

Precise Program Analysis through (Linear) Algebra

Markus Müller-Olm
FernUniversität Hagen
(on leave from Universität Dortmund)

Joint work with
Helmut Seidl (TU München)

CP+CV 2004, Barcelona, March 28, 2004

Overview

- Title, motivation and results
- Karr's algorithm
- Interprocedural analysis / linear algebra
- Intraprocedural analysis / algebra
- Conclusion

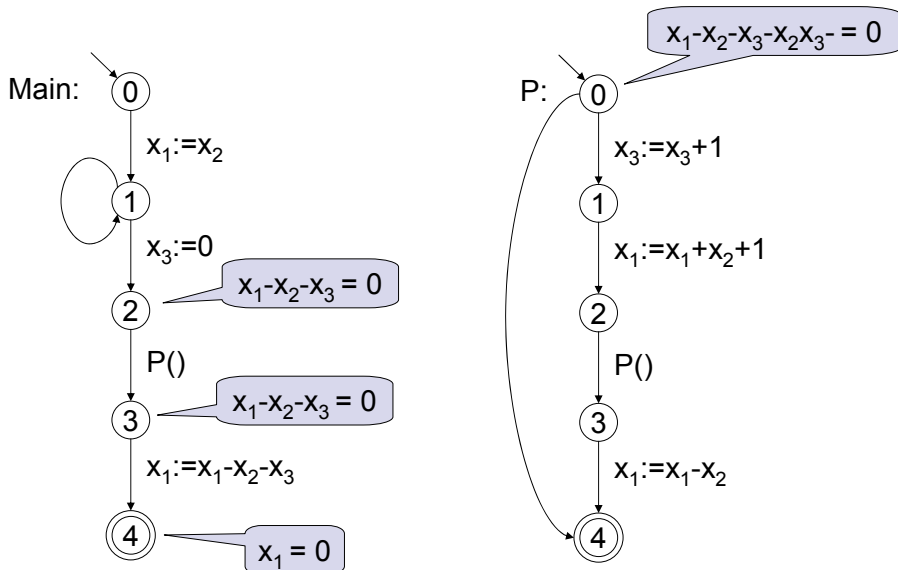


... through (Linear) Algebra

- Linear Algebra
 - vectors
 - vector spaces, sub-spaces, bases
 - linear maps, matrices
 - vector spaces of matrices
 - Gaussian elimination
- Algebra
 - rings
 - ideals
 - polynomial rings
 - ideals of polynomial rings
 - Buchberger's Algorithm



...(Interprocedural) Program Analysis...



Some Questions of Interest

- What is the value of variable x_1 at program exit ?
- Where is $x_1=x_2$?
- What is the relationship between x_1, x_2 , and x_3 at program point 3 ?
- ...

⇒ affine relations

$$a_0 + \sum a_i x_i = 0 \quad a_i \in \mathbb{F}$$

$$5x+7y-42=0$$

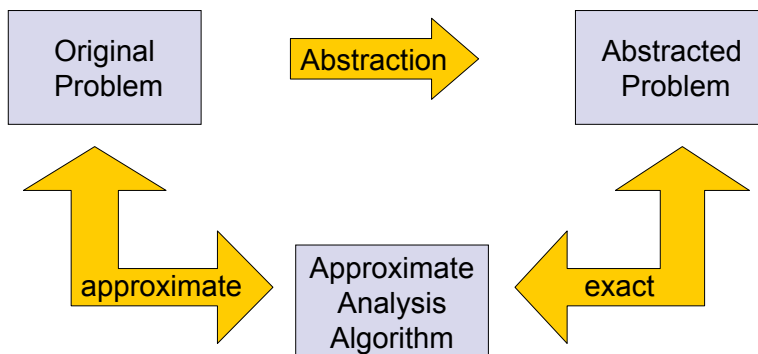
⇒ polynomial relations

$$p(x_1, \dots, x_k) = 0 \quad p \in \mathbb{F}[x_1, \dots, x_n]$$

$$5xy^2+7z^3-42=0$$



Exact Approximate Analysis



Abstractions of Interest

- Affine programs (first part):
 - affine assignments: $x_1 := x_1 - 2x_3 + 7$
 - unknown assignments: $x_i := ?$
→ abstract too complex statements!
 - non-deterministic instead of guarded branching
- Polynomial programs (second part):
 - polynomial assignments: $x_1 := x_1 x_2 - 5x_3$
 - negated polynomial guards: $\neg (x^2 - 3y = 0)$
 - the rest as for affine programs !



