Symbolic Computation and Theorem Proving in Program Analysis

Laura Kovács

Chalmers

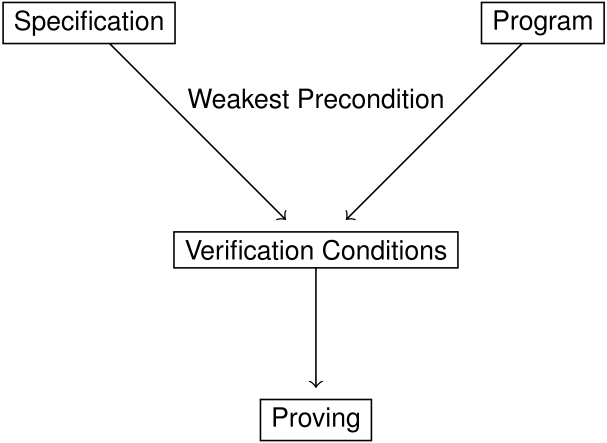
Outline

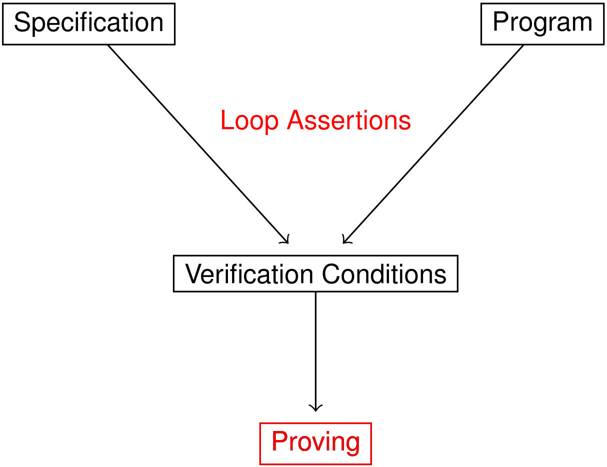
Part 1: Weakest Precondition for Program Analysis and Verification

Part 2: Polynomial Invariant Generation (TACAS'08, LPAR'10)

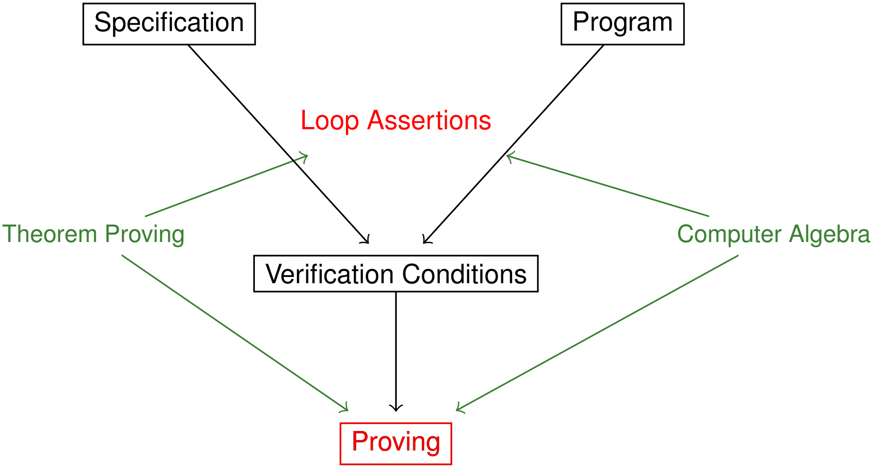
Part 3: Quantified Invariant Generation (FASE'09, MICAI'11)

Part 4: Invariants, Interpolants and Symbol Elimination (CADE'09, POPL'12, APLAS'12)


Specification
Program
Weakest Precondition
Verification Conditions
Proving



Specification
Program
Loop Assertions
Verification Conditions
Proving


Specification
Program
Loop Assertions
Theorem Proving
Computer Algebra
Verification Conditions
Proving

Assertion Synthesis — Example: Array Partition

Program

$a := 0;\ b := 0;\ c := 0;$
<u>while</u> $(a < N)$ <u>do</u>
  <u>if</u> $A[a] \geq 0$
    <u>then</u> $B[b] := A[a];\ b := b + 1$
    <u>else</u> $C[c] := A[a];\ c := c + 1;$
  $a := a + 1;$
<u>end while</u>

Loop Assertions

$$a = b + c$$

$$a \geq 0 \ \wedge \ b \geq 0 \ \wedge \ c \geq 0$$

$$a \leq N \quad \vee \quad N \leq 0$$

$$(\forall p)(p \geq b \implies B[p] = B_0[p])$$

$$(\forall p)(0 \leq p < b \implies B[p] \geq 0 \wedge (\exists i)(0 \leq i < a \ \wedge \ A[a] = B[p]))$$

Assertion Synthesis — Example: Array Partition

Program

$a := 0;\ b := 0;\ c := 0;$

<u>while</u> $(a < N)$ <u>do</u>

  <u>if</u> $A[a] \geq 0$

    <u>then</u> $B[b] := A[a];\ b := b + 1$

    <u>else</u> $C[c] := A[a];\ c := c + 1;$

  $a := a + 1;$

<u>end while</u>

Loop Assertions

Polynomial Equalities and Inequalities, Quantified FO properties

$$a = b + c$$

$$a \geq 0 \ \wedge\ b \geq 0 \ \wedge\ c \geq 0$$

$$a \leq N \quad \vee \quad N \leq 0$$

$$(\forall p)(p \geq b \implies B[p] = B_0[p])$$

$$(\forall p)(0 \leq p < b \implies B[p] \geq 0 \wedge (\exists i)(0 \leq i < a \ \wedge\ A[a] = B[p]))$$

Our Approach

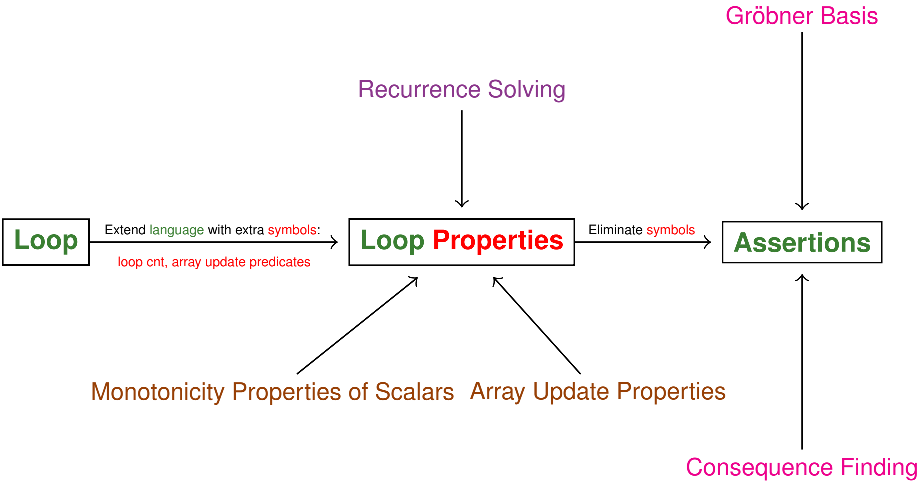
Loop

Assertions

Our Approach

Our Approach:
SYMBOL ELIMINATION

Our Approach:
SYMBOL ELIMINATION



Part 2: Polynomial Invariant Generation

Symbol Elimination by Gröbner Basis Computation

Outline

Overview of the Method

Algebraic Techniques for Invariant Generation

Polynomial Invariants for Loops with Assignments Only

Polynomial Invariants for Loops with Conditionals

Examples

Conclusions

Overview of Our Method - Division Example

$quo := 0;\ rem := x;$

<u>while</u> $y \leq rem$ <u>do</u> $\quad rem := rem - y;\ quo := quo + 1 \quad$ <u>end while</u>

Introduce LOOP COUNTER n $\quad (n \geq 0) \quad \rightarrow \quad$ nth iteration of the loop

$quo \quad \rightarrow \quad quo[n]$

$rem \quad \rightarrow \quad rem[n]$

System of Recurrences	System of Closed Forms	Loop body
$quo[n+1] = quo[n] + 1$	$\rightarrow CF_{quo}(n, quo[0])$ $\mathbf{quo[n] = quo[0] + n}$	$quo := quo + 1;$
$rem[n+1] = rem[n] - y$	$\rightarrow CF_{rem}(n, rem[0])$ $\mathbf{rem[n] = rem[0] - n * y}$	$rem := rem - y;$

Overview of Our Method - Division Example

$quo := 0;\ rem := x;$

<u>while</u> $y \leq rem$ <u>do</u> $rem := rem - y;\ quo := quo + 1$ <u>end while</u>

Introduce LOOP COUNTER n $(n \geq 0)$ $\rightarrow$ nth iteration of the loop

$quo \rightarrow quo[n]$

$rem \rightarrow rem[n]$

System of Recurrences	System of Closed Forms	Loop body
$quo[n+1] = quo[n] + 1$	$\rightarrow CF_{quo}(n, quo[0])$ $\mathbf{quo[n] = quo[0] + n}$	$quo := quo + 1;$
$rem[n+1] = rem[n] - y$	$\rightarrow CF_{rem}(n, rem[0])$ $\mathbf{rem[n] = rem[0] - n * y}$	$rem := rem - y;$

Overview of Our Method - Division Example

$quo := 0;\ rem := x;$

<u>while</u> $y \leq rem$ <u>do</u> $\quad rem := rem - y;\ quo := quo + 1 \quad$ <u>end while</u>

Introduce LOOP COUNTER n $\quad (n \geq 0) \quad \rightarrow \quad$ nth iteration of the loop

$$quo \rightarrow quo[n]$$
$$rem \rightarrow rem[n]$$

System of Recurrences | System of Closed Forms | Loop body

$quo[n+1] = quo[n] + 1 \quad \rightarrow CF_{quo}(n, quo[0]) \quad quo := quo + 1;$

$\mathbf{quo[n] = quo[0] + n}$

$rem[n+1] = rem[n] - y \quad \rightarrow CF_{rem}(n, rem[0]) \quad rem := rem - y;$

$\mathbf{rem[n] = rem[0] - n * y}$

Overview of Our Method - Division Example

$quo := 0;\; rem := x;$

$\underline{\text{while}}\; y \leq rem\; \underline{\text{do}}\quad rem := rem - y;\; quo := quo + 1\quad \underline{\text{end while}}$

Introduce LOOP COUNTER n $(n \geq 0)$ $\rightarrow$ nth iteration of the loop

$quo \rightarrow quo[n]$

$rem \rightarrow rem[n]$

$quo[n+1]$ depends algebraically upon $quo[n]$.

System of Recurrences	System of Closed Forms	Loop body
$quo[n+1] = quo[n] + 1$	$\rightarrow CF_{quo}(n, quo[0])$ $\mathbf{quo[n] = quo[0] + n}$	$quo := quo + 1;$
$rem[n+1] = rem[n] - y$	$\rightarrow CF_{rem}(n, rem[0])$ $\mathbf{rem[n] = rem[0] - n * y}$	$rem := rem - y;$

$quo[0]$ $quo[1]$

Overview of Our Method - Division Example

$quo := 0;\ rem := x;$

while $y \leq rem$ do $\quad rem := rem - y;\ quo := quo + 1$ end while

Introduce LOOP COUNTER n $(n \geq 0)$ $\rightarrow$ nth iteration of the loop

$quo \rightarrow quo[n]$

$rem \rightarrow rem[n]$

$quo[n+1]$ depends algebraically upon $quo[n]$.

System of Recurrences | System of Closed Forms | Loop body

$quo[n+1] = quo[n] + 1 \quad \rightarrow CF_{quo}(n, quo[0]) \quad quo := quo + 1;$

$\mathbf{quo[n] = quo[0] + n}$

$rem[n+1] = rem[n] - y \quad \rightarrow CF_{rem}(n, rem[0]) \quad rem := rem - y;$

$\mathbf{rem[n] = rem[0] - n * y}$

$quo[0] \quad quo[1] \quad quo[2] \quad \ldots$

Overview of Our Method - Division Example

$quo := 0;\ rem := x;$

$\underline{\text{while}}\ y \leq rem\ \underline{\text{do}} \quad rem := rem - y;\ quo := quo + 1 \quad \underline{\text{end while}}$

Introduce LOOP COUNTER n $(n \geq 0)$ $\rightarrow$ nth iteration of the loop

$quo \rightarrow quo[n]$

$rem \rightarrow rem[n]$

$rem[n+1]$ depends algebraically upon $rem[n+1]$.

System of Recurrences | System of Closed Forms | Loop body

$quo[n+1] = quo[n] + 1$ $\rightarrow CF_{quo}(n, quo[0])$ $quo := quo + 1;$

$\mathbf{quo[n] = quo[0] + n}$

$rem[n+1] = rem[n] - y$ $\rightarrow CF_{rem}(n, rem[0])$ $rem := rem - y;$

$\mathbf{rem[n] = rem[0] - n * y}$

$quo[0] \quad quo[1] \quad quo[2] \quad \ldots$

$rem[0] \quad rem[1]$

Overview of Our Method - Division Example

$quo := 0;\ rem := x;$

<u>while</u> $y \leq rem$ <u>do</u> $rem := rem - y;\ quo := quo + 1$ <u>end while</u>

Introduce LOOP COUNTER n $(n \geq 0)$ $\rightarrow$ nth iteration of the loop

$quo \rightarrow quo[n]$

$rem \rightarrow rem[n]$

$rem[n+1]$ depends algebraically upon $rem[n+1]$.

