

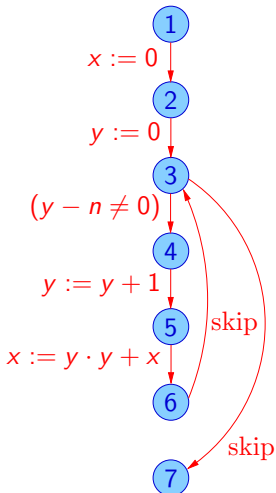
# Inferring polynomial invariants with Polyinvar

Helmut Seidl and Michael Petter

TU-München

Numerical & Symbolic Abstract Domains Workshop, 2005

## Problem:



### Question

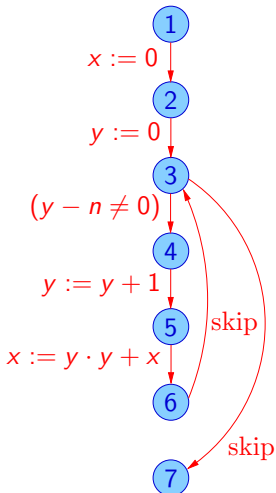
Is  $x - y$  valid at program point 3?

### Question

What relation holds at program point 7?

$\Rightarrow$  *Polynomial invariants*

## Problem:



### Question

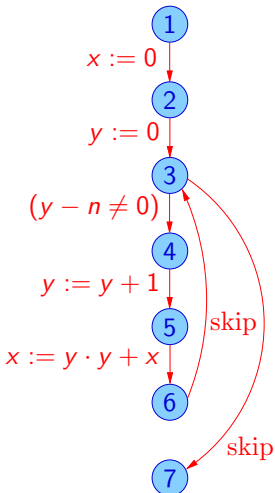
Is  $x - y$  valid at program point 3?

### Question

What relation holds at program point 7?

$\Rightarrow$  *Polynomial invariants*

## Problem:



### Question

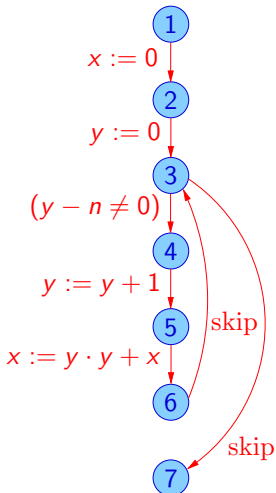
Is  $x - y$  valid at program point 3?

### Question

What relation holds at program point 7?

$\Rightarrow$  *Polynomial invariants*

## Problem:



### Question

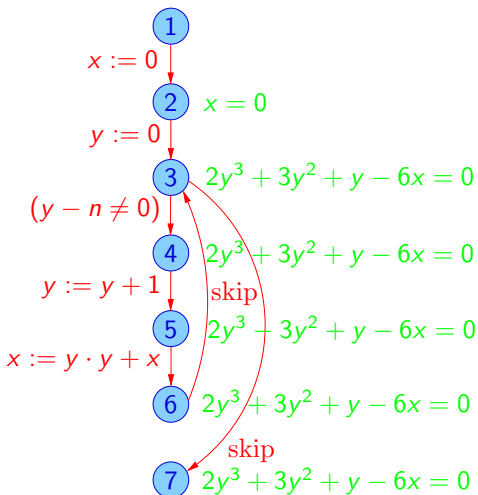
Is  $x - y$  valid at program point 3?

### Question

What relation holds at program point 7?

$\Rightarrow$  *Polynomial invariants*

## Valid invariants:



### Power sum

The example program calculates the square power sum, therefore

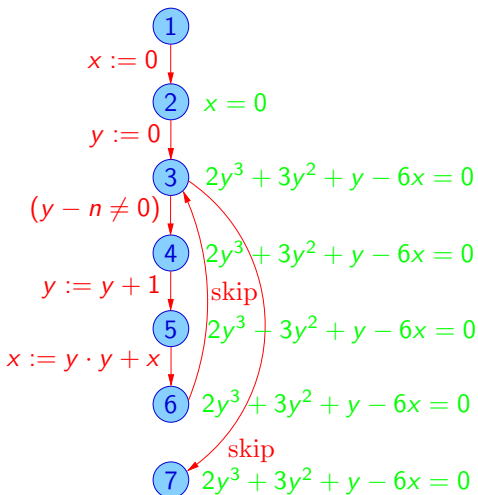
$$2y^3 + 3y^2 + y - 6x = 0$$

holds at program point 7

### Question

⇒ but how to automate this cognition?

