Generation of Polynomial Equality Invariants by Abstract Interpretation

Enric Rodríguez-Carbonell
Universitat Politècnica de Catalunya (UPC)
Barcelona

Joint work with **Deepak Kapur** (UNM)

Introduction
Why Care about Invariants ? (1)

- It is necessary to verify *safety* properties of systems:

  no program execution reaches an erroneous state

  (state = values of variables)

- For instance in:
  - Imperative programs
  - Reactive systems
  - Concurrent systems
  - ...

Introduction

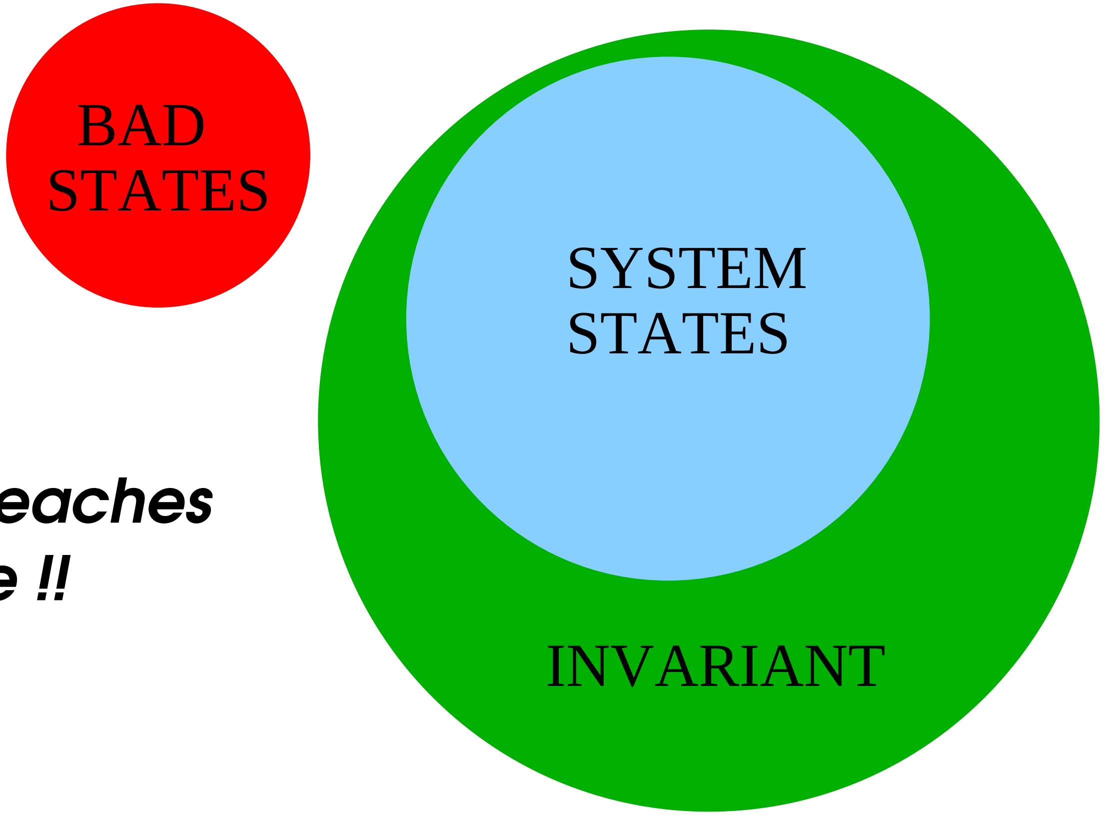
Why Care about Invariants ? (2)

- Systems often have an infinite number of states
  $\longrightarrow$ methods for finite-state systems *(e.g. model checking)* suffer from the **state explosion problem**
- Exact reachable set of a system is not computable generally
- **Solution:** overapproximate reachable states $\longrightarrow$

  INVARIANTS: properties that hold for all states

Introduction

Why Care about Invariants ? (3)



System never reaches a bad state !!

Introduction
Abstract Interpretation (1)

Abstract interpretation allows to compute invariants:

- intervals (Cousot & Cousot 1976, Harrison 1977)

$$x \in [0,1] \ \wedge \ y \in [0,\infty)$$

- congruences (Granger 1991)

$$x \equiv y \ \mathrm{mod}(2)$$

- linear inequalities (Cousot & Halbwachs 1978, Colón & Sankaranarayanan & Sipma 2003)

$$x + 2y - 3z \leq 3$$

- octagonal inequalities (Mine 2001)

$$x - y \leq 3$$

- octahedral inequalities (Clariso & Cortadella 2004)

