Inferring polynomial invariants

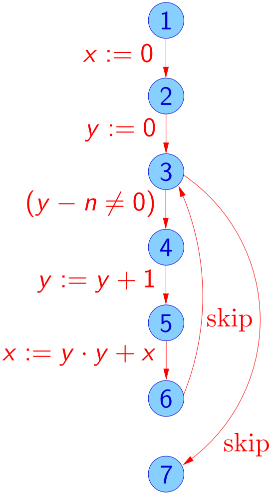
with Polyinvar

Helmut Seidl and Michael Petter

TU-München

Chair Workshop, 2005

Problem:



Question

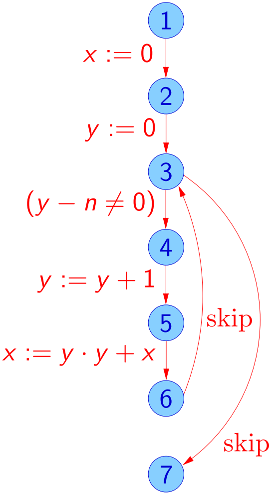
Is $x = y$ valid at program point 3?

Question

What relation holds at program point 7?

$\Rightarrow$ *Polynomial invariants*

Problem:



Question

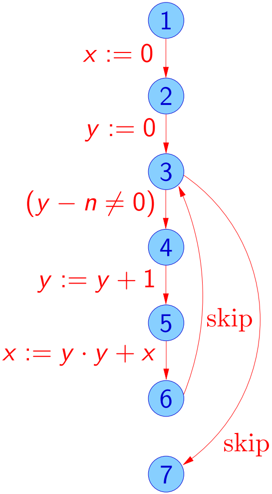
Is $x = y$ valid at program point 3?

Question

What relation holds at program point 7?

$\Rightarrow$ *Polynomial invariants*

Problem:



Question

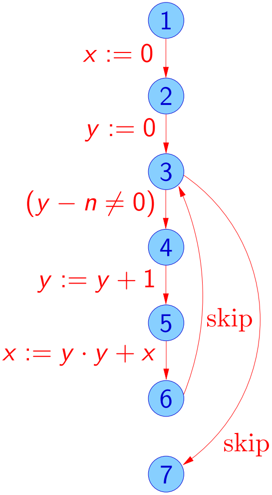
Is $x = y$ valid at program point 3?

Question

What relation holds at program point 7?

$\Rightarrow$ *Polynomial invariants*

Problem:



Question

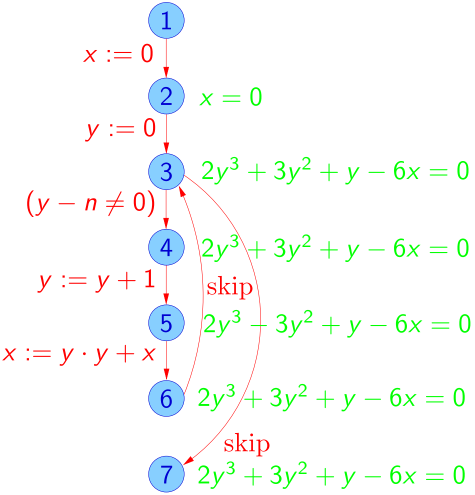
Is $x = y$ valid at program point 3?

Question

What relation holds at program point 7?

$\Rightarrow$ *Polynomial invariants*

Valid invariants:



Power sum

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

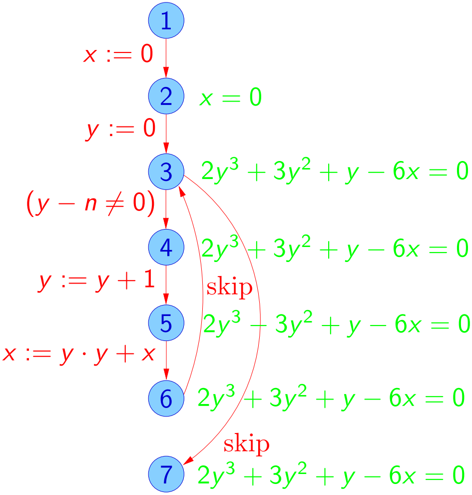
$$x = \frac{2y^3 + 3y^2 + y}{6}$$

holds at program point 7

Question

$\Rightarrow$ but how to automate this cognition?