System of Recurrences | System of Closed Forms | Loop body

$quo[n+1] = quo[n] + 1$ $\rightarrow CF_{quo}(n, quo[0])$ $quo := quo + 1;$

$\mathbf{quo[n] = quo[0] + n}$

$rem[n+1] = rem[n] - y$ $\rightarrow CF_{rem}(n, rem[0])$ $rem := rem - y;$

$\mathbf{rem[n] = rem[0] - n * y}$

$quo[0]$	$quo[1]$	$quo[2]$	...
$rem[0]$	$rem[1]$	$rem[2]$	...

Overview of Our Method - Division Example

$quo := 0; \; rem := x;$

$\underline{\text{while}}\ y \leq rem\ \underline{\text{do}} \quad rem := rem - y; \; quo := quo + 1 \quad \underline{\text{end while}}$

Introduce LOOP COUNTER n $(n \geq 0)$ $\rightarrow$ nth iteration of the loop

$$quo \rightarrow quo[n]$$
$$rem \rightarrow rem[n]$$

System of Recurrences	System of Closed Forms	Loop body
$quo[n+1] = quo[n] + 1$	$\rightarrow CF_{quo}(n, quo[0])$ $quo[n] = quo[0] + n$	$quo := quo + 1;$
$rem[n+1] = rem[n] - y$	$\rightarrow CF_{rem}(n, rem[0])$ $rem[n] = rem[0] - n * y$	$rem := rem - y;$

$quo[0]$	$quo[1]$	$quo[2]$	...	quo[n-1]	quo[n]	...
$rem[0]$	$rem[1]$	$rem[2]$	...	rem[n-1]	rem[n]	...

Overview of Our Method - Division Example

$quo := 0;\ rem := x;$

<u>while</u> $y \leq rem$ <u>do</u> $rem := rem - y;\ quo := quo + 1$ <u>end while</u>

Introduce LOOP COUNTER n $(n \geq 0)$ $\rightarrow$ nth iteration of the loop

$quo \rightarrow quo[n]$

$rem \rightarrow rem[n]$

System of Recurrences	System of Closed Forms	Loop body
$quo[n+1] = quo[n] + 1$	$\rightarrow CF_{quo}(n, quo[0])$ $\mathbf{quo[n] = quo[0] + n}$	$quo := quo + 1;$
$rem[n+1] = rem[n] - y$	$\rightarrow CF_{rem}(n, rem[0])$ $\mathbf{rem[n] = rem[0] - n * y}$	$rem := rem - y;$

$quo[0]$	$quo[1]$	$quo[2]$	...	quo[n-1]	quo[n]	...
$rem[0]$	$rem[1]$	$rem[2]$	...	rem[n-1]	rem[n]	...

Overview of Our Method - Division Example

$quo := 0;\ rem := x;$

<u>while</u> $y \leq rem$ <u>do</u> $rem := rem - y;\ quo := quo + 1$ <u>end while</u>

Introduce LOOP COUNTER n $(n \geq 0)$ → nth iteration of the loop

$quo \rightarrow quo[n]$

$rem \rightarrow rem[n]$

System of Recurrences	System of Closed Forms	Loop body
$quo[n+1] = quo[n] + 1$	$\rightarrow CF_{quo}(n, quo[0])$ $quo[n] = quo[0] + n$	$quo := quo + 1;$
$rem[n+1] = rem[n] - y$	$\rightarrow CF_{rem}(n, rem[0])$ $rem[n] = rem[0] - n * y$	$rem := rem - y;$

$quo[0]$	$quo[1]$	$quo[2]$	...	quo[n-1]	quo[n]	...
$rem[0]$	$rem[1]$	$rem[2]$	...	rem[n-1]	rem[n]	...

Overview of Our Method - Division Example

$quo := 0;\ rem := x;$

$\underline{\text{while}}\ y \leq rem\ \underline{\text{do}}\quad rem := rem - y;\ quo := quo + 1\quad \underline{\text{end while}}$

Introduce LOOP COUNTER n $(n \geq 0)$ $\rightarrow$ nth iteration of the loop

$quo \rightarrow quo[n]$

$rem \rightarrow rem[n]$

System of Recurrences	System of Closed Forms	Loop body
$quo[n+1] = quo[n] + 1$	$\rightarrow CF_{quo}(n, quo[0])$ $\mathbf{quo[n] = quo[0] + n}$	$quo := quo + 1;$
$rem[n+1] = rem[n] - y$	$\rightarrow CF_{rem}(n, rem[0])$ $\mathbf{rem[n] = rem[0] - n * y}$	$rem := rem - y;$

$quo[0]$	$quo[1]$	$quo[2]$	...	quo[n-1]	quo[n]	...
$rem[0]$	$rem[1]$	$rem[2]$	...	rem[n-1]	rem[n]	...

Overview of Our Method - Division Example

$quo := 0;\ rem := x;$

$\underline{\text{while}}\ y \leq rem\ \underline{\text{do}}\quad rem := rem - y;\ quo := quo + 1\quad \underline{\text{end while}}$

1. Express state from $(n+1)^{th}$ iteration in terms of n^{th} iteration $\rightarrow$ recurrence relations of variables;

$n \geq 0$

$$\begin{cases} rem[n+1] & = & rem[n] - y \\ quo[n+1] & = & quo[n] + 1 \end{cases}$$

2. Solve recurrence relations $\rightarrow$ closed forms of variables: functions of iteration counter n $\uparrow$ methods from symbolic summation;

$$\begin{cases} rem[n] & = & rem[0] - n * y \\ quo[n] & = & quo[0] + n \end{cases}$$

3. Eliminate n and (optionally) initial value substitution;

$$rem = rem[0] - (quo - quo[0]) * y$$

4. Result: set of invariants

$rem = x - quo * y$ $\leftarrow$ Poly invariant

Polynomial ideal $\rightarrow$ Finite basis

$p_1 = 0, p_2 = 0 \rightarrow p_1 + p_2 = 0, \quad p_1 \cdot q = 0, \forall q.$

Gröbner basis

Overview of Our Method - Division Example

$quo := 0;\ rem := x;$

<u>while</u> $y \leq rem$ <u>do</u> $rem := rem - y;\ quo := quo + 1$ <u>end while</u>

1. Express state from $(n+1)^{th}$ iteration in terms of n^{th} iteration $\rightarrow$ recurrence relations of variables;

$n \geq 0$

$$\begin{cases} rem[n+1] = rem[n] - y \\ quo[n+1] = quo[n] + 1 \end{cases}$$

2. Solve recurrence relations $\rightarrow$ closed forms of variables: functions of iteration counter n $\uparrow$ methods from symbolic summation;

$$\begin{cases} rem[n] = rem[0] - n * y \\ quo[n] = quo[0] + n \end{cases}$$

3. Eliminate n and (optionally) initial value substitution;

$rem = rem[0] - (quo - quo[0]) * y$

4. Result: set of invariants

$rem = x - quo * y$ → Poly Invariant

Polynomial ideal → Finite basis

$p_1 = 0, p_2 = 0 \rightarrow p_1 + p_2 = 0;\quad p_1 \cdot q = 0, \forall q.$

Gröbner basis

Overview of Our Method - Division Example

$$quo := 0;\ rem := x;$$

$$\underline{\text{while}}\ y \le rem\ \underline{\text{do}}\quad rem := rem - y;\ quo := quo + 1\quad \underline{\text{end while}}$$

1. Express state from $(n+1)^{th}$ iteration in terms of n^{th} iteration $\rightarrow$ recurrence relations of variables;

$n \ge 0$

$$\begin{cases} rem[n+1] & = & rem[n] - y \\ quo[n+1] & = & quo[n] + 1 \end{cases}$$

2. Solve recurrence relations $\rightarrow$ closed forms of variables: functions of iteration counter n $\uparrow$ methods from symbolic summation;

$$\begin{cases} rem[n] & = & rem[0] - n * y \\ quo[n] & = & quo[0] + n \end{cases}$$

3. Eliminate n and (optionally) initial value substitution;

$$rem = rem[0] - (quo - quo[0]) * y$$

4. Result: set of invariants

$rem = x - quo * y \rightarrow$ Poly Invariant

Polynomial ideal $\rightarrow$ Finite basis

$p_1 = 0, p_2 = 0 \rightarrow p_1 + p_2 = 0, \quad p_1 \cdot q = 0, \forall q.$

Gröbner basis

Overview of Our Method - Division Example

$quo := 0;\ rem := x;$

$\underline{\text{while}}\ y \leq rem\ \underline{\text{do}}\quad rem := rem - y;\ quo := quo + 1\quad \underline{\text{end while}}$

1. Express state from $(n+1)^{th}$ iteration in terms of n^{th} iteration $\rightarrow$ recurrence relations of variables;

$n \geq 0$

$$\begin{cases} rem[n+1] & = & rem[n] - y \\ quo[n+1] & = & quo[n] + 1 \end{cases}$$

2. Solve recurrence relations $\rightarrow$ closed forms of variables: functions of iteration counter n
$\uparrow$ methods from symbolic summation;

$$\begin{cases} rem[n] & = & rem[0] - n * y \\ quo[n] & = & quo[0] + n \end{cases}$$

3. Eliminate n and (optionally) initial value substitution;

$$rem = rem[0] - (quo - quo[0]) * y$$

4. Result: set of invariants

$$rem = x - quo * y \quad \rightarrow \text{Poly Invariant}$$

Polynomial ideal $\rightarrow$ Finite basis

$p_1 = 0, p_2 = 0 \rightarrow p_1 + p_2 = 0, \quad p_1 \cdot q = 0, \forall q.$

Gröbner basis

Overview of Our Method - Division Example

$quo := 0;\ rem := x;$

while $y \leq rem$ do $\quad rem := rem - y;\ quo := quo + 1 \quad$ end while

1. Express state from $(n+1)^{th}$ iteration in terms of n^{th} iteration → recurrence relations of variables;

$n \geq 0$

$$\begin{cases} rem[n+1] & = & rem[n] - y \\ quo[n+1] & = & quo[n] + 1 \end{cases}$$

2. Solve recurrence relations → closed forms of variables: functions of iteration counter n
↑ methods from symbolic summation;

$$\begin{cases} rem[n] & = & rem[0] - n * y \\ quo[n] & = & quo[0] + n \end{cases}$$

3. Eliminate n and (optionally) initial value substitution;

$$rem = rem[0] - (quo - quo[0]) * y$$

4. Result: set of invariants

$rem = x - quo * y$ → Poly Invariant

Polynomial ideal → Finite basis

$p_1 = 0, p_2 = 0 \rightarrow p_1 + p_2 = 0, \quad p_1 \cdot q = 0, \forall q.$

Gröbner basis

Overview of Our Method - Division Example

$quo := 0;\ rem := x;$

$\underline{\text{while}}\ y \leq rem\ \underline{\text{do}}\quad rem := rem - y;\ quo := quo + 1\quad \underline{\text{end while}}$

1. Express state from $(n+1)^{th}$ iteration in terms of n^{th} iteration $\rightarrow$ recurrence relations of variables;

   $n \geq 0$
   $$\begin{cases} rem[n+1] = rem[n] - y \\ quo[n+1] = quo[n] + 1 \end{cases}$$

2. Solve recurrence relations $\rightarrow$ closed forms of variables: functions of iteration counter n
   $\uparrow$ methods from symbolic summation;

   $$\begin{cases} rem[n] = rem[0] - n * y \\ quo[n] = quo[0] + n \end{cases}$$

3. Eliminate n and (optionally) initial value substitution;

   $$rem = rem[0] - (quo - quo[0]) * y$$

4. Result: set of invariants

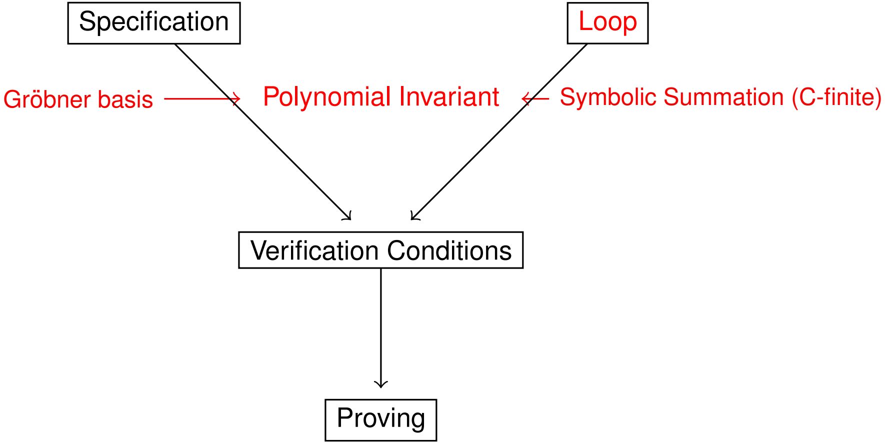
   $$rem = x - quo * y \quad \rightarrow \text{Poly Invariant}$$