The Challenge (Precise...)

Given an affine program with (recursive) procedures, local variables, parameters, return values, ...

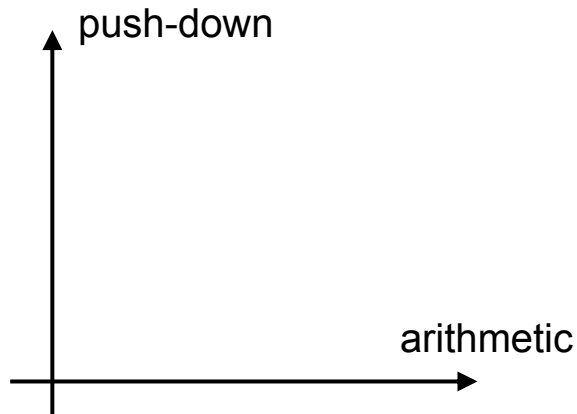
- determine all valid affine relations:
$$a_0 + \sum a_i x_i = 0 \quad a_i \in \mathbb{F}$$
- determine all valid polynomial relations:
$$p(x_1, \dots, x_k) = 0 \quad p \in \mathbb{F}[x_1, \dots, x_n]$$
- ... and all this in polynomial time ☺

Weaker goal:

- determine all polynomial relations of degree $\leq d$



Infinity Dimensions

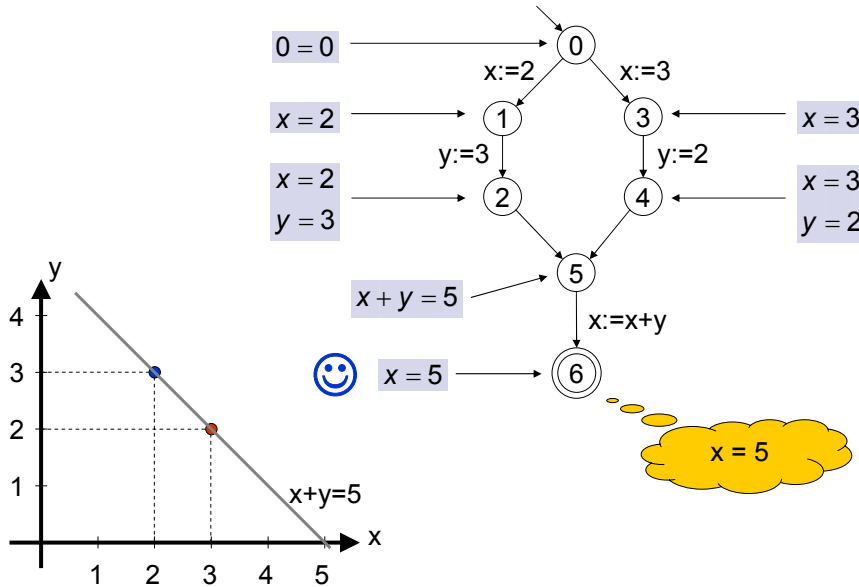


Applications

- Interprocedural analysis:
 - definite equalities: $x = y$
 - constant propagation: $x = 42$
 - discovery of symbolic constants: $x = 5yz+17$
 - complex common subexpressions: $xy+42 = y^2+5$
 - loop induction variables
- Program verification
 - strongest affine or polynomial assertions
(cf. Petri Net invariants)



Intraprocedural Algorithm of Karr

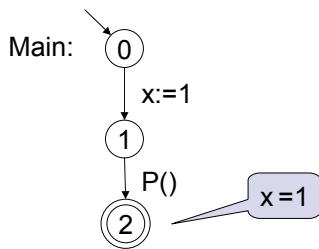


Use a Standard Approach for Interprocedural Generalization of Karr ?