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

$\Rightarrow$ 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...

- 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 $x :=?$

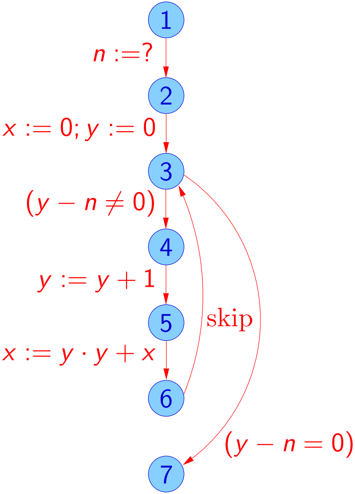
... with guards

- 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

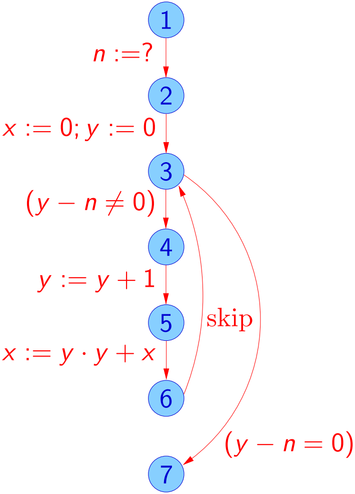
Still, we have to find an abstraction for program states that serves our analysis...