## Valid invariants:



### Power sum

The example program calculates the square power sum, therefore

$$2y^3 + 3y^2 + y - 6x = 0$$

holds at program point 7

### Question

⇒ but how to automate this cognition?

## Related work

### Approaches with ideals

E.RC, D.K. Program Verification Using Automatic Generation Of Invariants 2004

M.MO, H.S. Computing Polynomial Program Invariants 2004

S.S., H.B.S., Z.M. Non-linear Loop Invariant Generation 2004

### Approach with modules

M.P. Berechnung von polynomiellen Invarianten 2004

### Initial point

Interpret program states as Ideals of polynomials;  
Store generators of the ideal as representation

→ M.MO., H.S.

## Related work

### Approaches with ideals

E.RC, D.K. Program Verification Using Automatic Generation Of Invariants 2004

M.MO, H.S. Computing Polynomial Program Invariants 2004

S.S., H.B.S., Z.M. Non-linear Loop Invariant Generation 2004

### Approach with modules

M.P. Berechnung von polynomiellen Invarianten 2004

### Initial point

Interpret program states as Ideals of polynomials;  
Store generators of the ideal as representation

→ M.MO., H.S.

# Abstract Model

## Polynomial programs...

- polynomial assignments  $x := y \cdot y + x$
- unknown assignments  $x := ?$

## ... with guards

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

→ **Goal:** inferring all valid polynomial relations

# Abstract Model

## Polynomial programs...

- polynomial assignments  $x := y \cdot y + x$
- unknown assignments  $x := ?$

## ... with guards

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

→ **Goal:** inferring all valid polynomial relations

# Abstract Model

## Polynomial programs...

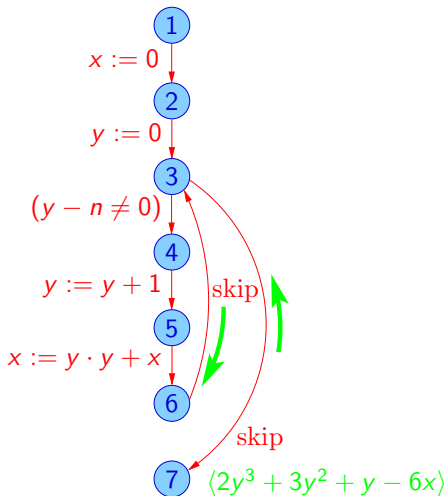
- polynomial assignments  $x := y \cdot y + x$
- unknown assignments  $x := ?$

## ... with guards

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

→ **Goal:** inferring all valid polynomial relations

# Verifying polynomial relations



## Verifying polynomials

Computing the weakest precondition for a polynomial invariant

## Inferring relations

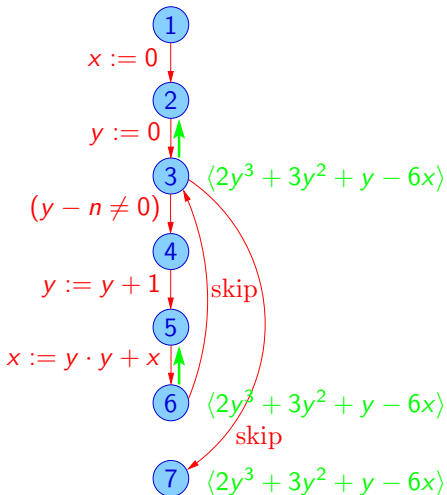
The weakest precondition for a generic polynomial of degree  $n$ . E.g:

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

## Problem

Number of generic variables turns polynomial reductions infeasible

# Verifying polynomial relations



## Verifying polynomials

Computing the weakest precondition for a polynomial invariant

## Inferring relations

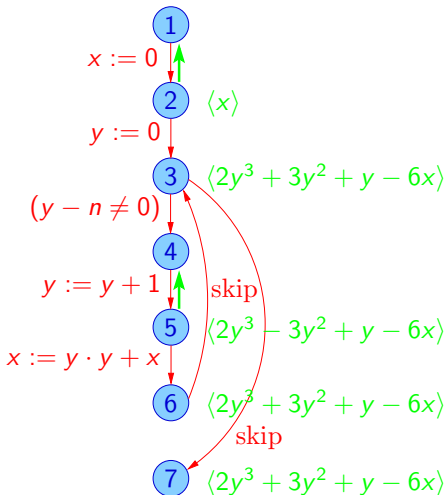
The weakest precondition for a generic polynomial of degree  $n$ . E.g:

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

## Problem

Number of generic variables turns polynomial reductions infeasible

# Verifying polynomial relations



## Verifying polynomials

Computing the weakest precondition for a polynomial invariant

## Inferring relations

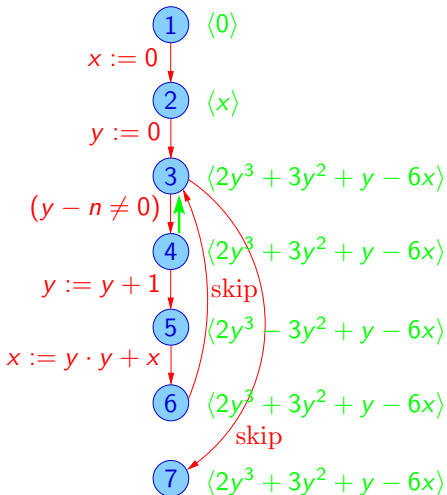
The weakest precondition for a generic polynomial of degree  $n$ . E.g:

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

## ▽ Problem

Number of generic variables turns polynomial reductions infeasible

# Verifying polynomial relations



## Verifying polynomials

Computing the weakest precondition for a polynomial invariant

## Inferring relations

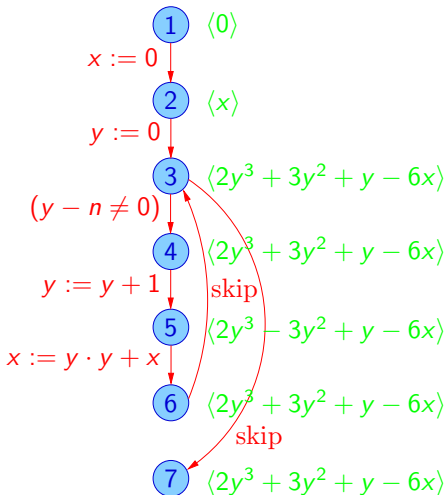
The weakest precondition for a generic polynomial of degree  $n$ . E.g:

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

## Problem

Number of generic variables turns polynomial reductions infeasible

# Verifying polynomial relations



## Verifying polynomials

Computing the weakest precondition for a polynomial invariant

## Inferring relations

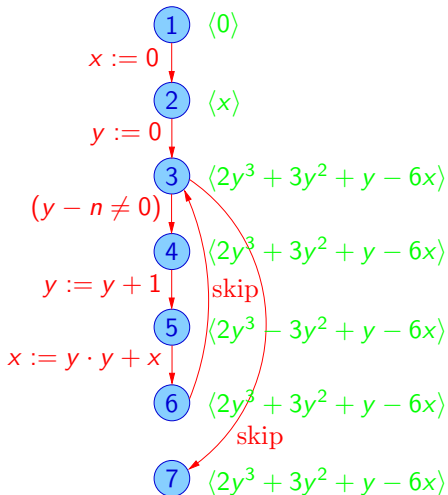
The weakest precondition for a generic polynomial of degree  $n$ . E.g:

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

## ▽ Problem

Number of generic variables turns polynomial reductions infeasible

# Verifying polynomial relations



## Verifying polynomials

Computing the weakest precondition for a polynomial invariant

## Inferring relations

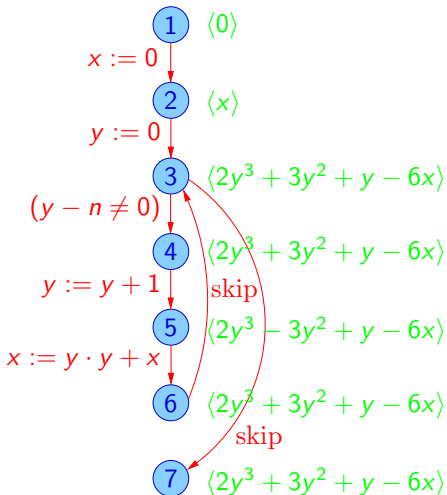
The weakest precondition for a generic polynomial of degree  $n$ . E.g:

$$\sum_{0 \leq i_1 + \dots + i_k \leq d} \mathbf{a}_{i_1, \dots, i_k} \cdot \mathbf{x}_1^{i_1} \cdot \dots \cdot \mathbf{x}_k^{i_k}$$

## ▽ Problem

Number of generic variables turns polynomial reductions infeasible

# Verifying polynomial relations



## Verifying polynomials

Computing the weakest precondition for a polynomial invariant

## Inferring relations

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

$$\sum_{0 \leq i_1 + \dots + i_k \leq d} \mathbf{a}_{i_1, \dots, i_k} \cdot \mathbf{x}_1^{i_1} \cdot \dots \cdot \mathbf{x}_k^{i_k}$$