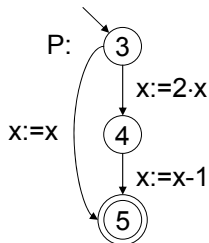
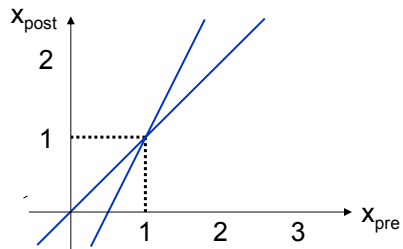
- **Functional approach** [Sharir/Pnueli, 1981], [Knoop/Steffen, 1992]
 - Idea: summarize each procedure by **function on data flow facts**
 - Problem: not applicable
- **Call-string approach** [Sharir/Pnueli, 1981]
 - Idea: take **just a finite piece of run-time stack** into account
 - Problem: not exact
- **Relational analysis** [Cousot², 1977]
 - Idea: summarize each procedure by **approximation of I/O relation**
 - Problem: not exact (next slide)



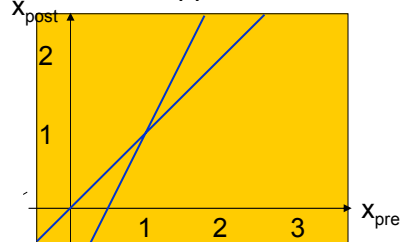
Relational Analysis is Not Strong Enough



True relational semantics of P:



Best affine approximation:



Overview

- Title and results
- Karr's algorithm
- **Interprocedural analysis / linear algebra**
- Intraprocedural analysis / algebra
- Conclusion





Concrete Semantics of an Execution Path

- Every execution path π induces an **affine transformation** of the program state:

$$\begin{aligned} & \llbracket x_1 := x_1 + x_2 + 1; x_3 := x_3 + 1 \rrbracket(v) \\ &= \llbracket x_3 := x_3 + 1 \rrbracket(\llbracket x_1 := x_1 + x_2 + 1 \rrbracket(v)) \\ &= \llbracket x_3 := x_3 + 1 \rrbracket \left(\begin{pmatrix} 1 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} v_1 \\ v_2 \\ v_3 \end{pmatrix} + \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} \right) \\ &= \left(\begin{pmatrix} 1 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} v_1 \\ v_2 \\ v_3 \end{pmatrix} + \begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix} \right) \end{aligned}$$

Affine Relations

- An affine relation can be represented by a vector:

$$5 + x_1 - 2x_2 - x_3 = 0 \quad \text{corresponds to} \quad \mathbf{a} = \begin{pmatrix} 5 \\ 1 \\ -2 \\ -1 \end{pmatrix}$$



Weakest Precondition of Affine Relations

- Every execution path π induces a **linear transformation** of affine post-conditions into their weakest pre-condition:

$$\begin{aligned} & \llbracket x_1 := x_1 + x_2 + 1; x_3 := x_3 + 1 \rrbracket^T (\mathbf{a}) \\ &= \llbracket x_1 := x_1 + x_2 + 1 \rrbracket^T \left(\llbracket x_3 := x_3 + 1 \rrbracket^T (\mathbf{a}) \right) \\ &= \llbracket x_1 := x_1 + x_2 + 1 \rrbracket^T \left(\begin{pmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{pmatrix} \right) \\ &= \begin{pmatrix} 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{pmatrix} \end{aligned}$$



WP of Affine Relations

$$\begin{pmatrix} 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 0 \\ 1 \\ -1 \\ -1 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \\ 0 \\ -1 \end{pmatrix}$$

Therefore:

$$\{x_1 - x_3 = 0\} \quad x_1 := x_1 + x_2 + 1; \quad x_3 := x_3 + 1 \quad \{x_1 - x_2 - x_3 = 0\}$$

weakest precondition!