Intraprocedural example

```
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...



State abstraction

Polynomials

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

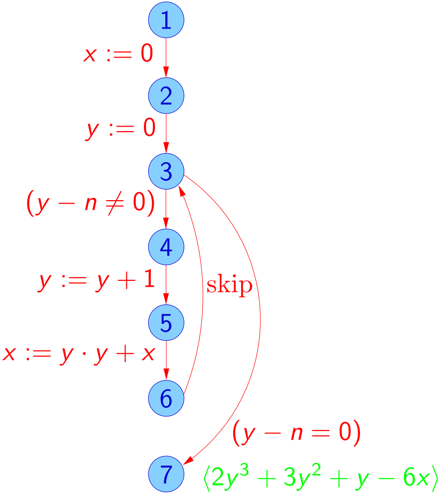
Polynomial relations

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

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



Fixpoint analysis

Associating program states with polynomial ideals.

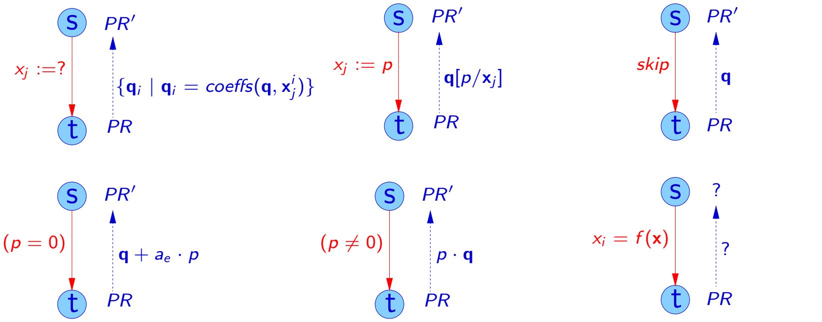
Verifing polynomials

Computing the weakest precondition for a polynomial invariant ideal

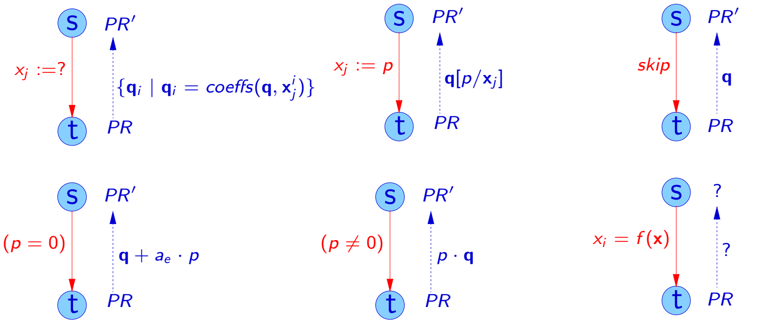
Weakest precondition

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

Incremental fixpoint iteration: semantics



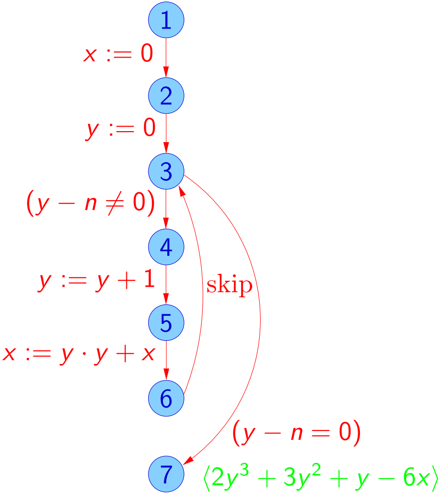
Incremental fixpoint iteration: semantics



Propagate only new generators – incremental iteration

Recalculation of ideals at each iteration step is expensive
$\Rightarrow$ Only new generators $\mathbf{q}$ have to be propagated via edges.

Verifying polynomial relations



Fixpoint analysis

Associating program states with polynomial ideals.

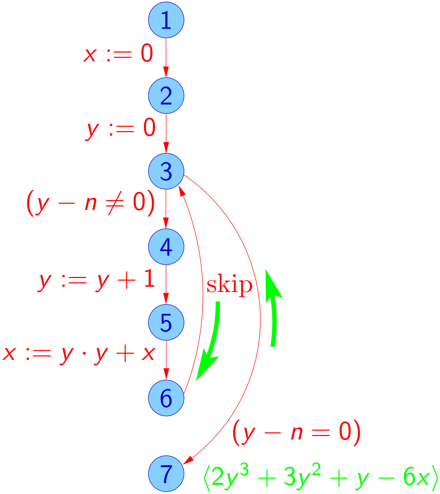
Verifing polynomials

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



Fixpoint analysis

Associating program states with polynomial ideals.

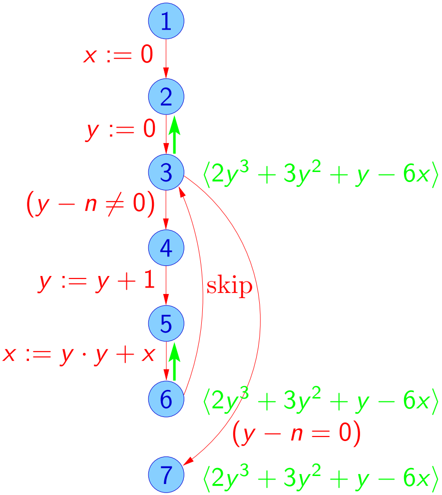
Verifing polynomials

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



Fixpoint analysis

Associating program states with polynomial ideals.

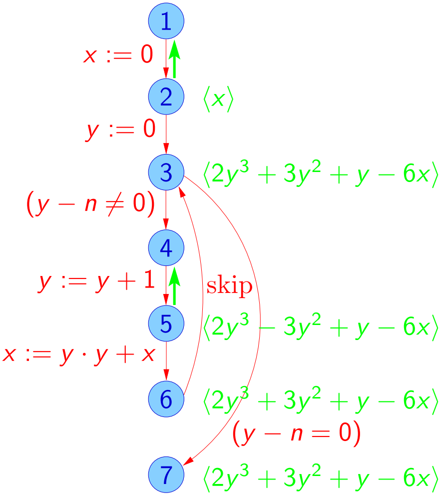
Verifing polynomials

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



Fixpoint analysis

Associating program states with polynomial ideals.

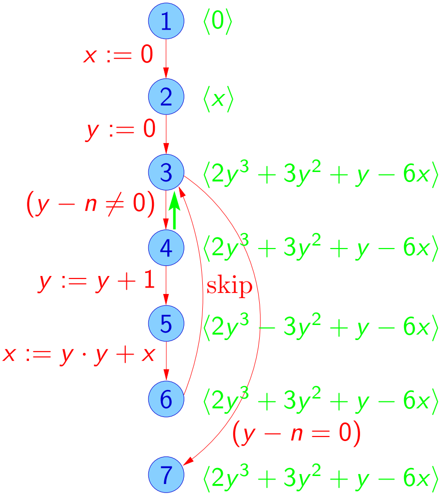
Verifing polynomials

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



Fixpoint analysis

Associating program states with polynomial ideals.

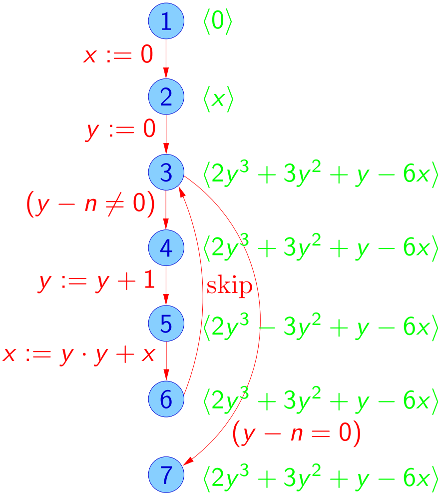
Verifing polynomials

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



Fixpoint analysis

