$. E.g:

$$\sum_{0 \leq i_1 + \ldots + i_k \leq d} a_{i_1, \ldots, i_k} \cdot x_1^{i_1} \cdot \ldots \cdot x_k^{i_k}$$
Performance issues

Bad news

- Reductions on polynomial ideals are perform doubly exponentially on the number of participating variables.
- Ideal membership is in general EXPSPACE-hard
- Ideal membership is NP-hard for fixed number of variables

Problem

Using generic polynomials with many variables turns polynomial reductions infeasible.

⇒ Observation: Generic variables don’t occur in programs, merely model the structure of invariants; they also contribute linear to the polynomials

Idea

Mark generic variables for special treatment in the reduction algorithm.
⇒ Model of vectors and Modules
Performance issues

Bad news

- Reductions on polynomial ideals are perform doubly exponentially on the number of participating variables.
- Ideal membership is in general EXPSPACE-hard
- Ideal membership is NP-hard for fixed number of variables

Problem

Using generic polynomials with many variables turns polynomial reductions infeasible.

⇒ Observation: Generic variables don’t occur in programs, merely model the structure of invariants; they also contribute linear to the polynomials

Idea

Mark generic variables for special treatment in the reduction algorithm.
⇒ Model of vectors and Modules
Performance issues

Bad news

- Reductions on polynomial ideals are perform doubly exponentially on the number of participating variables.
- Ideal membership is in general EXPSPACE-hard
- Ideal membership is NP-hard for fixed number of variables

Problem

Using generic polynomials with many variables turns polynomial reductions infeasible.

⇒ Observation: Generic variables don't occur in programs, merely model the structure of invariants; they also contribute linear to the polynomials

Idea

Mark generic variables for special treatment in the reduction algorithm.
⇒ Model of vectors and Modules
Interprocedural analysis

\[ \text{squarepowsum} \ (n \in \mathbb{N}) \in \mathbb{N} \{ \]
\[
    x, y \in \mathbb{N}; \\
    x \leftarrow 0, y \leftarrow 0; \\
    \text{while } (y \neq n) \{ \\
        y \leftarrow y + 1; \\
        x \leftarrow \text{sqr}(y) + x; \\
    \} \\
    \text{return } x; \\
\}
\]

\[ \text{sqr} \ (x \in \mathbb{N}) \in \mathbb{N} \{ \\
    \text{return } x \cdot x; \\
\}
\]

\[ \text{return} := x \cdot x \]
Incremental fixpoint iteration: semantics

\[ x_j := ? \]

\[ \{ q_i \mid q_i = \text{coeffs}(q, x_j) \} \]

\[ (p = 0) \]

\[ q + a_e \cdot p \]

\[ (p \neq 0) \]

\[ p \cdot q \]

\[ x_i = f(x) \]
Method call details

Idea

Use precomputed templates to carry the effect of each method call.

\[
\langle (a + b + d - 3c)y_2 + ay_3 + d \rangle \\
\rightarrow \text{But: Has yet to be implemented}
\]
Method call details

**Idea**

Use precomputed templates to carry the effect of each method call.

$$
\langle (a + b + d - 3c)y_2 + ay_3 + d \rangle
$$

$$
\langle ax_1 + bx_2 + c(x_1 + x_2) + d \rangle
$$

$$
\langle (a - c)y_1 + (b + d - 2c)y_2 + cy_3 + d \rangle
$$

$$
\xrightarrow{\text{But}: \text{Has yet to be implemented}}
$$
**Motivation**

**Model**

Intraprocedural analysis

Interprocedural analysis

**Conclusion**

---

**Complete analysis**

Set fixpointiteration (Node $u_t$, Vector $v_t$, Set Vars, Set Edges, Set Nodes) {
  Set [] $G \leftarrow$ new Set[|Nodes|];
  forall (u ∈ Nodes) $G[u] \leftarrow \emptyset$;
  Set $W \leftarrow \{(v_t, u_t)\};$
  while ($W \neq \emptyset$) {
    ($v, t$) $\leftarrow$ extract($W$);
    $v \leftarrow$ reduce($v, G[t]$);
    if ($v \neq 0$) {
      $G[t] \leftarrow G[t] \cup \{v\};$
      forall ( (s, "skip" , t) ∈ Edges)
        $W \leftarrow W \cup \{(v, s)\};$
      forall ( (s, "x_j := p" , t) ∈ Edges)
        $W \leftarrow W \cup \{(v[p/x_j], s)\};$
      forall ( (s, "(p \neq 0)" , t) ∈ Edges)
        $W \leftarrow W \cup \{(p \cdot v, s)\};$
      forall ( (s, "x_j :=?" , t) ∈ Edges)
        let $l = \max\{i | ax^i \in \text{monoms}(v)\} \)$
        in let $v \Rightarrow (p_0 x_0^0 + \ldots + p_0 x_0^l, \ldots, p_k x_0^0 + \ldots + p_k x_0^l) \)$
        in let $v_i \leftarrow (p_{0_i}, p_{1_i}, \ldots, p_{k_i}) \)$
        in $W \leftarrow W \cup \{(v_0, u), \ldots, (v_l, u)\};$
    }
  }
  return $\langle G[u_{start}] \rangle$;
## Benchmarks