$$x - y + z \leq 2$$

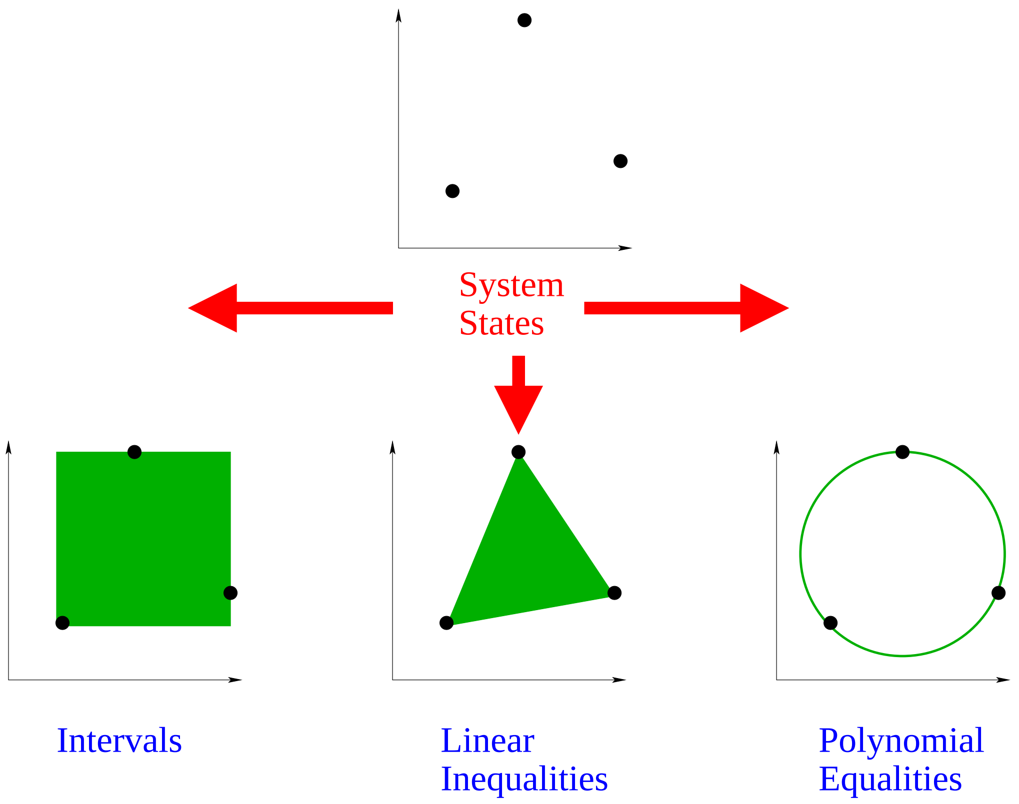
- ...
- **polynomial equalities** (Müller-Olm & Seidl 2004, Sankaranarayanan & Sipma& Manna 2004, Colón 2004, Rodríguez-Carbonell & Kapur 2004)

$$x = y^2$$

Introduction

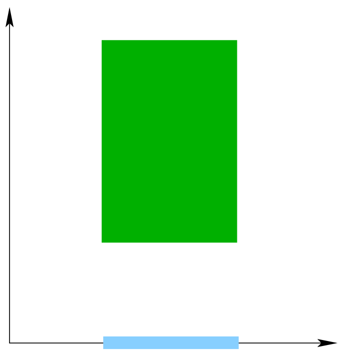
Abstract Interpretation (2)

Concrete variable values overapproximated by *abstract values*



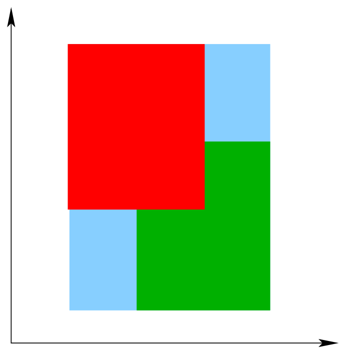
Introduction
Abstract Interpretation (3)

- Program semantics expressed in terms of abstract values
- Operations on states that must be abstracted:



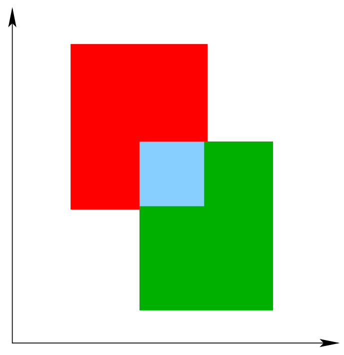
Projection

assignments



Union

merging in loops and conditionals



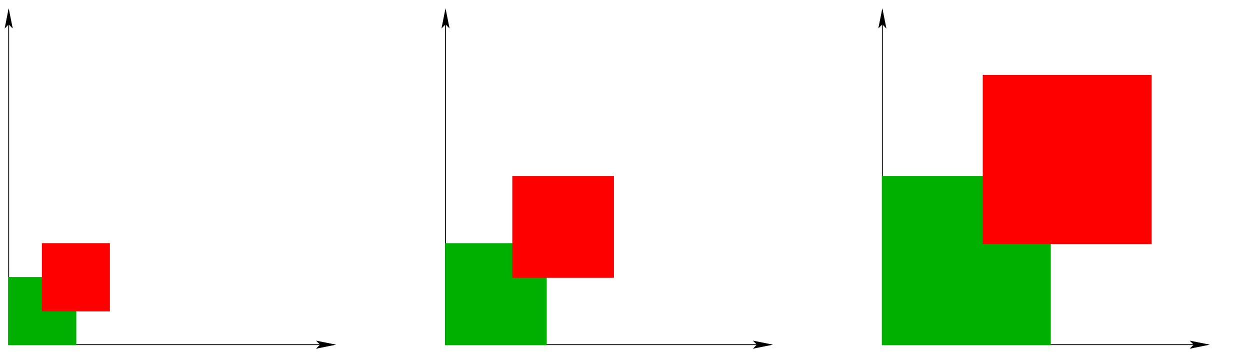
Intersection

guards in loops and conditionals

Introduction

Abstract Interpretation (4)

- Invariants are generated by symbolic execution of the program using the abstract semantics
- Termination is not guaranteeed in general:
  $\longrightarrow$ union in loops must be extrapolated



- **Widening operator** introduced to ensure termination

Related Work

Overview Polynomial Invariants

Work	Restrictions	Equality Conditions	Disequality Conditions	Complete
MOS, POPL'04	bounded degree	no	no	yes
SSM, POPL'04	prefixed form	yes	no	no
MOS, IPL'04	prefixed form	no	yes	yes
RCK, ISSAC'04	no restriction	no	no	yes
COL, SAS'04	bounded degree	yes	no	no
RCK, SAS'04	bounded degree	yes	yes	yes*

Overview of the Talk

1. **Overview of the Method**
2. **Ideals of Polynomials**
3. **Abstract Semantics**
4. **Widening Operator**
5. **Examples**
6. **Alternative Solution**
7. **Future Work & Conclusions**

Overview of the Method (1)

- Finds polynomial equality invariants
- States abstracted to ideal of polynomials evaluating to 0
- Programming language admits
  - Polynomial assignments: $variable := polynomial$
  - Polynomial equalities and disequalities in conditions:

$$polynomial = 0 \quad , \quad polynomial \neq 0$$

- Parametric widening ∇_d
- If conditions are ignored and assignments are linear, finds **all** polynomial invariants of degree $\leq d$

Overview of the Method (2)

- Our implementation has been successfully applied to a number of programs
- Ideals of polynomials represented by finite bases of generators: **Gröbner bases**
- There are several tools manipulating ideals, Gröbner bases
- Our implementation uses *Macaulay 2*

Overview of the Talk

1. **Overview of the Method**
2. **Ideals of Polynomials**
3. **Abstract Semantics**
4. **Widening Operator**
5. **Examples**
6. **Alternative Solution**
7. **Future Work & Conclusions**

Ideals of Polynomials
Preliminaries (1)

- Intuitively, an **ideal** is a set of polynomials and all their consequences
- An **ideal** is a set of polynomials I such that
  1. $0 \in I$
  2. If $p, q \in I$, then $p + q \in I$
  3. If $p \in I$ and q any polynomial, $pq \in I$

Ideals of Polynomials
Preliminaries (2)

- Example 1: polynomials evaluating to 0 on a set of points S

1. 0 evaluates to 0 everywhere

$$\forall \omega \in S, \quad 0(\omega) = 0$$

2. If p, q evaluate to 0 on S, then $p + q$ evaluates to 0 on S

$$\forall \omega \in S, \quad p(\omega) = q(\omega) = 0 \Longrightarrow p(\omega) + q(\omega) = 0$$

3. If p evaluates to 0 on S, then pq evaluates to 0 on S

$$\forall \omega \in S, \quad p(\omega) = 0 \Longrightarrow p(\omega) \cdot q(\omega) = 0$$

Ideals of Polynomials
Preliminaries (3)

- Example 2: multiples of a polynomial p, $\langle p \rangle$
  1. $0 = 0 \cdot p \in \langle p \rangle$
  2. $q_1 \cdot p + q_2 \cdot p = (q_1 + q_2)p \in \langle p \rangle$
  3. If q_2 is any polynomial, then $q_2 \cdot q_1 \cdot p \in \langle p \rangle$

- In general, ideal generated by $p_1,\ldots,p_k$:

$$\langle p_1, ..., p_k \rangle = \{\sum_{j=1}^{k} q_j \cdot p_j \text{ for arbitrary } q_j\}$$

- Hilbert's basis theorem: all ideals are finitely generated
  $\longrightarrow$ finite representation for ideals

Ideals of Polynomials
Operations with Ideals

- Several operations available. Given ideals I, J in the variables x_1, ..., x_n:
  - projection: $I \cap \mathbb{C}[x_1, ..., x_{i-1}, x_{i+1}, ..., x_n]$
  - addition: $I + J = \{p + q \mid p \in I, q \in J\}$
  - quotient: $I : J = \{p \mid \forall q \in J, p \cdot q \in I\}$
  - intersection: $I \cap J$
- All operations implemented using **Gröbner bases**
- These operations will be used when defining abstract semantics