## ▽ Problem

Number of generic variables turns polynomial reductions infeasible

# 1<sup>st</sup> Improvement: From Ideals to Modules

## Ideal

Ideals  $I \subseteq \mathbb{Q}[x_1, \dots, x_k]$  are sets of polynomials with:

- $q_1, q_2 \in I \Rightarrow q_1 + q_2 \in I$ ;
- $q \in I \Rightarrow \forall r \in \mathbb{Q}[x_1, \dots, x_k] r \cdot q \in I$ ;

## Module

Modules  $M \subseteq \mathbb{Q}[x_1, \dots, x_k]^n$  are sets of vectors of polynomials with similar properties as Ideals:

- $v_1, v_2 \in M \Rightarrow v_1 + v_2 \in M$ ;
- $v \in M \Rightarrow \forall r \in \mathbb{Q}[x_1, \dots, x_k] r \cdot v \in M$ ;

⇒ Why change?

# 1<sup>st</sup> Improvement: From Ideals to Modules

## Ideal

Ideals  $I \subseteq \mathbb{Q}[x_1, \dots, x_k]$  are sets of polynomials with:

- $q_1, q_2 \in I \Rightarrow q_1 + q_2 \in I$ ;
- $q \in I \Rightarrow \forall r \in \mathbb{Q}[x_1, \dots, x_k] r \cdot q \in I$ ;

## Module

Modules  $M \subseteq \mathbb{Q}[x_1, \dots, x_k]^n$  are sets of vectors of polynomials with similar properties as Ideals:

- $v_1, v_2 \in M \Rightarrow v_1 + v_2 \in M$ ;
- $v \in M \Rightarrow \forall r \in \mathbb{Q}[x_1, \dots, x_k] r \cdot v \in M$ ;

**⇒ Why change?**

# Expressing generic polynomials with vectors

## $d$ -generic polynomial structure

$$p = \sum_{0 \leq i_1 + \dots + i_k \leq d} \mathbf{a}_{i_1, \dots, i_k} \cdot \mathbf{x}_1^{i_1} \cdot \dots \cdot \mathbf{x}_k^{i_k}$$

## corresponding vector

$$\mathbf{v} = \sum_{0 \leq i_1 + \dots + i_k \leq d} \mathbf{x}_1^{i_1} \cdot \dots \cdot \mathbf{x}_k^{i_k} \cdot \text{gen}_{i_1, \dots, i_k}$$

## Observation

The generic variables  $\mathbf{a}_{i,\dots}$  disappear in the vector representation

# Expressing generic polynomials with vectors

## $d$ -generic polynomial structure

$$p = \sum_{0 \leq i_1 + \dots + i_k \leq d} \mathbf{a}_{i_1, \dots, i_k} \cdot \mathbf{x}_1^{i_1} \cdot \dots \cdot \mathbf{x}_k^{i_k}$$

## corresponding vector

$$\mathbf{v} = \sum_{0 \leq i_1 + \dots + i_k \leq d} \mathbf{x}_1^{i_1} \cdot \dots \cdot \mathbf{x}_k^{i_k} \cdot \mathbf{gen}_{i_1, \dots, i_k}$$

## Observation

The generic variables  $\mathbf{a}_{i,\dots}$  disappear in the vector representation

# Expressing generic polynomials with vectors

## $d$ -generic polynomial structure

$$p = \sum_{0 \leq i_1 + \dots + i_k \leq d} \mathbf{a}_{i_1, \dots, i_k} \cdot \mathbf{x}_1^{i_1} \cdot \dots \cdot \mathbf{x}_k^{i_k}$$

## corresponding vector

$$\mathbf{v} = \sum_{0 \leq i_1 + \dots + i_k \leq d} \mathbf{x}_1^{i_1} \cdot \dots \cdot \mathbf{x}_k^{i_k} \cdot \mathbf{gen}_{i_1, \dots, i_k}$$

## Observation

The generic variables  $\mathbf{a}_{i...}$  disappear in the vector representation

# Modules and Gröbner Bases

## Finite Representation

For an implementation, these Modules have to be representable effectively. Hilbert's basis theorem can also be applied to modules, which says, that Modules can be generated finitely just like Ideals.

## Efficient reductions

Gröbner base calculations, which are used for reductions, can be calculated more efficiently on Modules than on equivalent Ideals.

⇒ B.W.93

⇒ crucial speedup

# Modules and Gröbner Bases

## Finite Representation

For an implementation, these Modules have to be representable effectively. Hilbert's basis theorem can also be applied to modules, which says, that Modules can be generated finitely just like Ideals.

