Polynomial Invariant Generation for Multi-Path Loops

Andreas Humenberger

joint work with

Maximilian Jaroschek and Laura Kovács

SC^2 Workshop 2017

Kaiserslautern, July 29

Roadmap

- Why invariant generation?
- Single-path loops
- Multi-path loops

Verifying algorithms

Division of a by b where $b \neq 0$:

$quo := 0;$
$rem := a;$
while $b \leq rem$ **do**
  $rem := rem - b;$
  $quo := quo + 1;$
end while

Verifying algorithms

Division of a by b where $b \neq 0$:

$$
\begin{array}{l}
quo := 0; \\
rem := a; \\
\textbf{while } b \leq rem \textbf{ do} \\
\quad rem := rem - b; \\
\quad quo := quo + 1; \\
\textbf{end while}
\end{array}
$$

Question

- Is this algorithm correct?

Verifying algorithms

Division of a by b where $b \neq 0$:

$$
\begin{array}{l}
quo := 0; \\
rem := a; \\
\textbf{while } b \leq rem \textbf{ do} \\
\quad rem := rem - b; \\
\quad quo := quo + 1; \\
\textbf{end while}
\end{array}
$$

Question

- Is this algorithm correct?
- Does it terminate?

Loop invariants

Definition

A loop invariant $\mathcal{I}$ for the loop **while** $\mathcal{S}$ **do** B is an assertion that satisfies

$$\{\mathcal{I} \wedge \mathcal{S}\} B \{\mathcal{I}\}$$

Loop invariants

Definition

A loop invariant $\mathcal{I}$ for the loop **while** $\mathcal{S}$ **do** B is an assertion that satisfies

$$\{\mathcal{I} \wedge \mathcal{S}\}B\{\mathcal{I}\}$$

while $b \leq rem$ **do**
$rem := rem - b;$
$quo := quo + 1;$
end while

$$\{\overbrace{(quo \cdot b + rem - a = 0)}^{\mathcal{I}} \wedge \overbrace{(b \leq rem)}^{\mathcal{S}}\}$$
$$rem := rem - b;$$
$$quo := quo + 1;$$
$$\{\underbrace{(quo \cdot b + rem - a = 0)}_{\mathcal{I}}\}$$

Our programming model (for single-path loops)

while $\mathit{pred}(v_1, \ldots, v_m)$ **do**

$v_1 := f_1(v_1, \ldots, v_m)$

$\vdots$

$v_m := f_m(v_1, \ldots, v_m)$

end while

rational functions

Our programming model (for single-path loops)

while $\color{red}{pred(v_1, \ldots, v_m)}$ **do**

$v_1 := f_1(v_1, \ldots, v_m)$

$\vdots$

$v_m := f_m(v_1, \ldots, v_m)$

end while

(the f_i are rational functions)

Our programming model (for single-path loops)

while $pred(v_1, \ldots, v_m)$ **do**
$v_1 := f_1(v_1, \ldots, v_m)$
$\vdots$
$v_m := f_m(v_1, \ldots, v_m)$
end while

rational functions

Proposition (Müller-Olm and Seidl [2004])

The set of all equality invariants is not computable when considering affine equality tests.

Our programming model (for single-path loops)

while *true* **do**

$$
\left.\begin{array}{l} v_1 := f_1(v_1, \ldots, v_m) \\ \vdots \\ v_m := f_m(v_1, \ldots, v_m) \end{array}\right\} \text{rational functions}
$$

end while

Proposition (Müller-Olm and Seidl [2004])

The set of all equality invariants is not computable when considering affine equality tests.

Our programming model (for single-path loops)

while *true* **do**
$v_1 := f_1(v_1, \ldots, v_m)$
$\vdots$
$v_m := f_m(v_1, \ldots, v_m)$
end while

(rational functions)

Proposition (Müller-Olm and Seidl [2004])

The set of all equality invariants is not computable when considering affine equality tests.

$$\{\mathcal{I} \wedge \mathcal{S}\} B \{\mathcal{I}\}$$

Our programming model (for single-path loops)

while *true* **do**

$v_1 := f_1(v_1, \ldots, v_m)$

$\vdots$

$v_m := f_m(v_1, \ldots, v_m)$

end while

(rational functions)

Proposition (Müller-Olm and Seidl [2004])

The set of all equality invariants is not computable when considering affine equality tests.

$$\{\mathcal{I} \wedge \mathit{true}\} B \{\mathcal{I}\}$$

Our programming model (for single-path loops)

$$
\begin{array}{l}
\textbf{while } \textit{true} \textbf{ do} \\
\quad v_1 := f_1(v_1, \ldots, v_m) \\
\quad \vdots \\
\quad v_m := f_m(v_1, \ldots, v_m) \\
\textbf{end while}
\end{array}
$$

rational functions

Proposition (Müller-Olm and Seidl [2004])