   Polynomial ideal $\rightarrow$ Finite basis

   $p_1 = 0, p_2 = 0 \rightarrow p_1 + p_2 = 0, \quad p_1 \cdot q = 0, \forall q.$

   Gröbner basis

Overview of Our Method



Overview of the Method

$$x := 1;\ y := 0;$$

$$\underline{\text{while}}[\ldots,\ x := 2 * x;\ y := \frac{1}{2} * y + 1]$$

1. Express state from $(n+1)^{th}$ iteration in terms of n^{th} iteration → recurrence relations of variables;

   $n \geq 0,$

   $$\begin{cases} x[n+1] &= 2 * x[n] \\ y[n+1] &= \frac{1}{2} * y[n] + 1 \end{cases}$$

2. Solve recurrence relations → closed forms of variables: functions of iteration counter n ↑ methods from symbolic summation;

   $$\begin{cases} x[n] &= 2^n * x[0] \\ y[n] &= \frac{1}{2^n} * y[0] - \frac{2}{2^n} + 2 \end{cases}$$

3. Identify polynomial/algebraic dependencies among exponentials in n;

   $$\begin{cases} x &= a * x[0] \\ y &= b * y[0] - 2 * b + 2 \\ 0 &= a * b - 1 \end{cases}$$

4. Eliminate n and variables standing for algebraically related exponentials in n → elimination by Gröbner bases;

   $$x * y - 2 * x + 2 = 0$$

5. Result: Polynomial ideal → Gröbner basis

   $$p_1 = 0,\ p_2 = 0 \rightarrow p_1 + p_2 = 0,\ p_1 * q = 0,\ \forall q$$

Overview of the Method

$$x := 1;\; y := 0;$$

$$\underline{\text{while}}[\ldots,\; x := 2 * x;\; y := \frac{1}{2} * y + 1]$$

$n \geq 0$, $a = 2^n$, $b = 2^{-n}$

1. Express state from $(n+1)^{th}$ iteration in terms of n^{th} iteration $\rightarrow$ recurrence relations of variables;

$$\begin{cases} x[n+1] & = & 2 * x[n] \\ y[n+1] & = & \frac{1}{2} * y[n] + 1 \end{cases}$$

2. Solve recurrence relations $\rightarrow$ closed forms of variables: functions of iteration counter n
   $\uparrow$ methods from symbolic summation;

$$\begin{cases} x[n] & = & 2^n * x[0] \\ y[n] & = & \frac{1}{2^n} * y[0] - \frac{2}{2^n} + 2 \end{cases}$$

3. Identify polynomial/algebraic dependencies among exponentials in n;

$$\begin{cases} x & = & a * x[0] \\ y & = & b * y[0] - 2 * b + 2 \\ 0 & = & a * b - 1 \end{cases}$$

4. Eliminate n and variables standing for algebraically related exponentials in n $\rightarrow$ elimination by Gröbner bases;

$$x * y - 2 * x + 2 = 0$$

5. Result: Polynomial ideal $\rightarrow$ Gröbner basis

$p_1 = 0, p_2 = 0 \rightarrow p_1 + p_2 = 0;\quad p_1 \cdot q = 0, \forall q.$

Overview of the Method

$$x := 1;\ y := 0;$$

$$\underline{\text{while}}[\ldots,\ x := 2 * x;\ y := \frac{1}{2} * y + 1]$$

1. Express state from $(n+1)^{th}$ iteration in terms of n^{th} iteration → recurrence relations of variables;

   $n \geq 0$, $a = 2^n$, $b = 2^{-n}$

   $$\begin{cases} x[n+1] & = & 2 * x[n] \\ y[n+1] & = & \frac{1}{2} * y[n] + 1 \end{cases}$$

2. Solve recurrence relations → closed forms of variables: functions of iteration counter n ↑ methods from symbolic summation;

   $$\begin{cases} x[n] & = & 2^n * x[0] \\ y[n] & = & \frac{1}{2^n} * y[0] - \frac{2}{2^n} + 2 \end{cases}$$

3. Identify polynomial/algebraic dependencies among exponentials in n;

   $$\begin{cases} x & = & a * x[0] \\ y & = & b * y[0] - 2 * b + 2 \\ 0 & = & a * b - 1 \end{cases}$$

4. Eliminate n and variables standing for algebraically related exponentials in n → elimination by Gröbner bases;

   $$x * y - 2 * x + 2 = 0$$

5. Result: Polynomial ideal→ Gröbner basis

   $p_1 = 0, p_2 = 0 \rightarrow p_1 + p_2 = 0; \quad p_1 \cdot q = 0, \forall q.$

Overview of the Method

$$x := 1;\ y := 0;$$

$$\underline{\text{while}}[\ldots,\ x := 2 * x;\ y := \frac{1}{2} * y + 1]$$

1. Express state from $(n+1)^{th}$ iteration in terms of n^{th} iteration $\rightarrow$ recurrence relations of variables;

$n \geq 0,\ a = 2^n,\ b = 2^{-n}$

$$\begin{cases} x[n+1] &= 2 * x[n] \\ y[n+1] &= \frac{1}{2} * y[n] + 1 \end{cases}$$

2. Solve recurrence relations $\rightarrow$ closed forms of variables: functions of iteration counter n $\uparrow$ methods from symbolic summation;

$$\begin{cases} x[n] &= 2^n * x[0] \\ y[n] &= \frac{1}{2^n} * y[0] - \frac{2}{2^n} + 2 \end{cases}$$

3. Identify polynomial/algebraic dependencies among exponentials in n;

$$\begin{cases} x &= a * x[0] \\ y &= b * y[0] - 2 * b + 2 \\ 0 &= a * b - 1 \end{cases}$$

4. Eliminate n and variables standing for algebraically related exponentials in n $\rightarrow$ elimination by Gröbner bases;

$$x * y - 2 * x + 2 = 0$$

5. Result: Polynomial ideal $\rightarrow$ Gröbner basis

$$p_1 = 0, p_2 = 0 \rightarrow p_1 + p_2 = 0, \quad p_1 \cdot q = 0, \forall q.$$

Overview of the Method

$$x := 1;\; y := 0;$$

$$\underline{\text{while}}[\ldots,\; x := 2 * x;\; y := \frac{1}{2} * y + 1]$$

1. Express state from $(n+1)^{th}$ iteration in terms of n^{th} iteration $\rightarrow$ recurrence relations of variables;

   $$\begin{cases} x[n+1] & = & 2 * x[n] \\ y[n+1] & = & \frac{1}{2} * y[n] + 1 \end{cases}$$

2. Solve recurrence relations $\rightarrow$ closed forms of variables: functions of iteration counter n $\uparrow$ methods from symbolic summation;

   $$\begin{cases} x[n] & = & 2^n * x[0] \\ y[n] & = & \frac{1}{2^n} * y[0] - \frac{2}{2^n} + 2 \end{cases}$$

   $n \geq 0,\; a = 2^n,\; b = 2^{-n}$

3. Identify polynomial/algebraic dependencies among exponentials in n;

   $$\begin{cases} x & = & a * x[0] \\ y & = & b * y[0] - 2 * b + 2 \\ 0 & = & a * b - 1 \end{cases}$$

4. Eliminate n and variables standing for algebraically related exponentials in n $\rightarrow$ elimination by Gröbner bases;

   $$x * y - 2 * x + 2 = 0$$

5. Result: Polynomial ideal $\rightarrow$ Gröbner basis

   $p_1 = 0, p_2 = 0 \rightarrow p_1 + p_2 = 0, \quad p_1 \cdot q = 0, \forall q.$

Overview of the Method

$$x := 1;\ y := 0;$$

$$\underline{\text{while}}[\ldots,\ x := 2 * x;\ y := \frac{1}{2} * y + 1]$$

$n \geq 0,\ a = 2^n,\ b = 2^{-n}$

1. Express state from $(n+1)^{th}$ iteration in terms of n^{th} iteration $\rightarrow$ recurrence relations of variables;

$$\begin{cases} x[n+1] & = & 2 * x[n] \\ y[n+1] & = & \frac{1}{2} * y[n] + 1 \end{cases}$$

2. Solve recurrence relations $\rightarrow$ closed forms of variables: functions of iteration counter n $\uparrow$ methods from symbolic summation;

$$\begin{cases} x[n] & = & 2^n * x[0] \\ y[n] & = & \frac{1}{2^n} * y[0] - \frac{2}{2^n} + 2 \end{cases}$$

3. Identify polynomial/algebraic dependencies among exponentials in n;

$$\begin{cases} x & = & a * x[0] \\ y & = & b * y[0] - 2 * b + 2 \\ 0 & = & a * b - 1 \end{cases}$$

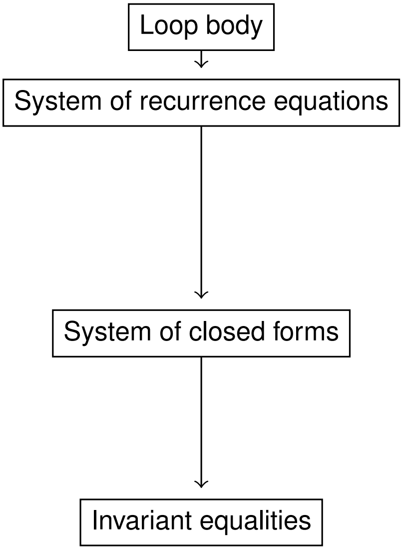
4. Eliminate n and variables standing for algebraically related exponentials in n $\rightarrow$ elimination by Gröbner bases;

$$x * y - 2 * x + 2 = 0$$

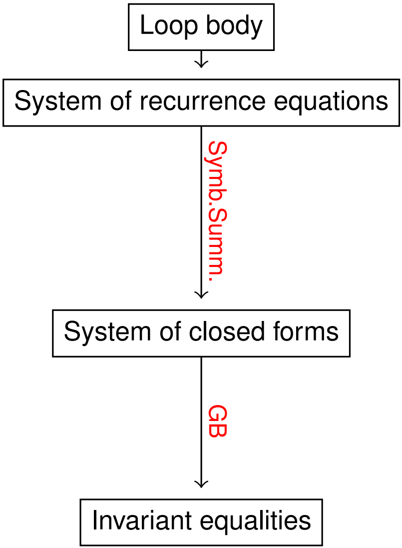
5. Result: Polynomial ideal$\rightarrow$ Gröbner basis

$$p_1 = 0, p_2 = 0 \rightarrow p_1 + p_2 = 0, \quad p_1 \cdot q = 0, \forall q.$$

Overview of Our Method



Overview of Our Method



Overview of the Method - Further Considerations

- Loops with assignments and with conditional branches.

  Structural constraints on assignments with polynomial rhs.

  ← Summation algorithms (Gosper's, C-finite)

  Tests are ignored → (basic) non-deterministic programs;

  while *cond* do S end while → while … do S do ⟶ S^*

- Automated Loop Invariant Generation by Algebraic Techniques Over the Rationals:

  polynomial invariant generation by symbolic summation and polynomial algebra algorithms ← P-solvable loops:

  $$\{p(X) = 0 \wedge X = X_0\} \quad S^* \quad \{p(X) = 0\}$$

Overview of the Method - Further Considerations

- Loops with assignments and with conditional branches.

  Structural constraints on assignments with polynomial rhs.

  $\leftarrow$ Summation algorithms (Gosper's, C-finite)

  Tests are ignored $\rightarrow$ (basic) non-deterministic programs;

  while *cond* do *S* end while $\rightarrow$ while ... do *S* do $\longrightarrow$ S^*

- Automated Loop Invariant Generation by Algebraic Techniques Over the Rationals:

  polynomial invariant generation by symbolic summation and polynomial algebra algorithms $\leftarrow$ P-solvable loops;

  $$\{p(X) = 0 \ \wedge \ X = X_0\} \quad S^* \quad \{p(X) = 0\}$$

Overview of the Method - Further Considerations

- Loops with assignments and with conditional branches.

  Structural constraints on assignments with polynomial rhs.

  $\leftarrow$ Summation algorithms (Gosper's, C-finite)

  Tests are ignored $\rightarrow$ (basic) non-deterministic programs;

  while *cond* do S end while $\rightarrow$ while ... do S do $\longrightarrow$ S^*

- Automated Loop Invariant Generation by Algebraic Techniques Over the Rationals:

  polynomial invariant generation by symbolic summation and polynomial algebra algorithms $\leftarrow$ P-solvable loops;

  $$\{p(X) = 0 \land X = X_0\} \quad S^* \quad \{p(X) = 0\}$$

Overview of the Method - Further Considerations

- Loops with assignments and with conditional branches.

  Structural constraints on assignments with polynomial rhs.

  $\leftarrow$ Summation algorithms (Gosper's, C-finite)

  Tests are ignored $\rightarrow$ (basic) non-deterministic programs;

  $\underline{\text{while}}$ *cond* $\underline{\text{do}}$ S $\underline{\text{end while}}$ $\rightarrow$ $\underline{\text{while}}$ $\ldots$ $\underline{\text{do}}$ S $\underline{\text{do}}$ $\longrightarrow$ S^*

- Automated Loop Invariant Generation by Algebraic Techniques Over the Rationals:

  polynomial invariant generation by symbolic summation and polynomial algebra algorithms $\leftarrow$ P-solvable loops;

$$\{p(X) = 0 \ \wedge \ X = X_0\} \quad S^* \quad \{p(X) = 0\}$$

Overview of the Method - Further Considerations

- Loops with assignments and with conditional branches.