Observation 1

- Only the zero relation is **valid** at program start:

$$\mathbf{0} : 0 + 0x_1 + \dots + 0x_k = 0$$

- Thus, relation $a_0 + a_1x_1 + \dots + a_kx_k = 0$ is **valid** at program point v iff

$$M a = \mathbf{0} \quad \text{for all } M \in \{ \llbracket \pi \rrbracket^T \mid \pi \text{ reaches } v \} .$$



Observation 2

- The following statements are equivalent for a :
 - $M a = \mathbf{0}$ for all $M \in \mathcal{R}$
 - $M a = \mathbf{0}$ for all $M \in \text{Span}(\mathcal{R})$
 - $M a = \mathbf{0}$ for all M in a basis of $\text{Span}(\mathcal{R})$



Observation 3

- The set of all affine relations valid at program point v equals the set of solutions of the linear equation system:

$$\mathbf{0} = M a, \quad M \in \mathcal{B}$$

where \mathcal{B} is a basis of

$$V = \text{Span}\{[\pi]^T \mid \pi \text{ reaches } v\} \quad (+)$$

\Rightarrow it suffices to compute a basis of V !



Observation 4

- The set of subspaces of $\mathbb{F}^{k \times k}$ is a complete lattice:
 - Ordering: $\sqsubseteq = \subseteq$
 - Least element: $\{0\}$
 - Least upper bound: $\mathcal{B}_1 \sqcup \mathcal{B}_2 = \text{Span}(\mathcal{B}_1 \cup \mathcal{B}_2)$
 - Height: k^2

→ abstract interpretation techniques apply !

$$\alpha(R) = \text{Span}\{ \llbracket \pi \rrbracket^T \mid \pi \in R \}$$

$$R(v) = \{ \pi \mid \pi \text{ reaches } v \}$$



Constraint System for Characterizing Execution Paths

Executions of base edges:

$$\llbracket x := t \rrbracket = \{ x_i := t \}$$

$$\llbracket x := ? \rrbracket = \{ x_i := d \mid d \in \mathbb{Q} \}$$

Same-level executions:

$$S(v) \supseteq \{ \varepsilon \}$$

v entry point

$$S(p) \supseteq S(v)$$

v return point of p

$$S(v) \supseteq S(u); \llbracket \text{lab}(u, v) \rrbracket$$

(u, v) base edge

$$S(v) \supseteq S(u); S(p)$$

(u, v) calls procedure p

Reaching executions:

$$R(v) \supseteq S(v)$$

v in Main

$$R(v) \supseteq R(p); S(v)$$

v in p

$$R(p) \supseteq R(u)$$

$(u, _)$ calls p

Abstract Interpretation (on Bases)

$$\llbracket x_j := a_0 + \sum a_i x_i \rrbracket^\# = \text{Span} \left\{ \left(\begin{array}{c|c|c} I & a_0 & 0 \\ \hline & \vdots & \\ 0 & a_k & I \end{array} \right) \right\}$$

$$\llbracket x_j := ? \rrbracket^\# = \text{Span} \left\{ \left(\begin{array}{c|c|c} I & 0 & 0 \\ \hline & \vdots & \\ 0 & 0 & I \end{array} \right), \left(\begin{array}{c|c|c} I & 1 & 0 \\ \hline & \vdots & \\ 0 & 0 & I \end{array} \right) \right\}$$

$$\text{Span}\{M_i \mid i \in I\}^\# \text{Span}\{N_j \mid j \in J\} = \text{Span}\{M_i N_j \mid i \in I, j \in J\}$$

$$\text{Span}\{M_i \mid i \in I\} \cup^\# \text{Span}\{N_j \mid j \in J\} = \text{Span}(\{M_i \mid i \in I\} \cup \{N_j \mid j \in J\})$$

Use **Gauss elimination** for simplifying sets of matrices



Theorem

In an affine program:

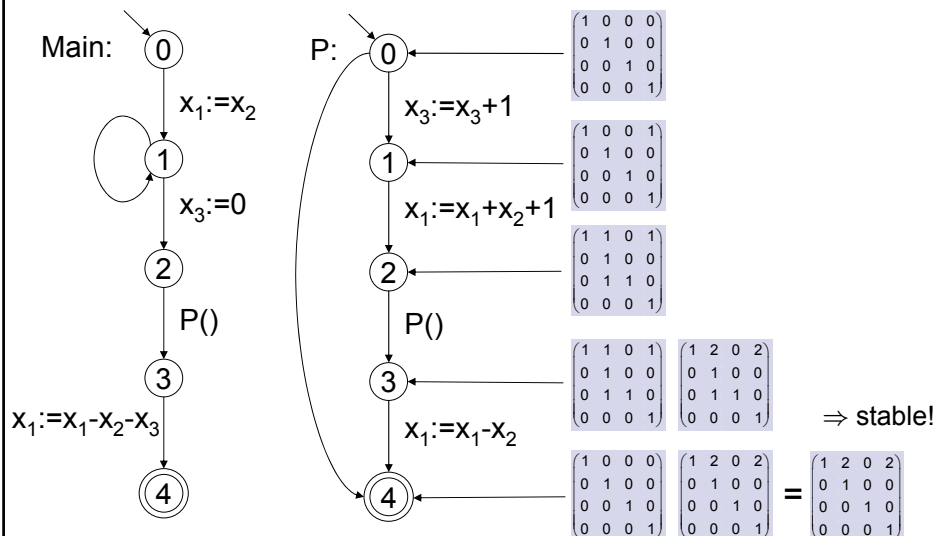
- The following vector spaces of matrices can be computed precisely:

$$\alpha(R(v)) = \text{Span} \{ \llbracket \pi \rrbracket^T \mid \pi \in R(v) \}$$
 for each prg. point v .
- The vector spaces

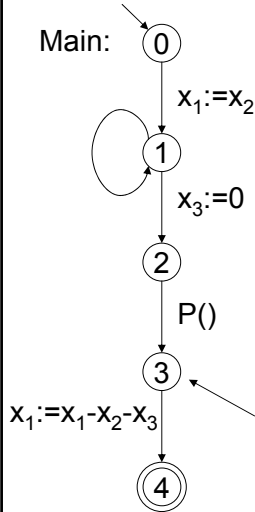
$$\{ a \in \mathbb{F}^{k+1} \mid \text{affine relation } a \text{ is valid at } v \}$$
 can be computed precisely for all prg. points v .
- The time complexity is **linear** in the program size and **polynomial** in the number of variables: $\mathcal{O}(n \cdot k^3)$
 (n size of the program, k number of variables)



An Example



An Example



$a_0 + a_1x_1 + a_2x_2 + a_3x_3 = 0$ is valid at 3

$$\Leftrightarrow \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{pmatrix} = 0 \text{ and } \begin{pmatrix} 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{pmatrix} = 0$$

$$\Leftrightarrow a_0 = 0 \wedge a_2 = a_3 = -a_1$$

\Rightarrow Just the affine relations of the form
 $a_1x_1 - a_1x_2 - a_1x_3 = 0 \quad (a_1 \in \mathbb{R})$
 are valid at 3



$$\text{Span} \left\{ \left(\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}, \begin{pmatrix} 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \right) \right\}$$

Extensions

- Local variables, value parameters, return values
- Computing **polynomial** relations of **degree $\leq d$**
- Affine pre-conditions



Overview

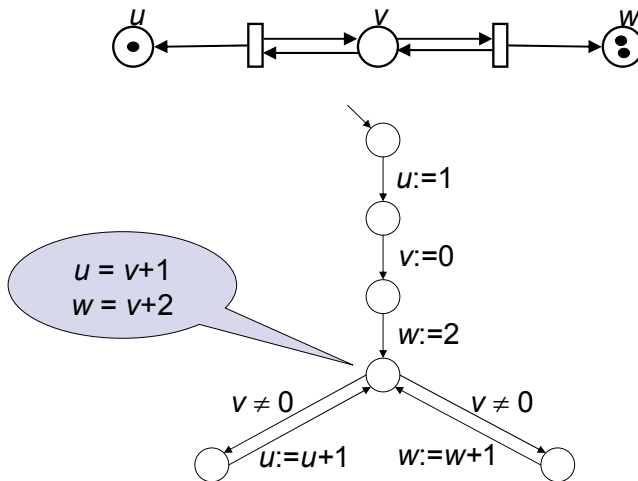
- Title and results
- Karr's algorithm
- Interprocedural analysis / linear algebra
- **Intraprocedural analysis / algebra**
- Conclusion