Ideals of Polynomials
Ideals as Abstract Values (1)

- States abstracted to ideal of polynomials evaluating to 0

- Abstraction function I

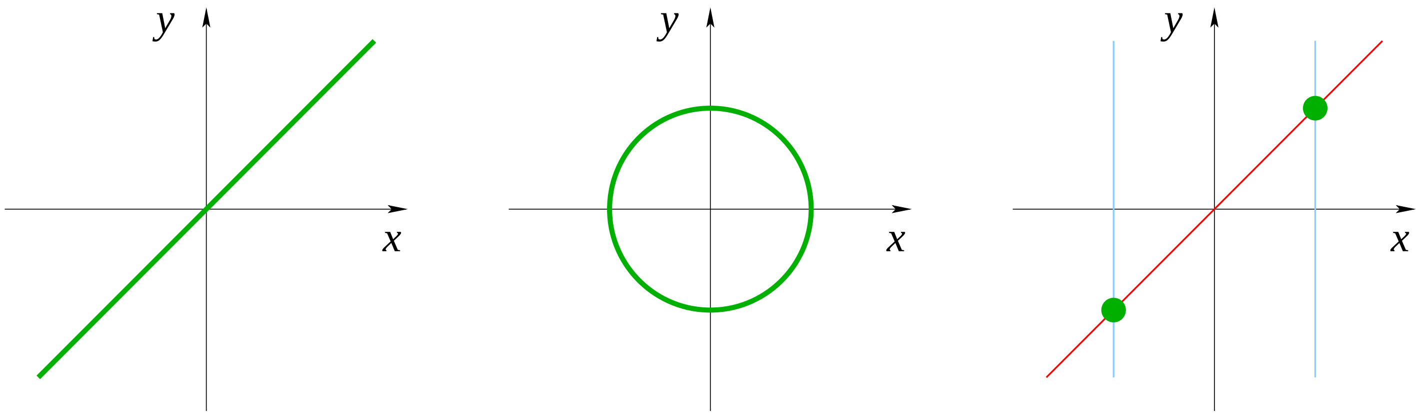
$$\begin{aligned} I : \{\text{sets of states}\} &\longrightarrow \{\text{ideals}\} \\ S &\longmapsto \{\textbf{polynomials evaluating} \\ &\qquad \text{to 0 on } S\} \end{aligned}$$

- Concretization function V

$$\begin{aligned} V : \{\text{ideals}\} &\longrightarrow \{\text{sets of states}\} \\ I &\longmapsto \{\textbf{zeroes of } I\} \end{aligned}$$

Ideals of Polynomials
Ideals as Abstract Values (2)

$$\langle p_1, \ldots, p_k \rangle \longleftrightarrow p_1 = 0 \wedge \cdots \wedge p_k = 0$$



$x = y$	$x^2 + y^2 = 1$	$x^2 = x \bigwedge x = y$
$\Updownarrow$	$\Updownarrow$	$\Updownarrow$
$\langle x - y \rangle$	$\langle x^2 + y^2 - 1 \rangle$	$\langle x^2 - x, x - y \rangle$

Overview of the Talk

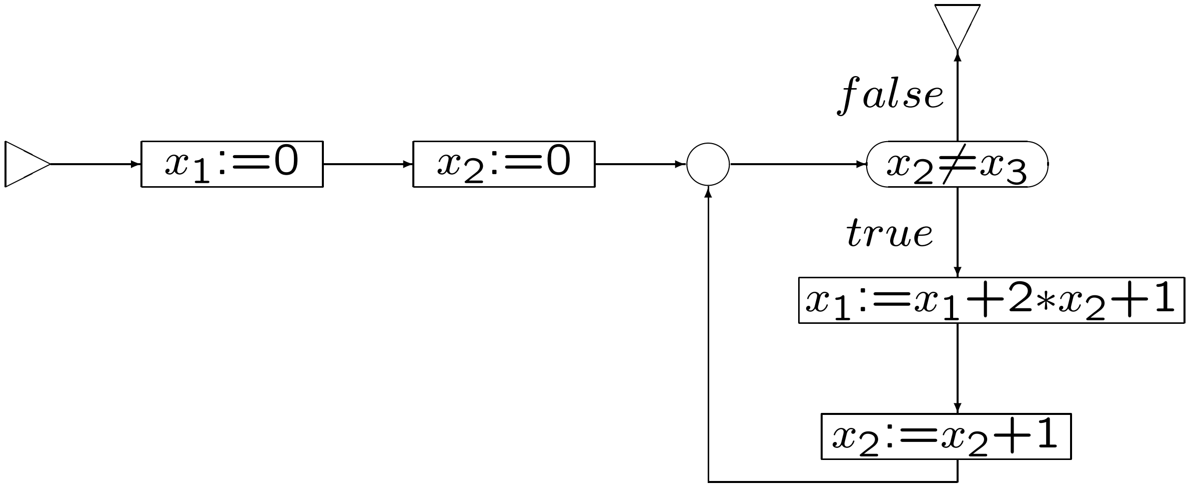
1. Overview of the Method
2. Ideals of Polynomials
3. **Abstract Semantics**
4. Widening Operator
5. Examples
6. Alternative Solution
7. Future Work & Conclusions