  Structural constraints on assignments with polynomial rhs.

  $\leftarrow$ Summation algorithms (Gosper's, C-finite)

  Tests are ignored $\rightarrow$ (basic) non-deterministic programs;

  $$\underline{\text{while}}\ \mathit{cond}\ \underline{\text{do}}\ S\ \underline{\text{end while}} \ \rightarrow\ \underline{\text{while}}\ \ldots\ \underline{\text{do}}\ S\ \underline{\text{do}} \ \longrightarrow\ S^*$$

- **A**utomated **L**oop **I**nvariant **G**eneration by **A**lgebraic **T**echniques **O**ver the **R**ationals:

  polynomial invariant generation by symbolic summation and polynomial algebra algorithms $\leftarrow$ P-solvable loops;

  $$\{p(X) = 0 \ \wedge\ X = X_0\} \quad S^* \quad \{p(X) = 0\}$$

- Implementation: **ALIGATOR** $\rightarrow$ programs working on numbers.

  `http://mtc.epfl.ch/software-tools/Aligator`

Outline

Overview of the Method

Algebraic Techniques for Invariant Generation

Polynomial Invariants for Loops with Assignments Only

Polynomial Invariants for Loops with Conditionals

Examples

Conclusions

C-finite Recurrences $x[n]$ in a Field $\mathbb{K}$

$\rightarrow$ linear recurrences with constant coefficients:

$$x[n+r] = a_0 x[n] + \ldots + a_{r-1} x[n+r-1],$$

where

- $r \in \mathbb{N}$ is the recurrence *order*;
- $a_0, \ldots, a_{r-1} \in \mathbb{K}$, with $a_0 \neq 0$.

C-finite Recurrences $x[n]$ in a Field $\mathbb{K}$

→ linear recurrences with constant coefficients:

$$x[n+r] = a_0 x[n] + \ldots + a_{r-1} x[n+r-1],$$

Examples.

- Fibonacci: $x[n+2] = x[n+1] + x[n], \quad x[0] = 0, \; x[1] = 1$
- Tribonacci: $x[n+3] = x[n+2] + x[n+1] + x[n], \quad x[0] = 0, \; x[1] = x[2] = 1$

C-finite Recurrences $x[n]$ in a Field $\mathbb{K}$

$\rightarrow$ linear recurrences with constant coefficients:

$$x[n+r] = a_0 x[n] + \ldots + a_{r-1} x[n+r-1],$$

CHARACTERISTIC POLYNOMIAL $c(y)$ of x[n] is:

$$c(y) = y^r - a_0 - a_1 y - \cdots - a_{r-1} y^{r-1}$$

$\rightarrow$ distinct roots: $\theta_1, \ldots, \theta_s \in \bar{\mathbb{K}}$ with multiplicity $e_i \geq 1$.

$\rightarrow$ CLOSED FORM OF $x[n]$: Linear combination of:

$$\begin{cases} \theta_1^n, & n\theta_1^n, & n(n-1)\theta_1^n, & \ldots\ \ldots & n(n-1)\cdots(n-e_1+1)\theta_1^n \\ \theta_2^n, & n\theta_2^n, & n(n-1)\theta_2^n & \ldots\ \ldots & n(n-1)\cdots(n-e_2+1)\theta_2^n \\ \ldots & \ldots & \ldots & \ldots\ \ldots & \ldots \\ \theta_s^n, & n\theta_s^n, & n(n-1)\theta_s^n & \ldots\ \ldots & n(n-1)\cdots(n-e_s+1)\theta_s^n \end{cases}$$

By regrouping:

$$x[n] = q(n, \theta_1^n, \cdots, \theta_s^n)$$

C-finite Recurrences $x[n]$ in a Field $\mathbb{K}$

$\rightarrow$ linear recurrences with constant coefficients:

$$x[n+r] = a_0 x[n] + \ldots + a_{r-1} x[n+r-1],$$

CHARACTERISTIC POLYNOMIAL $c(y)$ of x[n] is:

$$c(y) = y^r - a_0 - a_1 y - \cdots - a_{r-1} y^{r-1}$$

$\rightarrow$ distinct roots: $\theta_1, \ldots, \theta_s \in \bar{\mathbb{K}}$ with multiplicity $e_i \geq 1$.

$\rightarrow$ CLOSED FORM OF $x[n]$: Linear combination of:

$$\begin{cases} \theta_1^n, & n\theta_1^n, & n(n-1)\theta_1^n, & \ldots \ \ldots & n(n-1)\cdots(n-e_1+1)\theta_1^n \\ \theta_2^n, & n\theta_2^n, & n(n-1)\theta_2^n & \ldots \ \ldots & n(n-1)\cdots(n-e_2+1)\theta_2^n \\ \ldots & \ldots & \ldots & \ldots \ \ldots & \ldots \\ \theta_s^n, & n\theta_s^n, & n(n-1)\theta_s^n & \ldots \ \ldots & n(n-1)\cdots(n-e_s+1)\theta_s^n \end{cases}$$

By regrouping:

$$x[n] = q(n, \theta_1^n, \cdots, \theta_s^n)$$

C-finite Recurrences $x[n]$ in a Field $\mathbb{K}$

→ linear recurrences with constant coefficients:

$$x[n+r] = a_0 x[n] + \ldots + a_{r-1} x[n+r-1],$$

CHARACTERISTIC POLYNOMIAL $c(y)$ of x[n] is:

$$c(y) = y^r - a_0 - a_1 y - \cdots - a_{r-1} y^{r-1}$$

→ distinct roots: $\theta_1, \ldots, \theta_s \in \bar{\mathbb{K}}$ with multiplicity $e_i \geq 1$.

→ CLOSED FORM OF $x[n]$: Linear combination of:

$$\begin{cases} \theta_1^n, & n\theta_1^n, & n(n-1)\theta_1^n, & \ldots\ \ldots & n(n-1)\cdots(n-e_1+1)\theta_1^n \\ \theta_2^n, & n\theta_2^n, & n(n-1)\theta_2^n & \ldots\ \ldots & n(n-1)\cdots(n-e_2+1)\theta_2^n \\ \ldots & \ldots & \ldots & \ldots\ \ldots & \ldots \\ \theta_s^n, & n\theta_s^n, & n(n-1)\theta_s^n & \ldots\ \ldots & n(n-1)\cdots(n-e_s+1)\theta_s^n \end{cases}$$

By regrouping:

$$x[n] = q(n, \theta_1^n, \cdots, \theta_s^n)$$

C-finite Recurrences $x[n]$ in a Field $\mathbb{K}$

Example

Given $\quad x[n+2] = 3x[n+1] - 2x[n] \quad$ with $\quad x[0] = 0,\ x[1] = 1.$

1. Characteristic polynomial:

$$y^2 - 3y + 2 = 0 \quad \to \quad \text{Roots: } \theta_1 = 1,\ \theta_2 = 2 \quad \text{with } e_1 = e_2 = 1$$

2. Closed form of $x[n]$:

$$x[n] = \alpha 1^n + \beta 2^n$$

$$\begin{cases} 0 = & x[0] = & \alpha + \beta \\ 1 = & x[1] = & \alpha + 2\beta \end{cases}$$

3. Closed form of $x[n]$:

$$x[n] = 2^n - 1$$

C-finite Recurrences $x[n]$ in a Field $\mathbb{K}$

Example

Given $x[n+2] = 3x[n+1] - 2x[n]$ with $x[0] = 0,\ x[1] = 1.$

1. Characteristic polynomial:

$$y^2 - 3y + 2 = 0 \quad \rightarrow \text{ Roots: } \theta_1 = 1,\ \theta_2 = 2 \quad \text{with } e_1 = e_2 = 1$$

2. Closed form of $x[n]$:

$$x[n] = \alpha 1^n + \beta 2^n$$

$$\begin{cases} 0 = & x[0] = & \alpha + \beta \\ 1 = & x[1] = & \alpha + 2\beta \end{cases}$$

3. Closed form of $x[n]$:

$$x[n] = 2^n - 1$$

C-finite Recurrences $x[n]$ in a Field $\mathbb{K}$

Example

Given $x[n+2] = 3x[n+1] - 2x[n]$ with $x[0] = 0,\ x[1] = 1.$

1. Characteristic polynomial:

$$y^2 - 3y + 2 = 0 \quad \rightarrow \text{ Roots: } \theta_1 = 1,\ \theta_2 = 2 \quad \text{with } e_1 = e_2 = 1$$

2. Closed form of $x[n]$:

$$x[n] = \alpha 1^n + \beta 2^n$$

$$\begin{cases} 0 = & x[0] = & \alpha + \beta \\ 1 = & x[1] = & \alpha + 2\beta \end{cases} \quad \rightarrow \quad \alpha = -1,\ \beta = 1$$

3. Closed form of $x[n]$:

$$x[n] = 2^n - 1$$

C-finite Recurrences $x[n]$ in a Field $\mathbb{K}$

Example

Given $x[n+2] = 3x[n+1] - 2x[n]$ with $x[0] = 0,\ x[1] = 1.$

1. Characteristic polynomial:

$$y^2 - 3y + 2 = 0 \quad \rightarrow \text{ Roots: } \theta_1 = 1,\ \theta_2 = 2 \quad \text{with } e_1 = e_2 = 1$$

2. Closed form of $x[n]$:

$$x[n] = \alpha 1^n + \beta 2^n$$

$$\begin{cases} 0 = & x[0] = & \alpha + \beta \\ 1 = & x[1] = & \alpha + 2\beta \end{cases} \quad \rightarrow \quad \alpha = -1,\ \beta = 1$$

3. Closed form of $x[n]$:

$$x[n] = 2^n - 1$$

C-finite Recurrences $x[n]$ in a Field $\mathbb{K}$

Example

Given $x[n+2] = 3x[n+1] - 2x[n]$ with $x[0] = 0,\ x[1] = 1.$

1. Characteristic polynomial:

$$y^2 - 3y + 2 = 0 \quad \rightarrow \text{ Roots: } \theta_1 = 1,\ \theta_2 = 2 \quad \text{with } e_1 = e_2 = 1$$

2. Closed form of $x[n]$:

$$x[n] = \alpha 1^n + \beta 2^n$$

$$\begin{cases} 0 = & x[0] = & \alpha + \beta \\ 1 = & x[1] = & \alpha + 2\beta \end{cases} \quad \rightarrow \quad \alpha = -1,\ \beta = 1$$

3. Closed form of $x[n]$:

$$x[n] = 2^n - 1$$

C-finite Recurrences $x[n]$ in a Field $\mathbb{K}$

Example

Given $x[n+2] = 3x[n+1] - 2x[n]$ with $x[0] = 0,\ x[1] = 1.$

1. Characteristic polynomial:

$$y^2 - 3y + 2 = 0 \quad \rightarrow \text{Roots: } \theta_1 = 1,\ \theta_2 = 2 \quad \text{with } e_1 = e_2 = 1$$

2. Closed form of $x[n]$:

$$x[n] = \alpha 1^n + \beta 2^n$$

$$\begin{cases} 0 = & x[0] = & \alpha + \beta \\ 1 = & x[1] = & \alpha + 2\beta \end{cases} \quad \rightarrow \quad \alpha = -1,\ \beta = 1$$

3. Closed form of $x[n]$:

$$x[n] = 2^n - 1$$

Program Assignments and C-finite Recurrences

$$x := \alpha x + h \quad \longrightarrow \quad x[n+1] = \alpha x[n] + h(n)$$

with $h(n) = n^{d_1}\theta_1 + \cdots + n^{d_s}\theta_s^n$

Example

$x[n+1] = x[n] + 1 \quad \rightarrow \quad x[n+2] - 2x[n+1] + x[n] = 0$

$x[n+1] = 2x[n] + 4 \quad \rightarrow \quad x[n+2] - 3x[n+1] + 2x[n] = 0$

$\Downarrow$ $P_x = P_h \cdot (S - \alpha)$, where $P_h = (S - \theta_1)^{d_1+1} \cdots (S - \theta_s)^{d_s+1}$

$P_x \cdot x = 0$ $\qquad$ $P_h \cdot h = 0$

$$x[n+r] = a_{n+r-1}x[n+r-1] + \cdots + a_n x[n], \qquad \text{with } r \geq 1$$

Program Assignments and C-finite Recurrences

$$x := \alpha x + h \quad \longrightarrow \quad x[n+1] = \alpha x[n] + h(n)$$

with $\quad h(n) = n^{d_1}\theta_1 + \cdots + n^{d_s}\theta_s^n$

Example

$$x[n+1] = x[n] + 1 \quad \rightarrow \quad x[n+2] - 2x[n+1] + x[n] = 0$$

$$x[n+1] = 2x[n] + 4 \quad \rightarrow \quad x[n+2] - 3x[n+1] + 2x[n] = 0$$

$\Downarrow$ $P_x = P_h \cdot (S - \alpha)$, $\quad$ where $P_h = (S-\theta_1)^{d_1+1} \cdots (S-\theta_s)^{d_s+1}$

$P_x \cdot x = 0 \qquad\qquad P_h \cdot h = 0$

$$x[n+r] = a_{n+r-1}x[n+r-1] + \cdots + a_n x[n], \qquad \text{with } r \geq 1$$

Program Assignments and C-finite Recurrences

$$x := \alpha x + h \quad \longrightarrow \quad x[n+1] = \alpha x[n] + h(n)$$

with $\quad h(n) = n^{d_1}\theta_1 + \cdots + n^{d_s}\theta_s^n$

Example

$x[n+1] = x[n] + 1 \quad \to \quad x[n+2] - 2x[n+1] + x[n] = 0$

$x[n+1] = 2x[n] + 4 \quad \to \quad x[n+2] - 3x[n+1] + 2x[n] = 0$

$\Downarrow P_x = P_h \cdot (S - \alpha),$ where $P_h = (S-\theta_1)^{d_1+1} \cdots (S-\theta_s)^{d_s+1}$

$P_x \cdot x = 0 \qquad\qquad P_h \cdot h = 0$

$$x[n+r] = a_{n+r-1}x[n+r-1] + \cdots + a_n x[n], \qquad \text{with } r \geq 1$$

Algebraic Dependencies Among Exponential Sequences

Let $\theta_1, \ldots, \theta_s \in \bar{\mathbb{K}}$, and their exponential sequences $\theta_1^n, \ldots, \theta_s^n \in \bar{\mathbb{K}}$.