<table>
<thead>
<tr>
<th>Name</th>
<th>Calculation</th>
<th>ass-deg</th>
<th>Invariant</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>geoSeries1</td>
<td>$x = (z - 1) \cdot \sum_{k=0}^{K} z^k$</td>
<td>$y = z^K$</td>
<td>$x = y - 1$</td>
<td>0, 356s</td>
</tr>
<tr>
<td>geoSeries2</td>
<td>$x = \sum_{k=0}^{K} z^k$</td>
<td>$y = z^{K-1}$</td>
<td>$x \cdot (z - 1) = yz - 1$</td>
<td>0, 569s</td>
</tr>
<tr>
<td>geoSeries3</td>
<td>$x = \sum_{k=0}^{K} a \cdot z^k$</td>
<td>$y = z^{K-1}$</td>
<td>$x \cdot (z - 1) = ayz - a$</td>
<td>1, 47s</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Name</th>
<th>Calculation</th>
<th>ass-deg</th>
<th>Invariant</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>powSum1</td>
<td>$x = \sum_{k=0}^{K} 1$</td>
<td>$y = \sum_{k=0}^{K} 1$</td>
<td>$x = y$</td>
<td>0, 331s</td>
</tr>
<tr>
<td>powSum2</td>
<td>$x = \sum_{k=0}^{K} k$</td>
<td>$y = \sum_{k=0}^{K} 1$</td>
<td>$2x = y^2 + y$</td>
<td>0, 776s</td>
</tr>
<tr>
<td>powSum3</td>
<td>$x = \sum_{k=0}^{K} k^2$</td>
<td>$y = \sum_{k=0}^{K} 1$</td>
<td>$6x = 2y^3 + 3y^2 + y$</td>
<td>1, 47s</td>
</tr>
<tr>
<td>powSum4</td>
<td>$x = \sum_{k=0}^{K} k^3$</td>
<td>$y = \sum_{k=0}^{K} 1$</td>
<td>$4x = y^4 + 2y^3 + y^2$</td>
<td>2, 71s</td>
</tr>
<tr>
<td>powSum5</td>
<td>$x = \sum_{k=0}^{K} k^4$</td>
<td>$y = \sum_{k=0}^{K} 1$</td>
<td>$30x = 6y^5 + 15y^4 + 10y^3 - y$</td>
<td>10, 3s</td>
</tr>
<tr>
<td>powSum6</td>
<td>$x = \sum_{k=0}^{K} k^5$</td>
<td>$y = \sum_{k=0}^{K} 1$</td>
<td>$12x = 2y^6 + 6y^5 + 5y^4 - y^2$</td>
<td>787, 2s</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Strategy</th>
<th>gs3/5</th>
<th>gs3/6</th>
<th>ps3/5</th>
<th>ps4/5</th>
<th>ps4/6</th>
<th>ps5/5</th>
<th>ps5/6</th>
<th>ps6/6</th>
</tr>
</thead>
<tbody>
<tr>
<td>Original vector</td>
<td>8,4s</td>
<td>29,4s</td>
<td>3,83s</td>
<td>14,7s</td>
<td>. . .</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Reduced vector</td>
<td>7,3s</td>
<td>26,6s</td>
<td>2,9s</td>
<td>3,5s</td>
<td>8,1s</td>
<td>10,9s</td>
<td>30,0s</td>
<td>787s</td>
</tr>
</tbody>
</table>
## Future Work

### Implementation
- Treatment of procedure calls
- Scope on relevant variables
- Face large/real examples

### Theory
Find a better upper complexity bound

⇒ http://www2.cs.tum.edu/~petter/polyinvar
Thank You for Your attention!