Abstract Semantics
Programming Model (1)

Programs $\equiv$ finite connected flowcharts

- Entry node
- Assignment nodes: polynomial assignments
- Test nodes: polynomial dis/equalities
- Simple/loop junction nodes
- Exit nodes

Abstract Semantics
Programming Model (2)



$x_1 := 0; x_2 := 0;$

while $x_2 \neq x_3$ **do**

$\quad x_1 := x_1 + 2 * x_2 + 1; x_2 := x_2 + 1;$

end while

Abstract Semantics
Assignments (1)

- Assignment node labelled with $x_i := f(x_1, \ldots, x_n)$
- Input ideal: $\langle p_1, \ldots, p_k \rangle$
- Output ideal:
  - Want to express in terms of ideals

    $\exists x_i' (x_i = f(x_i \leftarrow x_i') \wedge p_1(x_i \leftarrow x_i') = 0 \wedge \cdots \wedge p_k(x_i \leftarrow x_i') = 0)$

    where $x_i' \equiv$ previous value of x_i before the assignment
  - **Solution: projection**
    - eliminate x_i' from the ideal

$$\langle x_i - f(x_i \leftarrow x_i') \ , \ p_1(x_i \leftarrow x_i') \ , \ \ldots \ , \ p_k(x_i \leftarrow x_i') \rangle$$

Abstract Semantics
Assignments (2)

Example:

- Assignment $x := x + 1$
- Input ideal: $\langle x \rangle \longleftrightarrow x = 0$
- Output ideal:
  - Have to eliminate x' from the ideal

$$\langle x - x' - 1, x' \rangle$$

  - Polynomials of $\langle x - x' - 1, x' \rangle$ depending only on x:

$$\langle x - 1 \rangle \longleftrightarrow x = 1$$

Abstract Semantics
Tests: Polynomial Equalities

- Test node labelled with $q = 0$
- Input ideal: $\langle p_1, ..., p_k \rangle$
- Output ideal: (*true* path)
  - Want to express in terms of ideals

$$p_1 = 0 \wedge \cdots \wedge p_k = 0 \wedge q = 0$$

  - **Solution: addition**
    - Add q to list of generators of input ideal
    - Take maximal set of polynomials with same zeroes

$$\mathbf{I}(\mathbf{V}(p_1, ..., p_k, q))$$

Abstract Semantics

Tests: Polynomial Disequalities

- Test node labelled with $q \neq 0$
- Input ideal: $\langle p_1, ..., p_k \rangle$
- Output ideal: (*true* path)
  - Want to express in terms of ideals

$$p_1 = 0 \wedge \cdots \wedge p_k = 0 \wedge q \neq 0$$

  - **Solution: quotient**
    - quotient ideal $\langle p_1, ..., p_k \rangle : \langle q \rangle \equiv$ **maximal** ideal of polynomials evaluating to 0 on zeroes of $\langle p_1, ..., p_k \rangle$ $\setminus$ zeroes of $\langle q \rangle$

Abstract Semantics
Tests

Example:

- Test node labelled with $x = 0$
- Input ideal: $\langle xy \rangle \longleftrightarrow x = 0 \ \vee \ y = 0$
- Output ideal: (*true* path)

$$\mathbf{I}(\mathbf{V}(\langle xy, x \rangle)) = \langle x \rangle \longleftrightarrow x = 0$$

- Output ideal: (*false* path)

$$\langle xy \rangle : \langle x \rangle = \langle y \rangle \longleftrightarrow y = 0$$

Abstract Semantics
Simple Junction Nodes (1)

- Input ideals (one for each path):

  Path 1: $\langle p_{11}, ..., p_{1k_1} \rangle$

  ...

  Path l: $\langle p_{l1}, ..., p_{lk_l} \rangle$

- Output ideal:
  - Want to express in terms of ideals

    $$\bigvee_{i=1}^{l} \bigwedge_{j=1}^{k_i} p_{ij} = 0$$

  - **Solution: intersection**
    - Take *common* polynomials for all paths $\equiv$ Compute *intersection* of all input ideals

      $$\bigcap_{i=1}^{l} \langle p_{i1}, ..., p_{ik_i} \rangle$$

Abstract Semantics
Simple Junction Nodes (2)

Example:

- Input ideal 1st path: $\langle x \rangle \longleftrightarrow x = 0$
- Input ideal 2nd path: $\langle x - 1 \rangle \longleftrightarrow x = 1$
- Input ideal 3rd path: $\langle x - 2 \rangle \longleftrightarrow x = 2$
- Output ideal:

$$\langle x \rangle \cap \langle x-1 \rangle \cap \langle x-2 \rangle = \langle x(x-1)(x-2) \rangle$$

$$\longleftrightarrow x = 0 \ \vee \ x = 1 \ \vee \ x = 2$$

Degree increases !!