An algebraic dependency of these sequences is a polynomial p :

$$p(\theta_1^n, \ldots, \theta_s^n) = 0, \quad (\forall n \geq 0).$$

$\rightarrow$ ideal: $I(\theta_1^n, \ldots, \theta_s^n) = I(n, \theta_1^n, \ldots, \theta_s^n)$.

Example.

- The algebraic dependency among the exponential sequences of $\theta_1 = 2$ and $\theta_2 = \frac{1}{2}$ is:

$$\theta_1^n * \theta_2^n - 1 = 0;$$

- The algebraic dependency among the exponential sequences of $\theta_1 = \frac{1+\sqrt{5}}{2}$ and $\theta_2 = \frac{1-\sqrt{5}}{2}$ is:

$$\left(\theta_1^n\right)^2 * \left(\theta_2^n\right)^2 - 1 = 0;$$

- There is no algebraic dependency among the exponential sequences of $\theta_1 = 2$ and $\theta_2 = 3$.

Algebraic Dependencies Among Exponential Sequences

Let $\theta_1, \ldots, \theta_s \in \bar{\mathbb{K}}$, and their exponential sequences $\theta_1^n, \ldots, \theta_s^n \in \bar{\mathbb{K}}$.

An algebraic dependency of these sequences is a polynomial p :

$$p(\theta_1^n, \ldots, \theta_s^n) = 0, \quad (\forall n \geq 0).$$

$\rightarrow$ ideal: $I(\theta_1^n, \ldots, \theta_s^n) = I(n, \theta_1^n, \ldots, \theta_s^n)$.

Example.

- The algebraic dependency among the exponential sequences of $\theta_1 = 2$ and $\theta_2 = \frac{1}{2}$ is:

$$\theta_1^n * \theta_2^n - 1 = 0;$$

- The algebraic dependency among the exponential sequences of $\theta_1 = \frac{1+\sqrt{5}}{2}$ and $\theta_2 = \frac{1-\sqrt{5}}{2}$ is:

$$(\theta_1^n)^2 * (\theta_2^n)^2 - 1 = 0;$$

- There is no algebraic dependency among the exponential sequences of $\theta_1 = 2$ and $\theta_2 = 3$.

Algebraic Dependencies Among Exponential Sequences

Let $\theta_1, \ldots, \theta_s \in \bar{\mathbb{K}}$, and their exponential sequences $\theta_1^n, \ldots, \theta_s^n \in \bar{\mathbb{K}}$.

An algebraic dependency of these sequences is a polynomial p :

$$p(\theta_1^n, \ldots, \theta_s^n) = 0, \quad (\forall n \geq 0).$$

$\rightarrow$ ideal: $I(\theta_1^n, \ldots, \theta_s^n) = I(n, \theta_1^n, \ldots, \theta_s^n)$.

Example.

- The algebraic dependency among the exponential sequences of $\theta_1 = 2$ and $\theta_2 = \frac{1}{2}$ is:

$$\theta_1^n * \theta_2^n - 1 = 0;$$

- The algebraic dependency among the exponential sequences of $\theta_1 = \frac{1+\sqrt{5}}{2}$ and $\theta_2 = \frac{1-\sqrt{5}}{2}$ is:

$$\left(\theta_1^n\right)^2 * \left(\theta_2^n\right)^2 - 1 = 0;$$

- There is no algebraic dependency among the exponential sequences of $\theta_1 = 2$ and $\theta_2 = 3$.

Algebraic Dependencies Among Exponential Sequences

Let $\theta_1, \ldots, \theta_s \in \bar{\mathbb{K}}$, and their exponential sequences $\theta_1^n, \ldots, \theta_s^n \in \bar{\mathbb{K}}$.

An algebraic dependency of these sequences is a polynomial p :

$$p(\theta_1^n, \ldots, \theta_s^n) = 0, \quad (\forall n \geq 0).$$

$\rightarrow$ ideal: $I(\theta_1^n, \ldots, \theta_s^n) = I(n, \theta_1^n, \ldots, \theta_s^n)$.

Example.

- The algebraic dependency among the exponential sequences of $\theta_1 = 2$ and $\theta_2 = \frac{1}{2}$ is:

$$\theta_1^n * \theta_2^n - 1 = 0;$$

- The algebraic dependency among the exponential sequences of $\theta_1 = \frac{1+\sqrt{5}}{2}$ and $\theta_2 = \frac{1-\sqrt{5}}{2}$ is:

$$\left(\theta_1^n\right)^2 * \left(\theta_2^n\right)^2 - 1 = 0;$$

- There is no algebraic dependency among the exponential sequences of $\theta_1 = 2$ and $\theta_2 = 3$.

Outline

Overview of the Method

Algebraic Techniques for Invariant Generation

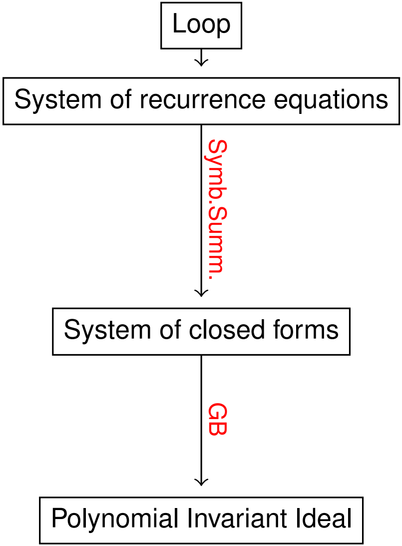
Polynomial Invariants for Loops with Assignments Only

Polynomial Invariants for Loops with Conditionals

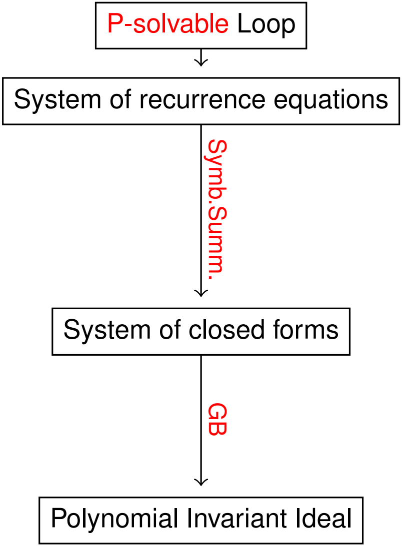
Examples

Conclusions

Invariant Generation Algorithm



Invariant Generation Algorithm



P-solvable Loops: while ... do S end while

The closed forms of the loop variables $X = \{x_1, \ldots, x_m\}$:

$$\mathrm{CF}(S^n, E_S, X, X_0): \begin{cases} x_1[n] &= q_1(n, \theta_1^n, \ldots, \theta_s^n) \\ & \vdots \\ x_m[n] &= q_m(n, \theta_1^n, \ldots, \theta_s^n) \end{cases},$$

with algebraic dependencies $A = I(n, \theta_1^n, \ldots, \theta_s^n)$.

Notations:

1. $n \in \mathbb{N}$ is the loop counter and S^n denotes $\underbrace{S; \ldots; S}_{n \text{ times}}$;
2. $x_i[n]$ is the value of x_i at iteration n.
   X_0 are the initial values of loop variables before S^n;
3. $q_1, \ldots, q_m \in \bar{\mathbb{K}}[n, \theta_1^n, \ldots, \theta_s^n]$;
4. $\theta_1, \ldots, \theta_s \in \bar{\mathbb{K}}$, $E_S = \{\theta_1^n, \ldots, \theta_s^n\}$.

P-solvable Loops: while ... do S end while

The closed forms of the loop variables $X = \{x_1, \dots, x_m\}$:

$$\mathrm{CF}(S^n, E_S, X, X_0): \begin{cases} x_1[n] & = & q_1(n, \theta_1^n, \dots, \theta_s^n) \\ & \vdots & \\ x_m[n] & = & q_m(n, \theta_1^n, \dots, \theta_s^n) \end{cases},$$

with algebraic dependencies $A = I(n, \theta_1^n, \dots, \theta_s^n)$.

Polynomial Invariant Ideal: $I(x_1, \dots, x_m)$.

$$I(x_1, \dots, x_m) \overset{GB}{=} \langle x_i - q_i(n, \theta_1^n, \dots, \theta_s^n) \rangle + A \cap \mathbb{K}[x_1, \dots, x_m]$$

$$\{p(X) = 0 \ \wedge \ X = X_0\} \quad S^* \quad \{p(X) = 0\}$$

Invariant Generation for Loops with Assignments Only

Example. [K. Zuse, 1993]

$z := 0;\ y := 1;\ x := 1/2;$

<u>while</u> ... <u>do</u>

$z := 2 * z - 2 * y - x;$

$y := y + x;$

$x := x/2$

<u>end while</u>

Invariant Generation for Loops with Assignments Only

Example. [K. Zuse, 1993]

$z := 0;\ y := 1;\ x := 1/2;$

<u>while</u> ... <u>do</u>

$z := 2*z - 2*y - x;$

$y := y + x;$

$x := x/2$

<u>end while</u>

$x[n]$ $x[n+1]$

Invariant Generation for Loops with Assignments Only

Example. [K. Zuse, 1993]

$z := 0;\ y := 1;\ x := 1/2;$

<u>while</u> ... <u>do</u>

$z := 2 * z - 2 * y - x;$

$y := y + x;$

$x := x/2$

<u>end while</u>

$x[n] \quad x[n+1] \quad x[n+2] \quad \dots$

Invariant Generation for Loops with Assignments Only

Example. [K. Zuse, 1993]

$z := 0;\ y := 1;\ x := 1/2;$

while ... do

$z := 2 * z - 2 * y - x;$

$y := y + x;$

$x := x/2$

end while

$x[n] \quad x[n+1] \quad x[n+2] \quad \ldots$

$y[n] \quad y[n+1]$

Invariant Generation for Loops with Assignments Only

Example. [K. Zuse, 1993]

$z := 0;\ y := 1;\ x := 1/2;$

<u>while</u> ... <u>do</u>

$z := 2 * z - 2 * y - x;$

$y := y + x;$

$x := x/2$

<u>end while</u>

$x[n]$	$x[n+1]$	$x[n+2]$	...
$y[n]$	$y[n+1]$	$y[n+2]$	...

Invariant Generation for Loops with Assignments Only

Example. [K. Zuse, 1993]

$z := 0;\ y := 1;\ x := 1/2;$
<u>while</u> ... <u>do</u>
    $z := 2 * z - 2 * y - x;$
    $y := y + x;$
    $x := x/2$
<u>end while</u>

$x[n]$	$x[n+1]$	$x[n+2]$	...
$y[n]$	$y[n+1]$	$y[n+2]$	...
$z[n]$	$z[n+1]$		

Invariant Generation for Loops with Assignments Only

Example. [K. Zuse, 1993]

$z := 0;\ y := 1;\ x := 1/2;$

<u>while</u> ... <u>do</u>

$\quad z := 2 * z - 2 * y - x;$

$\quad y := y + x;$

$\quad x := x/2$

<u>end while</u>

$x[n]$	$x[n+1]$	$x[n+2]$	...
$y[n]$	$y[n+1]$	$y[n+2]$	...
$z[n]$	$z[n+1]$	$z[n+2]$	...