Associating program states with polynomial ideals.

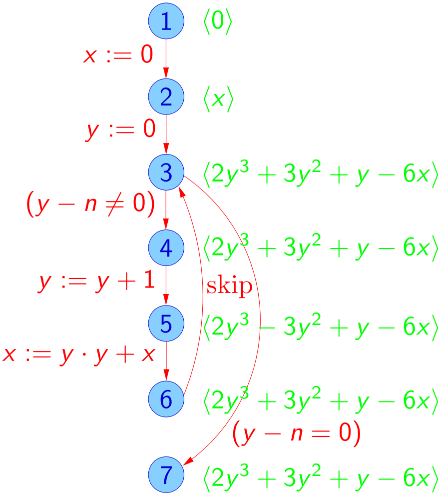
Verifing polynomials

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



Fixpoint analysis

Associating program states with polynomial ideals.

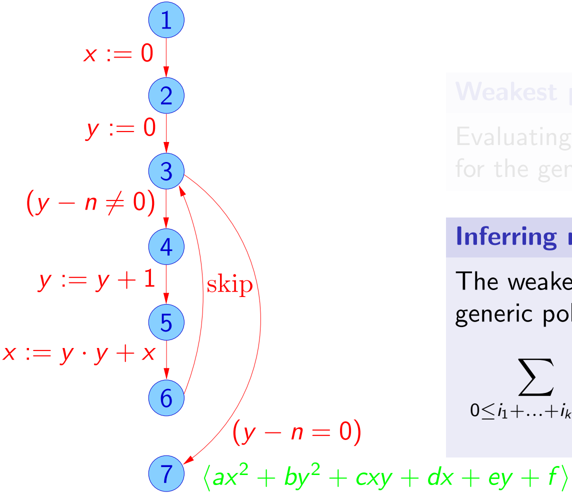
Verifing polynomials

Computing the weakest precondition for a polynomial invariant ideal

Weakest precondition

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

Infering polynomial relations



Weakest precondition

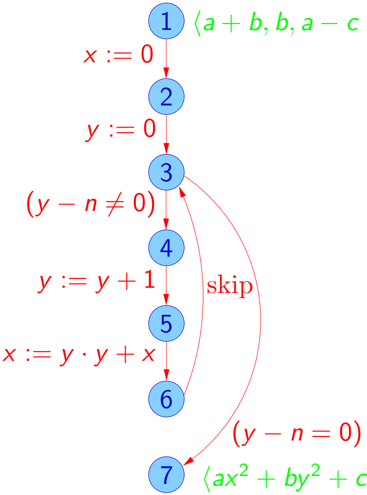
Evaluating the WP provides values for the generic parameters

Inferring relations

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

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

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:

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

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.
$\Longrightarrow$ 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.

$\Rightarrow$ **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.
$\Longrightarrow$ 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.

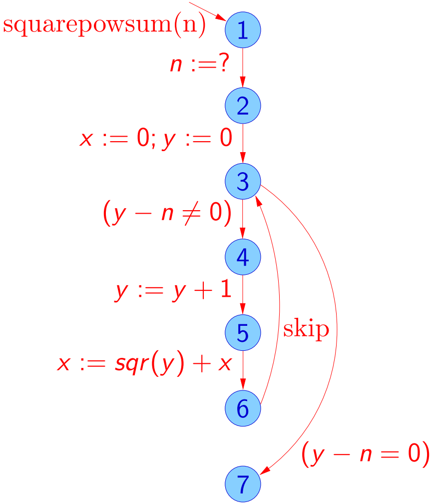
$\Rightarrow$ **Observation:** Generic variables don't occur in programs, merely model the structure of invariants; they also contribute linear to the polynomials

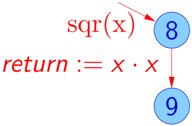
▽ Idea

Mark generic variables for special treatment in the reduction algorithm.
$\Longrightarrow$ Model of vectors and Modules

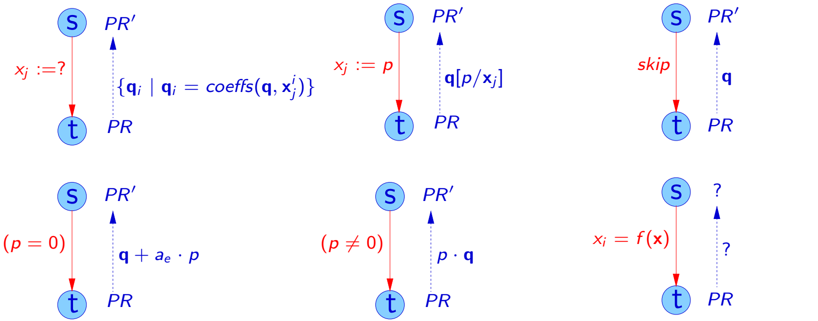
Interprocedural example

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





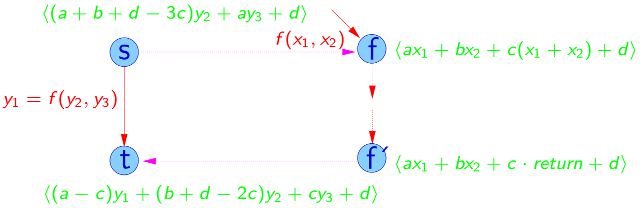
Incremental fixpoint iteration: semantics



Method call details

Idea

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

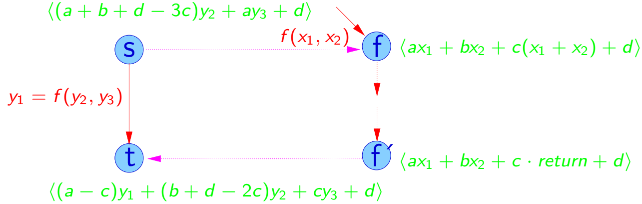


→ **But**: Has yet to be implemented

Method call details

Idea

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



$\rightarrow$ **But**: Has yet to be implemented

Complete analysis