Abstract Semantics

Loop Junction Nodes (1)

- Input ideals: $J_1, \cdots, J_l$
- Output ideal:
  - As with simple junction nodes:

$$\bigcap_{i=1}^{l} J_i$$

  - **Problem:** Non-termination of symbolic execution !
  - **Solution: WIDENING** $\longrightarrow$ bounding degree

Abstract Semantics
Loop Junction Nodes (2)

Example:

$x := 0;$
while ? **do**
    $x := x + 1;$
end while

Generating loop invariant by symbolic execution:

- 1st iteration: $\langle x \rangle \longleftrightarrow x = 0$
- 2nd iteration: $\langle x(x-1) \rangle \longleftrightarrow x = 0 \lor x = 1$
- 3rd iteration: $\langle x(x-1)(x-2) \rangle \longleftrightarrow x = 0 \lor x = 1 \lor x = 2$
- ...

Unless we bound the degree, the procedure does not terminate

Overview of the Talk

1. Overview of the Method
2. Ideals of Polynomials
3. Abstract Semantics
4. **Widening Operator**
5. Examples
6. Alternative Solution
7. Future Work & Conclusions

Widening Operator
Definition

- Parametric widening $I \nabla_d J$
- Based on taking polynomials of $I \cap J$ of degree $\leq d$
- Definition uses **Gröbner bases**

$$I \nabla_d J := \mathbf{IV}(\{p \in GB(I \cap J) \mid \text{deg}(p) \leq d\})$$

- Termination guaranteed since $\{p \in I \mid \text{deg}(p) \leq d\}$ are vector spaces of finite dimension

Widening Operator
Loop Junction Nodes

- Input ideals: $J_1, \cdots, J_l$
- Previously computed output ideal: I
- Output ideal:

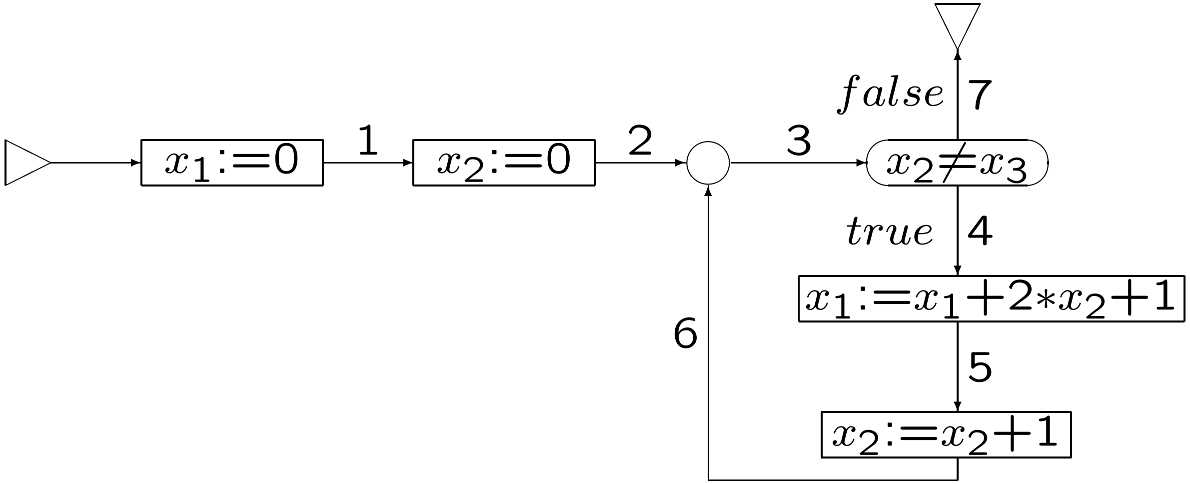
$$I \, \nabla_d \Big(\bigcap_{i=1}^{l} J_i \Big)$$

Widening Operator
A Completeness Result

- **THEOREM**. If conditions are ignored and assignments are linear, procedure computes **all** invariants of degree $\leq d$
- Key ideas of the proof:
  - $I \nabla_d J$ retains all polynomials of degree d of $I \cap J$
  - Graded term orderings used in Gröbner bases: glex, grevlex
- Conditions must be ignored: the set of all *linear invariants* in programs with *linear equality conditions* is not computable