Extracting and Solving Recurrences

$z := 0;\ y := 1;\ x := 1/2;$

while ... do

$z := 2 * z - 2 * y - x;$

$y := y + x;$

$x := x/2$

end while

$n \geq 0$

$$\begin{cases} x[n+1] = x[n]/2 \\ y[n+1] = y[n] + x[n] \\ z[n] = 2 * z[n] - 2 * y[n] - x[n] \end{cases}$$

$$\begin{cases} x[n] \overset{C-finite}{=} \frac{1}{2^n} x[0] \\ y[n] \overset{C-finite}{=} y[0] + 2x[0] - \frac{1}{2^{n-1}} x[0] \\ z[n] \overset{C-finite}{=} 2^n (z[0] - 2y[0] - 2x[0]) - \frac{1}{2^{n-1}} x[0] + 2y[0] + 4x[0] \end{cases}$$

Extracting and Solving Recurrences

$z := 0;\ y := 1;\ x := 1/2;$

<u>while</u> ... <u>do</u>

$z := 2 * z - 2 * y - x;$

$y := y + x;$

$x := x/2$

<u>end while</u>

$n \geq 0$

$$\begin{cases} x[n+1] & = & x[n]/2 \\ y[n+1] & = & y[n] + x[n] \\ z[n] & = & 2 * z[n] - 2 * y[n] - x[n] \end{cases}$$

$$\begin{cases} x[n] & \overset{C-finite}{=} & \frac{1}{2^n} x[0] \\ y[n] & \overset{C-finite}{=} & y[0] + 2x[0] - \frac{1}{2^{n-1}} x[0] \\ z[n] & \overset{C-finite}{=} & 2^n(z[0] - 2y[0] - 2x[0]) - \\ & & \frac{1}{2^{n-1}} x[0] + 2y[0] + 4x[0] \end{cases}$$

Algebraic Dependencies and Variable Elimination

$z := 0;\ y := 1;\ x := 1/2;$

$\underline{\text{while}}[\ldots,$

$\quad z := 2 * z - 2 * y - x;$

$\quad y := y + x;$

$\quad x := x/2\ ]$

$n \geq 0, \quad a = 2^n, b = 2^{-n}$

$$\begin{cases} x & \overset{C-finite}{=} & b * x[0] \\ y & \overset{C-finite}{=} & y[0] + 2 * x[0] - 2 * b * x[0] \\ z & \overset{C-finite}{=} & a * (z[0] - 2 * y[0] - 2 * x[0]) - \\ & & 2 * b * x[0] + 2 * y[0] + 4 * x[0] \\ 0 & \overset{Alg.Dep.}{=} & a * b - 1 \end{cases}$$

P-solvable Loop

Algebraic Dependencies and Variable Elimination

$z := 0;\ y := 1;\ x := 1/2;$
$\underline{\text{while}}[\ldots,$
$\quad z := 2*z - 2*y - x;$
$\quad y := y + x;$
$\quad x := x/2\ ]$

$n \geq 0, \quad a = 2^n, b = 2^{-n}$

$$\begin{cases} x & \overset{C-finite}{=} & b*x[0] \\ y & \overset{C-finite}{=} & y[0] + 2*x[0] - 2*b*x[0] \\ z & \overset{C-finite}{=} & a*(z[0] - 2*y[0] - 2*x[0]) - \\ & & 2*b*x[0] + 2*y[0] + 4*x[0] \\ 0 & \overset{Alg.Dep.}{=} & a*b - 1 \end{cases}$$

P-solvable Loop

Polynomial Invariant (GB):

$$(2x+y-2x[0]-y[0]=0) \wedge (y^2-y*z+2*z*x[0]+z*y[0]-y[0]^2-2*x[0]*z[0]=0)$$

Algebraic Dependencies and Variable Elimination

$z := 0;\ y := 1;\ x := 1/2;$

$\underline{\text{while}}[\ldots,$

$\quad z := 2 * z - 2 * y - x;$

$\quad y := y + x;$

$\quad x := x/2\ ]$

$n \geq 0, \quad a = 2^n, b = 2^{-n}$

$$\begin{cases} x & \overset{C-finite}{=} & b * x[0] \\ y & \overset{C-finite}{=} & y[0] + 2 * x[0] - 2 * b * x[0] \\ z & \overset{C-finite}{=} & a * (z[0] - 2 * y[0] - 2 * x[0]) - \\ & & 2 * b * x[0] + 2 * y[0] + 4 * x[0] \\ 0 & \overset{Alg.Dep.}{=} & a * b - 1 \end{cases}$$

P-solvable Loop

Polynomial Invariant (GB):

$$(2 * x + y - 2 = 0)\ \wedge\ (2 * z - y * z + y^2 - 1 = 0)$$

Further Examples.

$k := 0;\ j := 1;\ m := 1;$
<u>while</u> $m \leq n$ <u>do</u>
  $k := k + 1;\ j := j + 2;\ m := m + j$
<u>end while</u>

$i := 0;\ f := 1; g := 1$
<u>while</u> $i < n$ <u>do</u>
  $t := f;\ f := f + g;\ g := t$
<u>end while</u>

Further Examples.

(1) Integer square root:

$k := 0;\; j := 1;\; m := 1;$
<u>while</u> $m \leq n$ <u>do</u>
  $k := k + 1;\; j := j + 2;\; m := m + j$
<u>end while</u>

(2) Fibonacci numbers

$i := 0;\; f := 1; g := 1$
<u>while</u> $i < n$ <u>do</u>
  $t := f;\; f := f + g;\; g := t$
<u>end while</u>

Further Examples.

(1) Integer square root:

$k := 0;\ j := 1;\ m := 1;$
<u>while</u> $m \leq n$ <u>do</u>
  $k := k + 1;\ j := j + 2;\ m := m + j$
<u>end while</u>

Invariant:

$$2k + 1 = j \wedge 4m = (j + 1)^2$$

(2) Fibonacci numbers

$i := 0;\ f := 1; g := 1$
<u>while</u> $i < n$ <u>do</u>
  $t := f;\ f := f + g;\ g := t$
<u>end while</u>

Invariant:

$$f^4 + g^4 + 2f * g^3 - 2f^3 * g - f^2 * g^2 - 1 = 0$$

Outline

Overview of the Method

Algebraic Techniques for Invariant Generation

Polynomial Invariants for Loops with Assignments Only

Polynomial Invariants for Loops with Conditionals

Examples

Conclusions

Overview of Our Method - Further Considerations

- Loops with assignments and with conditional branches.

  Structural constraints on assignments with polynomial rhs.

  $\leftarrow$ Summation algorithms (Gosper's, C-finite)

  Tests are ignored $\rightarrow$ non-deterministic programs:

  $$\underline{\text{if}}[b\ \underline{\text{then}}\ S_1\ \underline{\text{else}}\ S_2] \rightarrow \underline{\text{if}}[\ldots\underline{\text{then}}\ S_1\ \underline{\text{else}}\ S_2] \longrightarrow S_1|S_2$$
  $$\underline{\text{while}}[cond, S] \rightarrow \underline{\text{while}}[\ldots, S] \longrightarrow S^*$$
  $$\underline{\text{while}}[\ldots, \underline{\text{if}}[\ldots, S_1]; \ldots; \underline{\text{if}}[\ldots, S_k]] \longrightarrow (S_1|\ldots|S_k)^*$$

- Automated Loop Invariant Generation by Algebraic Techniques Over the Rationals for P-solvable loops:

  $$\{p(X) = 0 \ \wedge \ X = X_0\} \quad (S_1|\ldots|S_k)^* \quad \{p(X) = 0\}$$

- Polynomial invariant ideal represented by Gröbner bases;

Overview of Our Method - Further Considerations

- Loops with assignments and with conditional branches.

  Structural constraints on assignments with polynomial rhs.

  $\leftarrow$ Summation algorithms (Gosper's, C-finite)

  Tests are ignored $\rightarrow$ non-deterministic programs:

  $\underline{\text{if}}[b\ \underline{\text{then}}\ S_1\ \underline{\text{else}}\ S_2] \rightarrow \underline{\text{if}}[\ldots\underline{\text{then}}\ S_1\ \underline{\text{else}}\ S_2] \longrightarrow S_1|S_2$

  $\underline{\text{while}}[cond, S] \rightarrow \underline{\text{while}}[\ldots, S] \longrightarrow S^*$

  $\underline{\text{while}}[\ldots, \underline{\text{if}}[\ldots, S_1]; \ldots; \underline{\text{if}}[\ldots, S_k]] \longrightarrow (S_1|\ldots|S_k)^*$

- Automated Loop Invariant Generation by Algebraic Techniques Over the Rationals for P-solvable loops:

  $$\{p(X) = 0 \ \wedge \ X = X_0\} \quad (S_1|\ldots|S_k)^* \quad \{p(X) = 0\}$$

- Polynomial invariant ideal represented by Gröbner bases;

Overview of Our Method - Further Considerations

- Loops with assignments and with conditional branches.

  Structural constraints on assignments with polynomial rhs.

  $\leftarrow$ Summation algorithms (Gosper's, C-finite)

  Tests are ignored $\rightarrow$ non-deterministic programs:

  $\underline{\text{if}}[b\ \underline{\text{then}}\ S_1\ \underline{\text{else}}\ S_2] \rightarrow \underline{\text{if}}[\ldots \underline{\text{then}}\ S_1\ \underline{\text{else}}\ S_2] \longrightarrow S_1|S_2$

  $\underline{\text{while}}[cond, S] \rightarrow \underline{\text{while}}[\ldots, S] \longrightarrow S^*$

  $\underline{\text{while}}[\ldots, \underline{\text{if}}[\ldots, S_1]; \ldots; \underline{\text{if}}[\ldots, S_k]] \longrightarrow (S_1|\ldots|S_k)^*$

- Automated Loop Invariant Generation by Algebraic Techniques Over the Rationals for P-solvable loops:

  $$\{p(X) = 0\ \wedge\ X = X_0\}\quad (S_1|\ldots|S_k)^*\quad \{p(X) = 0\}$$

- Polynomial invariant ideal represented by Gröbner bases;

Overview of Our Method - Further Considerations

- Loops with assignments and with/without conditional branches.

  Structural constraints on assignments with polynomial rhs.

  $\leftarrow$ Summation algorithms (Gosper's, C-finite)

  Tests are ignored $\rightarrow$ non-deterministic programs:

  $\underline{\text{if}}[b\ \underline{\text{then}}\ S_1\ \underline{\text{else}}\ S_2] \rightarrow \underline{\text{if}}[\ldots\underline{\text{then}}\ S_1\ \underline{\text{else}}\ S_2] \longrightarrow S_1|S_2$
  $\underline{\text{while}}[b, S] \rightarrow \underline{\text{while}}[\ldots, S] \longrightarrow S^*$
  $\underline{\text{while}}[\ldots, \underline{\text{if}}[\ldots, S_1]; \ldots; \underline{\text{if}}[\ldots, S_k]] \longrightarrow (S_1|\ldots|S_k)^*$

- **A**utomated **L**oop **I**nvariant **G**eneration by **A**lgebraic **T**echniques **O**ver the **R**ationals for P-solvable loops:

  $$\{p(X) = 0\ \wedge\ X = X_0\}\quad (S_1|\ldots|S_k)^*\quad \{p(X) = 0\}$$

- Polynomial invariant ideal represented by Gröbner bases;

- Implementation: `ALIGATOR` $\rightarrow$ programs working on numbers.

RECAP – P-solvable Loop S^* with Assignments Only

Values of loop variables $X = \{x_1, \ldots, x_m\}$ at loop iteration $n \in \mathbb{N}$:

$$S^n \equiv \underbrace{S; \ldots; S}_{n \text{ times}} : \begin{cases} x_1[n] = q_1(n, \theta_1^n, \ldots, \theta_s^n) \\ \vdots \\ x_m[n] = q_m(n, \theta_1^n, \ldots, \theta_s^n) \end{cases},$$

with algebraic dependencies:

$$A = I(n, \theta_1^n, \ldots, \theta_s^n) = \langle r \mid r(n, \theta_1^n, \ldots, \theta_s^n) = 0, \ \forall n \in \mathbb{N} \rangle \trianglelefteq \bar{\mathbb{K}}[n, \theta_1^n, \ldots, \theta_s^n]$$

RECAP – P-solvable Loop S^* with Assignments Only

Values of loop variables $X = \{x_1, \ldots, x_m\}$ at loop iteration $n \in \mathbb{N}$:

$$S^n \equiv \underbrace{S; \ldots; S}_{n \text{ times}} : \left\{ \begin{array}{lcl} x_1[n] & = & q_1(n, \theta_1^n, \ldots, \theta_s^n) \\ & \vdots & \\ x_m[n] & = & q_m(n, \theta_1^n, \ldots, \theta_s^n) \end{array} \right. ,$$

with algebraic dependencies:

$$A = I(n, \theta_1^n, \ldots, \theta_s^n) = \langle r \mid r(n, \theta_1^n, \ldots, \theta_s^n) = 0,\ \forall n \in \mathbb{N} \rangle \trianglelefteq \bar{\mathbb{K}}[n, \theta_1^n, \ldots, \theta_s^n]$$

Examples.