Precise Analysis through Algebra

- Algebra
 - Polynomial rings, ideals, Gröbner bases, ...
- Polynomial programs:
 - Polynomial assignments: $x := xy - 5z$
 - Negated polynomial guards: $\neg (xy - 3z = 0)$
 - The rest as for affine programs !
- Intraprocedural computation of „polynomial constants“ [SAS 2002]
- Intraprocedural derivation of **all** valid polynomial relations of degree $\leq d$ [MO/Seidl 2003]

Negated Polynomial Guards are Useful



Note: we need the power of polynomials in order to cope with the guards ! 😊

Representing other Models

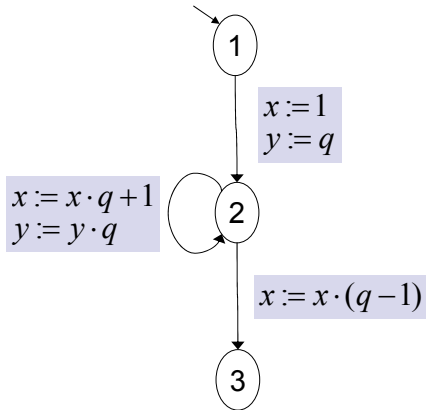
- Polynomial programs can represent:

- Petri Nets
- Vector Addition Systems (VAS)
- VAS with state
- ...

⇒ polynomial invariants for these models !



A Polynomial Program



After n iterations at 2:

$$x = \sum_{i=0}^n q^i = \frac{q^{n+1} - 1}{q - 1} \quad (\text{Horner's method})$$

$$y = q^{n+1}$$

$$\Rightarrow x \cdot (q - 1) = y - 1$$

$$\Rightarrow x \cdot q - x - y + 1 = 0$$

At 3:

$$x - y + 1 = 0$$



Idea

- Use ideals instead of vector spaces:
 - An ideal $I \subseteq \mathbb{F}[x_1, \dots, x_k]$ is a set of polynomials with:
 - 1) $q_1, q_2 \in I$ implies $q_1 + q_2 \in I$
 - 2) $q \in I$ implies $rq \in I$ for all $r \in \mathbb{F}[x_1, \dots, x_k]$
 - $B \subseteq \mathbb{F}[x_1, \dots, x_n]$ generates I iff

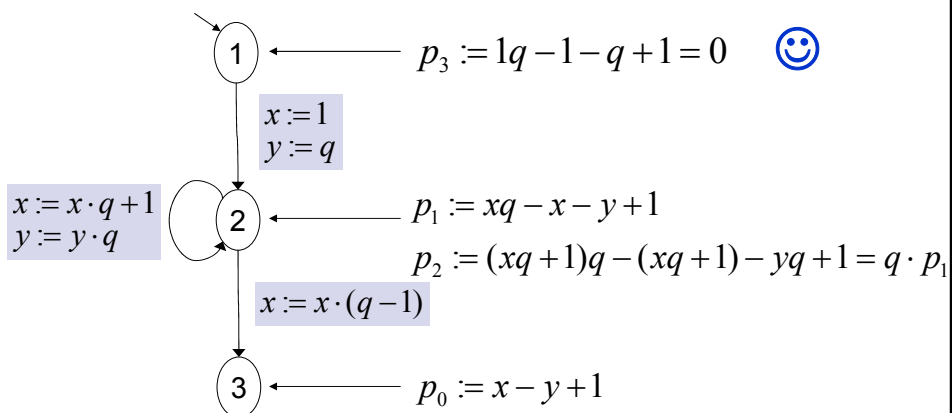
$$I = \langle B \rangle = \{r_1 b_1 + \dots + r_k b_k \mid r_1, \dots, r_k \in \mathbb{F}[x_1, \dots, x_n], b_1, \dots, b_k \in B\}$$
- Intuition:
 - Ideal $\langle p_1, \dots, p_n \rangle$ captures the 'essence' of polynomial constraint

$$\bigwedge_{i=1, \dots, n} p_i(x_1, \dots, x_n) = 0$$
- Problem:
 - no substitute for linear maps:
 - \Rightarrow just intra-procedural analysis

Observations

- **Hilbert's Basis Theorem:**
Every polynomial ideal is finitely generated.
 - ⇒ every ascending chain of polynomial ideals is ultimately stable.
 - ⇒ iterative least fixpoint computations stabilize.
- **Buchberger's Algorithm** allows us to check ideal membership, ideal inclusion, ideal equality.
 - ⇒ termination can be checked effectively
- Only the zero ideal $\langle 0 \rangle$ is valid at program start.
 - ⇒ Validity of weakest pre-condition checkable.