```
Set fixpointiteration (Node u_t, Vector v_t, Set Vars, Set Edges, Set Nodes) {
  Set [] G ⇐ new Set[|Nodes|];
  forall (u ∈ Nodes) G[u] ⇐ ∅;
  Set W ⇐ {(v_t, u_t)};
  while (W ≠ ∅) {
    (v, t) ⇐ extract(W);
    v ⇐ reduce(v, G[t]);
    if (v ≠ 0) {
      G[t] ⇐ G[t] ∪ {v};
      forall ( (s, ''skip;'' , t) ∈ Edges)
        W ⇐ W ∪ {(v, s)};
      forall ( (s, ''x_j := p'' , t) ∈ Edges)
        W ⇐ W ∪ {(v[p/x_j], s)};
      forall ( (s, ''(p ≠ 0)'' , t) ∈ Edges)
        W ⇐ W ∪ {(p · v, s)};
      forall ( (s, ''x_j :=?'' , t) ∈ Edges)
        let l = max({i|ax^i ∈ monoms(v)})
          in let v ⇒ (p_{0_0} x_j^0 + ... + p_{0_l} x_j^l, ..., p_{k_0} x_j^0 + ... + p_{k_l} x_j^l)
            in let v_i ⇐ (p_{0_i}, p_{1_i}, ..., p_{k_i})
              in  W ⇐ W ∪ {(v_0, u), ..., (v_l, u)};
    }
  }
  return ⟨G[u_start]⟩;
}
```

Benchmarks

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

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

Implementation

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

Theory

Find a better upper complexity bound

$\Longrightarrow$ http://www2.cs.tum.edu/~petter/polyinvar

Thank You for Your attention!