- $I(n, 2^n, 4^n) = \langle (2^n)^2 - 4^n \rangle$
- $I(n, 2^n, 2^{-n}) = \langle (2^n) * (2^{-n}) - 1 \rangle$
- $I(n, 2^n, 3^n) = \emptyset$
- $I(n, \frac{1+\sqrt{5}}{2}^n, \frac{1-\sqrt{5}}{2}^n) = \langle (\frac{1+\sqrt{5}}{2}^n)^2 * (\frac{1-\sqrt{5}}{2}^n)^2 - 1 \rangle$

RECAP – P-solvable Loop S^* with Assignments Only

Values of loop variables $X = \{x_1, \dots, x_m\}$ at loop iteration $n \in \mathbb{N}$:

$$S^n \equiv \underbrace{S; \dots; S}_{n \text{ times}} : \begin{cases} x_1[n] & = & q_1(n, \theta_1^n, \dots, \theta_s^n) \\ & \vdots & \\ x_m[n] & = & q_m(n, \theta_1^n, \dots, \theta_s^n) \end{cases},$$

with algebraic dependencies:

$$A = I(n, \theta_1^n, \dots, \theta_s^n) = \langle r \mid r(n, \theta_1^n, \dots, \theta_s^n) = 0,\ \forall n \in \mathbb{N} \rangle \trianglelefteq \bar{\mathbb{K}}[n, \theta_1^n, \dots, \theta_s^n]$$

Polynomial Invariant Ideal: $I_* = I(x_1, \dots, x_m) = \langle x_i - q_i \rangle + A \cap \mathbb{K}[x_1, \dots, x_m]$

$$\{p(X) = 0 \ \wedge \ X = X_0\} \quad S^* \quad \{p(X) = 0\}$$

P-solvable Loops with Conditionals $(S_1 | \dots | S_k)^*$

P-solvable loop $(S_1 | \dots | S_k)^*$ $\Longleftrightarrow$ P-solvable inner loops S_i^*

$(S_1 | \dots | S_k)^*$ is equivalent to $(S_1^* * \dots * S_k^*)^*$

Polynomial Invariant Ideal: $I_* = I(x_1, \dots, x_m)$

P-solvable Loops with Conditionals $(S_1|\ldots|S_k)^*$

P-solvable loop $(S_1|\ldots|S_k)^* \Longleftrightarrow$ P-solvable inner loops S_i^*

$(S_1|\ldots|S_k)^*$ is equivalent to $(S_1^* * \cdots * S_k^*)^*$

Polynomial Invariant Ideal: $I_* = I(x_1, \ldots, x_m)$

$$\{p(X) = 0 \ \wedge \ X = X_0\} \quad (S_1|\ldots|S_k)^* \quad \{p(X) = 0\}$$

P-solvable Loops with Conditionals $(S_1|\dots|S_k)^*$

P-solvable loop $(S_1|\dots|S_k)^* \iff$ P-solvable inner loops S_i^*

$(S_1|\dots|S_k)^*$ is equivalent to $(S_1^* * \dots * S_k^*)^*$

Polynomial Invariant Ideal: $I_* = I(x_1, \dots, x_m)$

$$\{p(X) = 0 \ \wedge \ X = X_0\} \quad (S_1^* * \dots * S_k^*)^* \quad \{p(X) = 0\}$$

P-solvable Loops with Conditionals $(S_1|\dots|S_k)^*$

P-solvable loop $(S_1|\dots|S_k)^* \Longleftrightarrow$ P-solvable inner loops S_i^*

$(S_1|\dots|S_k)^*$ is equivalent to $(S_1^* * \dots * S_k^*)^*$

Polynomial Invariant Ideal: $I_* = I(x_1, \dots, x_m)$

$$\{p(X) = 0 \ \wedge \ X = X_0\} \quad (S_1^* * \dots * S_k^*)^* \quad \{p(X) = 0\}$$

First algorithmic attempt:

$$\underbrace{(S_1^* * \dots * S_k^*)^*}_{\text{arbitrary MANY iterations}} \dashleftarrow \underbrace{S_1^* * \dots * S_k^*}_{\text{ONE iteration}} \leftarrow \underbrace{S_1^*; \dots; S_k^*}_{\text{loop sequence}}$$

$$I_* \quad \subseteq \quad PI_0 = \bigcap I_k \quad\quad I_k$$

P-solvable Loops with Conditionals $(S_1 | \dots | S_k)^*$

P-solvable loop $(S_1 | \dots | S_k)^*$ $\Longleftrightarrow$ P-solvable inner loops S_i^*

$(S_1 | \dots | S_k)^*$ is equivalent to $(S_1^* * \dots * S_k^*)^*$

Polynomial Invariant Ideal: $I_* = I(x_1, \dots, x_m)$

$$\{p(X) = 0 \ \wedge \ X = X_0\} \quad (S_1^* * \dots * S_k^*)^* \quad \{p(X) = 0\}$$

Idea of the algorithm:

$$\underbrace{(S_1^* * \dots * S_k^*)^*}_{\text{arbitrary MANY iterations}} \leftarrow \dots \leftarrow \underbrace{(S_1^* * \dots * S_k^*); S_i^*}_{\text{TWO iterations}} \leftarrow \underbrace{S_1^* * \dots * S_k^*}_{\text{ONE iteration}}$$

$$I_* \subseteq \dots \subseteq PI_1 = \bigcap I_{k+1} \subseteq PI_0 = \bigcap I_k$$

P-solvable Loops with Conditionals $(S_1|\dots|S_k)^*$

P-solvable loop $(S_1|\dots|S_k)^* \Longleftrightarrow$ P-solvable inner loops S_i^*

$(S_1|\dots|S_k)^*$ is equivalent to $(S_1^* * \dots * S_k^*)^*$

Polynomial Invariant Ideal: $I_* = I(x_1, \dots, x_m)$

$$\{p(X) = 0 \ \wedge \ X = X_0\} \quad (S_1^* * \dots * S_k^*)^* \quad \{p(X) = 0\}$$

Idea of the algorithm:

$$\underbrace{(S_1^* * \dots * S_k^*)^*}_{\text{arbitrary MANY iterations}} \leftarrow \dots \leftarrow \underbrace{(S_1^* * \dots * S_k^*); S_i^*}_{\text{TWO iterations}} \leftarrow \underbrace{S_1^* * \dots * S_k^*}_{\text{ONE iteration}}$$

$$I_* \subseteq \dots \subseteq PI_1 = \bigcap I_{k+1} \subseteq PI_0 = \bigcap I_k$$

Termination? Not guaranteed for arbitrary ideals!

P-solvable Loops with Conditionals $(S_1 | \dots | S_k)^*$

P-solvable loop $(S_1 | \dots | S_k)^* \iff$ P-solvable inner loops S_i^*

$(S_1 | \dots | S_k)^*$ is equivalent to $(S_1^* * \dots * S_k^*)^*$

Polynomial Invariant Ideal: $I_* = I(x_1, \dots, x_m)$

$$\{p(X) = 0 \ \wedge \ X = X_0\} \quad (S_1^* * \dots * S_k^*)^* \quad \{p(X) = 0\}$$

Idea of the algorithm:

$$\underbrace{(S_1^* * \dots * S_k^*)^*}_{\text{arbitrary MANY iterations}} \leftarrow \dots \leftarrow \underbrace{(S_1^* * \dots * S_k^*); S_i^*}_{\text{TWO iterations}} \leftarrow \underbrace{S_1^* * \dots * S_k^*}_{\text{ONE iteration}}$$

$$I_* \subseteq \dots \subseteq PI_1 = \bigcap I_{k+1} \subseteq PI_0 = \bigcap I_k$$

Termination? Always terminates!

P-solvable Loops with Conditionals $(S_1 | \ldots | S_k)^*$

Program

while $[\ldots,$

if$[\ldots, S_1]$;

$\vdots$

if$[\ldots, S_k]]$

Algorithm

$l = k, \quad L_l = \text{Perm}[1, \ldots, k]$

$$PI = \bigcap_{I_l = \{ p \mid \{p(X)=0 \wedge X = X_0\} S^*_{w_1}; \ldots; S^*_{w_l} \{p(X)=0\} \},\ (w_1, \ldots, w_l) \in L_l} I_k$$

repeat

$PI' = PI$

$$L_{l+1} = \bigcup_{i=1}^{k} L_l \circ S_i, \quad l = l+1$$

$$PI = \bigcap_{I_l = \{ p \mid \{p(X)=0 \wedge X = X_0\} S^*_{w_1}; \ldots; S^*_{w_l}; \{p(X)=0\} \},\ (w_1, \ldots, w_l) \in L_l} I_l$$

until $PI = PI' = \mathbf{I}_*$

Idea of the algorithm:

$$\underbrace{(S_1^* * \cdots * S_k^*)^*}_{\text{arbitrary MANY iterations}} \leftarrow \ldots \leftarrow \underbrace{(S_1^* * \cdots * S_k^*); S_i^*}_{\text{TWO iterations}} \leftarrow \underbrace{S_1^* * \cdots * S_k^*}_{\text{ONE iteration}}$$

$$I_* \subseteq \ldots \subseteq PI_1 = \bigcap I_{k+1} \subseteq PI_0 = \bigcap I_k$$

Termination? Always terminates!

Proof of Termination - Example

Initial values: $a = x,\ b = y,\ p = 1,\ q = 0,\ r = 0,\ s = 1$

Loop: $\underbrace{a := a - b; p := p - q; r := r - s}_{S_1} \ \mid\ \underbrace{b := b - a; q := q - p; s := s - r}_{S_2}$

Step 0: $PI_0 = \langle \text{18 polynomials} \rangle =$

Step 1: $PI_1 = \langle b - qx - sy, br - as + x, qr - ps + 1, px + ry - a, bp - aq - y \rangle$

Step 2: $PI_2 = \langle b - qx - sy, br - as + x, qr - ps + 1, px + ry - a, bp - aq - y \rangle$

$$PI_1 = PI_2 = I_*$$

Proof of Termination - Example

Initial values: $a = x,\ b = y,\ p = 1,\ q = 0,\ r = 0,\ s = 1$

Loop: $\underbrace{a := a - b; p := p - q; r := r - s}_{S_1} \mid \underbrace{b := b - a; q := q - p; s := s - r}_{S_2}$

Step 0: $PI_0 = \langle 18 \text{ polynomials} \rangle =$

$$\left\{ \bigwedge \begin{array}{l} P(a,b,p,q,r,s) = 0 \\ (a,b,p,q,r,s) = (x,y,1,0,0,1) \end{array} \right\} \quad \begin{array}{l} S_1^*; S_2^* \\ S_2^*; S_1^* \end{array} \quad \{P(a,b,p,q,r,s) = 0\}$$

Step 1: $PI_1 = \langle b - qx - sy, br - as + x, qr - ps + 1, px + ry - a, bp - aq - y \rangle$

Step 2: $PI_2 = \langle b - qx - sy, br - as + x, qr - ps + 1, px + ry - a, bp - aq - y \rangle$

$$PI_1 = PI_2 = I_*$$

Proof of Termination - Example

Initial values: $a = x,\ b = y,\ p = 1,\ q = 0,\ r = 0,\ s = 1$

Loop: $\underbrace{a := a - b; p := p - q; r := r - s}_{S_1} \ \mid\ \underbrace{b := b - a; q := q - p; s := s - r}_{S_2}$

Step 0: $PI_0 = \langle 18 \text{ polynomials} \rangle =$

$$\left\{ \bigwedge \begin{array}{l} P(a,b,p,q,r,s) = 0 \\ (a,b,p,q,r,s) = (x,y,1,0,0,1) \end{array} \right\} \quad \begin{array}{l} S_1^*; S_2^* \\ S_2^*; S_1^* \end{array} \quad \{P(a,b,p,q,r,s) = 0\}$$

Step 1: $PI_1 = \langle b - qx - sy, br - as + x, qr - ps + 1, px + ry - a, bp - aq - y \rangle$

$$\left\{ \bigwedge \begin{array}{l} P(a,b,p,q,r,s) = 0 \\ (a,b,p,q,r,s) = (x,y,1,0,0,1) \end{array} \right\} \quad \begin{array}{l} S_1^*; S_2^*; S_1^* \\ S_1^*; S_2^*; S_2^* \\ S_2^*; S_1^*; S_2^* \\ S_2^*; S_1^*; S_1^* \end{array} \quad \{P(a,b,p,q,r,s) = 0\}$$

Step 2: $PI_2 = \langle b - qx - sy, br - as + x, qr - ps + 1, px + ry - a, bp - aq - y \rangle$

$$PI_1 = PI_2 = I_*$$

Proof of Termination - Example

Initial values: $a = x,\ b = y,\ p = 1,\ q = 0,\ r = 0,\ s = 1$

Loop: $\underbrace{a := a - b; p := p - q; r := r - s}_{S_1} \quad | \quad \underbrace{b := b - a; q := q - p; s := s - r}_{S_2}$

Step 0: $PI_0 = \langle 18 \text{ polynomials} \rangle =$

$$\left\{ \bigwedge \begin{array}{l} P(a,b,p,q,r,s) = 0 \\ (a,b,p,q,r,s) = (x,y,1,0,0,1) \end{array} \right\} \quad \begin{array}{l} S_1^*; S_2^* \\ S_2^*; S_1^* \end{array} \quad \{P(a,b,p,q,r,s) = 0\}$$