Checking Polynomial Relations



Question:

How to infer unknown identities ?

Idea:

Consider **generic** polynomial !

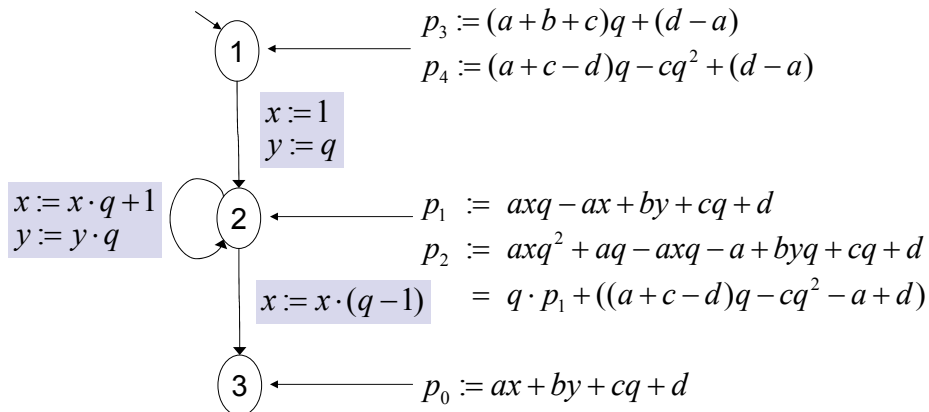
$$p_J = \sum_{(j_1, \dots, j_k) \in J} y_{j_1, \dots, j_k} x_1^{j_1} \cdot \dots \cdot x_k^{j_k} \quad (y_{j_1, \dots, j_k} \text{ fresh variables})$$

Lemma:

$$\text{Suppose } p = \sum_{(j_1, \dots, j_k) \in J} a_{j_1, \dots, j_k} x_1^{j_1} \cdot \dots \cdot x_k^{j_k}$$

$$\text{Then: } \llbracket \pi \rrbracket^t p = \left(\llbracket \pi \rrbracket^t p_J \right) [a/y]$$

Computing Polynomial Relations



$$\begin{array}{l} a + b + c = 0 \quad d - a = 0 \\ a + c - d = 0 \quad c = 0 \quad d - a = 0 \end{array} \Leftrightarrow a = d = -b \quad c = 0$$

\Rightarrow All identities of the form $ax - ay + a = 0$ are valid. 😊

Summary

- Precise program analysis through (linear) algebra
- Affine programs:
 - Interprocedural derivation of all valid polynomial relations of degree $\leq d$ (under affine pre-condition)
 - Summarize procedures by linear space of matrices
- Polynomial programs:
 - Intraprocedural derivation of all valid polynomial relations of degree $\leq d$

Future Challenges

- Affine & polynomial programs:
 - can we do without a degree bound for polynomial relations ?
- Affine programs:
 - guards ?
- Polynomial programs:
 - interprocedural analysis ?
 - complexity bound ?
- Other abstractions



References

- Seidl, MO: *Precise Interprocedural Analysis through Linear Algebra*. POPL 2004.
- Seidl, MO: *Polynomial constants are decidable*. SAS 2002, LNCS 2477, pages 4-19.
- MO: *Variations on Constants*. Habilitationsschrift, Uni Dortmund, 2002.
- Seidl, MO: *Computing Polynomial Program Invariants*. Submitted for publication (TR 310, FernUniversität Hagen).
- Rütting, MO: *On the Complexity of Constant Propagation*. ESOP 2001.

available from: <http://ls5-www.cs.uni-dortmund.de/~mmo> !

Questions?