Overview of the Talk

1. Overview of the Method
2. Ideals of Polynomials
3. Abstract Semantics
4. Widening Operator
5. Examples
6. Alternative Solution
7. Future Work & Conclusions



$$F_0(I) = \langle 0 \rangle$$

$$F_1(I) = (\langle x_1 \rangle + \langle I_0(x_1 \leftarrow x_1') \rangle) \cap \mathbb{C}[x_1, x_2, x_3]$$

$$F_2(I) = (\langle x_2 \rangle + \langle I_1(x_2 \leftarrow x_2') \rangle) \cap \mathbb{C}[x_1, x_2, x_3]$$

$$F_3(I) = I_3 \nabla_2 (I_2 \cap I_6)$$

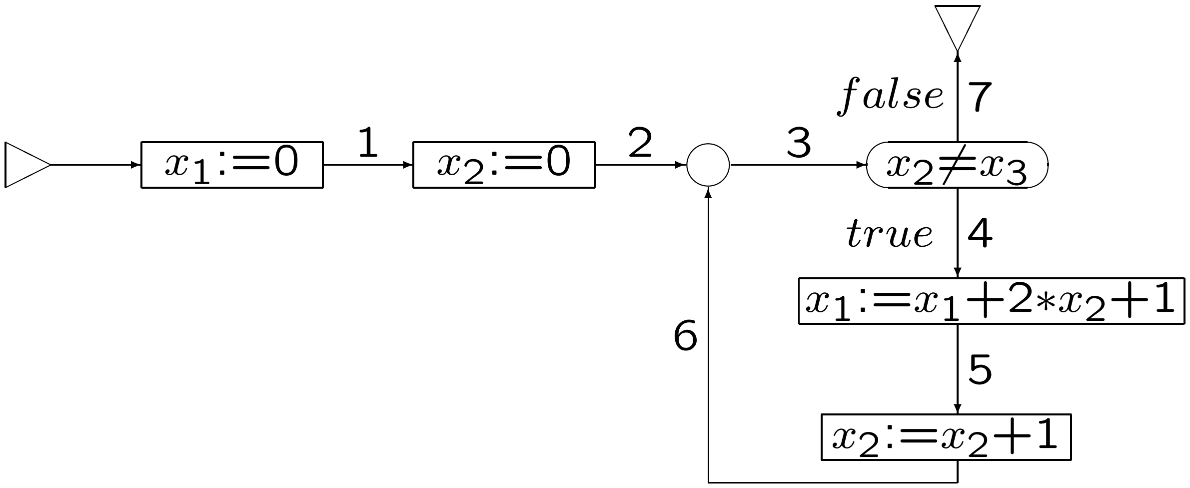
$$F_4(I) = \langle I_3 \rangle : \langle x_2 - x_3 \rangle$$

$$F_5(I) = I_4(x_1 \leftarrow x_1 - 2x_2 - 1)$$

$$F_6(I) = I_5(x_2 \leftarrow x_2 - 1)$$

$$F_7(I) = \mathbf{I}(\mathbf{V}(I_3 + \langle x_2 - x_3 \rangle))$$

ABSTRACT PROGRAM SEMANTICS



$$I_0^{(0)} = \langle 1 \rangle \qquad I_0^{(1)} = \langle 0 \rangle$$

$$I_1^{(0)} = \langle 1 \rangle \qquad I_1^{(1)} = (\langle x_1 \rangle + \langle 0 \rangle) \cap \mathbb{C}[x_1, x_2, x_3] = \langle x_1 \rangle$$

$$I_2^{(0)} = \langle 1 \rangle \qquad I_2^{(1)} = (\langle x_2 \rangle + \langle x_1 \rangle) \cap \mathbb{C}[x_1, x_2, x_3] = \langle x_1, x_2 \rangle$$

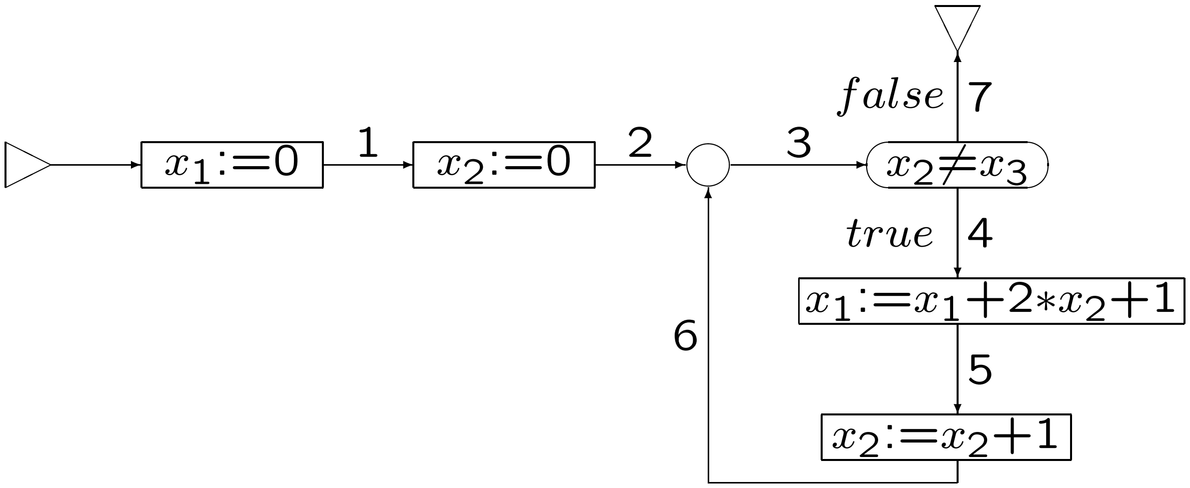
$$I_3^{(0)} = \langle 1 \rangle \qquad I_3^{(1)} = I_3^{(0)} \nabla_2 (I_2^{(1)} \cap I_6^{(0)}) = I_2^{(1)} = \langle x_1, x_2 \rangle$$