Step 1: $PI_1 = \langle b - qx - sy, br - as + x, qr - ps + 1, px + ry - a, bp - aq - y \rangle$

$$\left\{ \bigwedge \begin{array}{l} P(a,b,p,q,r,s) = 0 \\ (a,b,p,q,r,s) = (x,y,1,0,0,1) \end{array} \right\} \quad \begin{array}{l} S_1^*; S_2^*; S_1^* \\ S_1^*; S_2^*; S_2^* \\ S_2^*; S_1^*; S_2^* \\ S_2^*; S_1^*; S_1^* \end{array} \quad \{P(a,b,p,q,r,s) = 0\}$$

Step 2: $PI_2 = \langle b - qx - sy, br - as + x, qr - ps + 1, px + ry - a, bp - aq - y \rangle$

$$PI_1 = PI_2 = I_*$$

Proof of Termination - Example

Initial values: $a = x,\ b = y,\ p = 1,\ q = 0,\ r = 0,\ s = 1$

Loop: $\underbrace{a := a - b; p := p - q; r := r - s}_{S_1} \ \mid\ \underbrace{b := b - a; q := q - p; s := s - r}_{S_2}$

Step 0: $PI_0 = \langle 18 \text{ polynomials} \rangle =$

$$\left\{ \bigwedge \begin{array}{l} P(a,b,p,q,r,s) = 0 \\ (a,b,p,q,r,s) = (x,y,1,0,0,1) \end{array} \right\} \quad \begin{array}{l} S_1^*; S_2^* \\ S_2^*; S_1^* \end{array} \quad \{P(a,b,p,q,r,s) = 0\}$$

Step 1: $PI_1 = \langle b - qx - sy, br - as + x, qr - ps + 1, px + ry - a, bp - aq - y \rangle$

$$\left\{ \bigwedge \begin{array}{l} P(a,b,p,q,r,s) = 0 \\ (a,b,p,q,r,s) = (x,y,1,0,0,1) \end{array} \right\} \quad \begin{array}{l} S_1^*; S_2^*; S_1^* \\ S_1^*; S_2^*; S_2^* \\ S_2^*; S_1^*; S_2^* \\ S_2^*; S_1^*; S_1^* \end{array} \quad \{P(a,b,p,q,r,s) = 0\}$$

Step 2: $PI_2 = \langle b - qx - sy, br - as + x, qr - ps + 1, px + ry - a, bp - aq - y \rangle$

$$PI_1 = PI_2 = I_*$$

$$\left\{ \bigwedge \begin{array}{l} P(a,b,p,q,r,s) = 0 \\ (a,b,p,q,r,s) = (x,y,1,0,0,1) \end{array} \right\} \quad (S_1 | S_2)^* \quad \{P(a,b,p,q,r,s) = 0\}$$

Proof of Termination - Example

Initial values: $a = x,\ b = y,\ p = 1,\ q = 0,\ r = 0,\ s = 1$

Loop: $\underbrace{a := a - b; p := p - q; r := r - s}_{S_1} \ \mid\ \underbrace{b := b - a; q := q - p; s := s - r}_{S_2}$

Step 0: $PI_0 = \langle 18 \text{ polynomials} \rangle =$

$$\bigcap \begin{array}{l} \langle s-1, b-qx-y, br-a+x, qr-p+1, px+ry-a, bp-aq-y \rangle \\ \langle p-1, b-qx-sy, br-as+x, qr-s+1, x+ry-a, b-aq-y, asy-ab+bx-xy \rangle \end{array}$$

Dimension: 4

Step 1: $PI_1 = \langle b-qx-sy, br-as+x, qr-ps+1, px+ry-a, bp-aq-y \rangle$

Dimension: 3

Step 2: $PI_2 = \langle b-qx-sy, br-as+x, qr-ps+1, px+ry-a, bp-aq-y \rangle$

$$PI_1 = PI_2 = I_*$$

Dimension: 3

Proof of Termination

Properties of the algorithm:

- $I_* \subseteq \cdots \subseteq \; PI_1 = \bigcap I_{k+1} \;\subseteq\; PI_0 = \bigcap I_k$
- if $PI_n = PI_{n+1}$ then $I_* = PI_n$ (TERMINATION CRITERIA)

Proof of Termination

Assume $PI_1 = \bigcap I_{k+1} \subsetneq PI_0 = \bigcap I_k$. Then:

- $I_{k+1} \subsetneq I_k$ for some k-loop sequence;
- $\dim W_{r'} < \dim U_r$ for some r, r'.

Termination Proof.

1. I_{k+1}, I_k are prime ideals;

   $pq \in I_k \implies p \in I_k$ or $q \in I_k$

2. The minimal prime ideal decomposition:

$$I_k = \Big(\bigcap_r U_r\Big) \cap \bar{\mathbb{K}}[X] \quad \text{and} \quad I_{k+1} = \Big(\bigcap_{r'} W_{r'}\Big) \cap \bar{\mathbb{K}}[X]$$

   with

   - $U_r, W_{r'}$ are prime ideals;
   - $U_a \not\subseteq U_b$ and $W_{a'} \not\subseteq W_{b'}$ for any $a \neq b$ and $a' \neq b'$;
   - $(\forall r')(\exists r)\ \dim W_{r'} \leq \dim U_r$.

3. Dimension is finite. It cannot infinitely decrease.

$$(\exists n)\ PI_n = \bigcap I_{k+n} = \bigcap I_{k+n+1} = PI_{n+1}$$

Proof of Termination

Assume $PI_1 = \bigcap I_{k+1} \subsetneq PI_0 = \bigcap I_k$. Then:

- $I_{k+1} \subsetneq I_k$ for some k-loop sequence;
- $\dim W_{r'} < \dim U_r$ for some r, r'.

Termination Proof.

1. I_{k+1}, I_k are prime ideals;
   $pq \in I_k \implies p \in I_k$ or $q \in I_k$
2. The minimal prime ideal decomposition:
   $$I_k = \left(\bigcap_r U_r\right) \cap \bar{\mathbb{K}}[X] \quad \text{and} \quad I_{k+1} = \left(\bigcap_{r'} W_{r'}\right) \cap \bar{\mathbb{K}}[X]$$
   with
   - $U_r, W_{r'}$ are prime ideals;
   - $U_a \not\subseteq U_b$ and $W_{a'} \not\subseteq W_{b'}$ for any $a \neq b$ and $a' \neq b'$;
   - $(\forall r')(\exists r)\ \dim W_{r'} \leq \dim U_r$.
3. Dimension is finite. It cannot infinitely decrease.
   $$(\exists n)\ PI_n = \bigcap I_{k+n} = \bigcap I_{k+n+1} = PI_{n+1}$$

Proof of Termination

Assume $PI_1 = \bigcap I_{k+1} \subsetneq PI_0 = \bigcap I_k$. Then:

- $I_{k+1} \subsetneq I_k$ for some k-loop sequence;
- $\dim W_{r'} < \dim U_r$ for some r, r'.

Termination Proof.

1. I_{k+1}, I_k are prime ideals;

   $pq \in I_k \implies p \in I_k$ or $q \in I_k$

2. The minimal prime ideal decomposition:

$$I_k = \left(\bigcap_r U_r\right) \cap \bar{\mathbb{K}}[X] \quad \text{and} \quad I_{k+1} = \left(\bigcap_{r'} W_{r'}\right) \cap \bar{\mathbb{K}}[X]$$

   with

   - $U_r, W_{r'}$ are prime ideals;
   - $U_a \not\subseteq U_b$ and $W_{a'} \not\subseteq W_{b'}$ for any $a \neq b$ and $a' \neq b'$;
   - $(\forall r')(\exists r)\ \dim W_{r'} \leq \dim U_r$.

3. Dimension is finite. It cannot infinitely decrease.

$$(\exists n)\ PI_n = \bigcap I_{k+n} = \bigcap I_{k+n+1} = PI_{n+1}$$

Proof of Termination

Assume $PI_1 = \bigcap I_{k+1} \subsetneq PI_0 = \bigcap I_k$. Then:

- $I_{k+1} \subsetneq I_k$ for some k-loop sequence;
- $\dim W_{r'} < \dim U_r$ for some r, r'.

Termination Proof.

1. I_{k+1}, I_k are prime ideals;

   $pq \in I_k \implies p \in I_k$ or $q \in I_k$

2. The minimal prime ideal decomposition:

$$I_k = \left(\bigcap_r U_r\right) \cap \bar{\mathbb{K}}[X] \quad \text{and} \quad I_{k+1} = \left(\bigcap_{r'} W_{r'}\right) \cap \bar{\mathbb{K}}[X]$$

   with

   - $U_r, W_{r'}$ are prime ideals;
   - $U_a \not\subseteq U_b$ and $W_{a'} \not\subseteq W_{b'}$ for any $a \neq b$ and $a' \neq b'$;
   - $(\forall r')(\exists r)\ \dim W_{r'} \leq \dim U_r$.

3. Dimension is finite. It cannot infinitely decrease.

$$(\exists n)\ PI_n = \bigcap I_{k+n} = \bigcap I_{k+n+1} = PI_{n+1}$$

Proof of Termination

Assume $PI_1 = \bigcap I_{k+1} \subsetneq PI_0 = \bigcap I_k$. Then:

- $I_{k+1} \subsetneq I_k$ for some k-loop sequence;
- $\dim W_{r'} < \dim U_r$ for some r, r'.

Termination Proof.

1. I_{k+1}, I_k are prime ideals;

   $pq \in I_k \implies p \in I_k$ or $q \in I_k$

2. The minimal prime ideal decomposition:

$$I_k = \left(\bigcap_r U_r\right) \cap \bar{\mathbb{K}}[X] \quad \text{and} \quad I_{k+1} = \left(\bigcap_{r'} W_{r'}\right) \cap \bar{\mathbb{K}}[X]$$

   with

   - $U_r, W_{r'}$ are prime ideals;
   - $U_a \not\subseteq U_b$ and $W_{a'} \not\subseteq W_{b'}$ for any $a \neq b$ and $a' \neq b'$;
   - $(\forall r')(\exists r)\ \dim W_{r'} \leq \dim U_r$.

3. Dimension is finite. It cannot infinitely decrease.

$$(\exists n)\ PI_n = \bigcap I_{k+n} = \bigcap I_{k+n+1} = PI_{n+1}$$

Outline

Overview of the Method

Algebraic Techniques for Invariant Generation

Polynomial Invariants for Loops with Assignments Only

Polynomial Invariants for Loops with Conditionals

Examples

Conclusions

Extended Euclid's Algorithm for $GCD[x, y]$

Program

$a := x;\ b := y;$
$p := 1;\ r := 0;$
$q := 0;\ s := 1;$
$\underline{\text{while}}[(a \neq b),$
$\underline{\text{if}}[a > b$
$\quad \underline{\text{then}}\ a := a - b;$
$\quad\quad p := p - q;$
$\quad\quad r := r - s$
$\quad \underline{\text{else}}\ b := b - a;$
$\quad\quad q := q - p;$
$\quad\quad s := s - r]]$

Polynomial Invariants

$y = bp - aq$
$x = as - br$
$1 = ps - qr$
$a = px + ry$
$b = qx + sy$

Related Work

Work	Invariant Restrictions	Loop Restrictions	Complete
MOS, 2006	yes	no	no*
SSM, 2004	yes	no	no*
RCK, 2007a tool: Inv	yes	no	no*
RCK, 2007b tool: Solvable	no	yes*	yes
LK, 2008/09 tool: Aligator	no	yes	**yes**

Some Experimental Results wrt Solvable [RCK07b] and Inv [RCK07a]

		Timing	♯ Iters	♯ Polys
Binary Division	Aligator	**0.55 s**	**1**	**1**
	Solvable	1.78 s	3	**1**
	Inv	1.77 s	5	**1**
Euclid's Alg.	Aligator	9.02 s	**2**	**5**
	Solvable	**3.05 s**	5	**5**
	Inv	4.13 s	8	1
Fermat's Alg.	Aligator	**0.24 s**	**1**	**1**
	Solvable	1.73 s	4	**1**
	Inv	2.95 s	8	**1**
LCM-GCD	Aligator	**1.23 s**	**2**	**1**
	Solvable	2.01 s	5	**1**
	Inv	4.32 s	9	**1**
Binary Product	Aligator	**0.63 s**	**1**	**1**
	Solvable	1.74 s	4	**1**
	Inv	2.79 s	8	**1**
Square Root	Aligator	**0.19 s**	**1**	**2**
	Solvable	1.34 s	2	**2**
	Inv	2.17 s	6	**2**
Wensley's Alg.	Aligator	**0.63 s**	**1**	**3**
	Solvable	1.95 s	4	**3**
	Inv	3.53 s	8	**3**

Outline

Overview of the Method

Algebraic Techniques for Invariant Generation

Polynomial Invariants for Loops with Assignments Only

Polynomial Invariants for Loops with Conditionals

Examples

Conclusions

Conclusions

- Correct and complete algorithm: finds all polynomial invariants;
- Implementation: ALIGATOR
  1. Solving recurrences; } Symbolic Summation
  2. Computing algebraic dependencies;
  3. Eliminating non-program variables;
  4. Intersecting ideals; } Gröbner basis
- ALIGATOR successfully tried on many examples;

  Less time/iteration needed than other tools.

  http://mtc.epfl.ch/software-tools/Aligator/

End of Session 2

Slides for session 2 ended here . . .