## Efficient reductions

Gröbner base calculations, which are used for reductions, can be calculated more efficiently on Modules than on equivalent Ideals.

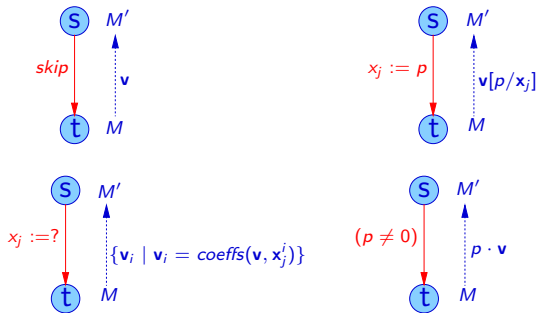
⇒ B.W.93

⇒ crucial speedup

## 2<sup>nd</sup> Improvement: Incremental fixpoint iteration

### Propagate only new generators

Recalculation of Modules at each iteration step is expensive  
 ⇒ Only new generators  $\mathbf{v}$  have to be propagated via edges.



## 3<sup>rd</sup> Improvement: Optimized generator treatment

### Conventional way

When the fixpoint iteration reaches a program state, three things have to be done:

- 1 test, whether the propagated vector is not member of the module
- 2 if it is new, it has to be added to the module
- 3 if it is new, it has to be propagated through all inbound edges

### Idea

The member test involves reducing the vector by means of the module;  
How does the reduced vector compare to the original one?

# Generator selection

## Theory

Reduction of a vector by a module can increase its length twice exponentially

## Heuristic approach

Compare original vector to reduced one and decide in favor of lesser degree and lesser length

## Practice

Simple reduction of a vector indeed seems to be the right decision

⇒ Simple reduction

# Generator selection

## Theory

Reduction of a vector by a module can increase its length twice exponentially

## Heuristic approach

Compare original vector to reduced one and decide in favor of lesser degree and lesser length

## Practice

Simple reduction of a vector indeed seems to be the right decision

⇒ **Simple reduction**

# Complete analysis

```

Set fixpointiteration (Node  $u_t$ , Vector  $v_t$ , Set Vars, Set Edges, Set Nodes) {
  Set []  $G \leftarrow$  new Set[|Nodes|];
  forall ( $u \in$  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 ( $(s, \text{"skip"}, t) \in$  Edges)
         $W \leftarrow W \cup \{(v, s)\}$ ;
      forall ( $(s, \text{"x}_j := p''$ ,  $t) \in$  Edges)
         $W \leftarrow W \cup \{(v[p/x_j], s)\}$ ;
      forall ( $(s, \text{"(p \neq 0)"}$ ,  $t) \in$  Edges)
         $W \leftarrow W \cup \{(p \cdot v, s)\}$ ;
      forall ( $(s, \text{"x}_j := ?"$ ,  $t) \in$  Edges)
        let  $l = \max(\{i \mid ax^i \in \text{monoms}(v)\})$ 
          in let  $v \Rightarrow (p_{0_0} x_j^0 + \dots + p_{0_l} x_j^l, \dots, p_{k_0} x_j^0 + \dots + p_{k_l} x_j^l)$ 
            in let  $v_i \leftarrow (p_{0_i}, p_{1_i}, \dots, p_{k_i})$ 
              in  $W \leftarrow W \cup \{(v_0, u), \dots, (v_l, u)\}$ ;
    }
  }
  return  $\langle G[u_{start}] \rangle$ ;
}

```

# Benchmarks

Name	Calculation	ass-deg	Invariant	Time	
geoSeries1	$x = (z - 1) \cdot \sum_{k=0}^K z^k$	$y = z^K$	$\leq 2$	$x = y - 1$	0, 356s
geoSeries2	$x = \sum_{k=0}^K z^k$	$y = z^{K-1}$	$\leq 2$	$x \cdot (z - 1) = yz - 1$	0, 569s
geoSeries3	$x = \sum_{k=0}^K a \cdot z^k$	$y = z^{K-1}$	$\leq 2$	$x \cdot (z - 1) = azy - a$	1, 47s

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

Strategy	gs3/5	gs3/6	ps3/5	ps4/5	ps4/6	ps5/5	ps5/6	ps6/6
Original vector	8,4s	29,4s	3,83s	14,7s	...			
Reduced vector	7,3s	26,6s	2,9s	3,5s	8,1s	10,9s	30,0s	787s

# Future Work

## Abstraction

Treatment of equality guards

## Implementation

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

## Theory

Find an upper complexity bound

$\implies$  <http://www2.cs.tum.edu/~petter/polyinvar>



**Thank You for Your attention!**