$$I_4^{(0)} = \langle 1 \rangle \qquad I_4^{(1)} = I_3^{(1)} : \langle x_2 - x_3 \rangle = \langle x_1, x_2 \rangle$$

$$I_5^{(0)} = \langle 1 \rangle \qquad I_5^{(1)} = I_4^{(1)}(x_1 \leftarrow x_1 - 2x_2 - 1) = \langle x_1 - 2x_2 - 1, x_2 \rangle$$

$$I_6^{(0)} = \langle 1 \rangle \qquad I_6^{(1)} = I_5^{(1)}(x_2 \leftarrow x_2 - 1) = \langle x_1 - 2x_2 + 1, x_2 - 1 \rangle$$

$$I_7^{(0)} = \langle 1 \rangle \qquad I_7^{(1)} = \mathbf{I}(\mathbf{V}(\langle x_2 - x_3 \rangle + I_3^{(1)})) = \langle x_1, x_2, x_3 \rangle$$



$$I_0^{(2)} = \langle 0 \rangle$$

$$I_1^{(2)} = \langle x_1 \rangle$$

$$I_2^{(2)} = \langle x_1, x_2 \rangle$$

$$I_3^{(2)} = \langle x_1 - x_2^2, x_2(x_2 - 1) \rangle$$

$$I_4^{(2)} = \langle x_1 - x_2^2, x_2(x_2 - 1) \rangle$$

$$I_5^{(2)} = \langle x_1 - x_2^2 - 2x_2 - 1, x_2(x_2 - 1) \rangle$$

$$I_6^{(2)} = \langle x_1 - x_2^2, (x_2 - 1)(x_2 - 2) \rangle$$

$$I_7^{(2)} = \langle x_1 - x_2^2, x_2(x_2 - 1), x_2 - x_3 \rangle$$

In 6 iterations we get the loop invariant

$$x_1 = x_2^2$$

Examples

Table

PROGRAM	COMPUTING	d	VARS	IF'S	LOOPS	LOOP DEPTH	TIME
cohencu	cube	3	5	0	1	1	2.45
dershowitz	real division	2	7	1	1	1	1.71
divbin	integer division	2	5	1	2	1	1.91
euclidex1	Bezout's coefs	2	10	0	2	2	7.15
euclidex2	Bezout's coefs	2	8	1	1	1	3.69
fermat	divisor	2	5	0	3	2	1.55
prod4br	product	3	6	3	1	1	8.49
freire1	integer sqrt	2	3	0	1	1	0.75
hard	integer division	2	6	1	2	1	2.19
lcm2	lcm	2	6	1	1	1	2.03
readers	simulation	2	6	3	1	1	4.15

Overview of the Talk

1. Overview of the Method
2. Ideals of Polynomials
3. Abstract Semantics
4. Widening Operator
5. Examples
6. **Alternative Solution**
7. Future Work & Conclusions

Alternative Solution (1)

- Alternative approach (Colón, SAS'04)
- Based on approximating ideals using degree bound d
- **Key observation:** given an ideal I, polynomials in I of degree $\leq d$ form a vector space of finite dimension
  $\longrightarrow$ use linear algebra instead of Gröbner bases
- A **pseudo-ideal** is a set P of polynomials of degree $\leq d$ such that
  1. $0 \in P$
  2. If $p, q \in P$, then $p + q \in P$
  3. If $p \in P$, q any polynomial and $\deg(pq) \leq d$, then $pq \in P$
- Pseudo-ideals are vector spaces of finite dimension

Alternative Solution (2)

- Operations on ideals approximated by operations on vector spaces
- **Advantages**
  - Easier to implement
  - Better complexity bounds
- **Disadvantages**
  - Loss of precision
  - Dimension of vector spaces increments exponentially with degree
- Combination of both techniques would be better ?

Overview of the Talk

1. **Overview of the Method**
2. **Ideals of Polynomials**
3. **Abstract Semantics**
4. **Widening Operator**
5. **Examples**
6. **Alternative Solution**
7. **Future Work & Conclusions**

Future Work

- Design widening operators not bounding degree
- Integrate with linear inequalities
- Study abstract domains for polynomial inequalities
- Apply to other classes of programs

Conclusions

- Method for generating polynomial equality invariants
- Based on abstract interpretation
- Programming language admits
  - Polynomial assignments
  - Polynomial dis/equalities in conditions
- If conditions are ignored and assignments are linear, finds **all** polynomial invariants of degree $\leq d$
- Implemented using Macaulay 2
- Successfully applied to many programs