The set of all equality invariants is not computable when considering affine equality tests.

$$\{\mathcal{I}\} B \{\mathcal{I}\}$$

Our programming model (for single-path loops)

while *true* **do**
$v_1 := f_1(v_1, \ldots, v_m)$
$\vdots$
$v_m := f_m(v_1, \ldots, v_m)$
end while

(rational functions)

Proposition (Müller-Olm and Seidl [2004])

The set of all equality invariants is not computable when considering affine equality tests.

$\{\mathcal{I}\} B^n \{\mathcal{I}\}$ for all $n \in \mathbb{N}$

Polynomial invariants

Definition

A polynomial $p \in \mathbb{K}[X]$ is a polynomial invariant of **while** $\mathcal{S}$ **do** B among the loop variables $V = v_1, \ldots, v_m$ with initial values V_0 if

$$\{p(V) = 0 \wedge V = V_0\}\ B^n\ \{p(V) = 0\} \qquad \text{for all } n \in \mathbb{N}.$$

Polynomial invariants

Definition

A polynomial $p \in \mathbb{K}[X]$ is a polynomial invariant of **while** $\mathcal{S}$ **do** B among the loop variables $V = v_1, \ldots, v_m$ with initial values V_0 if

$$\{p(V) = 0 \wedge V = V_0\} \; B^n \; \{p(V) = 0\} \qquad \text{for all } n \in \mathbb{N}.$$

Observation

The set of polynomial invariants is an ideal.

Definition

A subset $\mathcal{I} \subset \mathbb{K}[X]$ is an ideal if it satisfies:

(1) $0 \in \mathcal{I}$.

(2) If $f, g \in \mathcal{I}$, then $f + g \in \mathcal{I}$.

(3) If $f \in \mathcal{I}$ and $h \in \mathbb{K}[X]$, then $h \cdot f \in \mathcal{I}$.

Polynomial invariants

Definition

A polynomial $p \in \mathbb{K}[X]$ is a polynomial invariant of **while** $\mathcal{S}$ **do** B among the loop variables $V = v_1, \ldots, v_m$ with initial values V_0 if

$$\{p(V) = 0 \wedge V = V_0\} \; B^n \; \{p(V) = 0\} \qquad \text{for all } n \in \mathbb{N}.$$

Observation

The set of polynomial invariants is an ideal.

$\Rightarrow$ Symbolic Computation

Some facts about ideals

An ideal is generated by a set of elements:

$$\langle \underbrace{e_1, \ldots, e_n}_{basis} \rangle = \{ r_1 e_1 + \cdots + r_n e_n \mid r_i \in \mathbb{K}[X] \}$$

Some facts about ideals

An ideal is generated by a set of elements:

$$\underbrace{\langle e_1, \ldots, e_n \rangle}_{basis} = \{ r_1 e_1 + \cdots + r_n e_n \mid r_i \in \mathbb{K}[X] \}$$

Theorem

Every ideal $\mathcal{I} \lhd \mathbb{K}[X]$ has a finite basis.

Some facts about ideals

An ideal is generated by a set of elements:

$$\underbrace{\langle e_1, \ldots, e_n \rangle}_{basis} = \{r_1 e_1 + \cdots + r_n e_n \mid r_i \in \mathbb{K}[X]\}$$

Theorem

Every ideal $\mathcal{I} \lhd \mathbb{K}[X]$ has a finite basis.

Proposition

For $I, J \lhd \mathbb{K}[X, Y]$ we can compute

- $I + J$ *(sum)*
- $I \cap \mathbb{K}[X]$ *(elimination ideal; via Gröbner bases)*

Assignments and Recurrences

$x := 10$
$y := 10$
while $y > 0$ **do**
  $x \ := \ 2 \cdot x + 3$
  $y \ := \ {}^{1}\!/_{2} \cdot y - 1$
end while

Assignments and Recurrences

$x := 10$
$y := 10$
while $y > 0$ **do**
  $x := 2 \cdot x + 3$
  $y := 1/2 \cdot y - 1$
end while

$$\left.\begin{aligned} x_0 &= 10 \\ y_0 &= 10 \end{aligned}\right\} \quad \text{initial values}$$

$$\left.\begin{aligned} x_{n+1} &= 2 \cdot x_n + 3 \\ y_{n+1} &= 1/2 \cdot y_n - 1 \end{aligned}\right\} \quad \text{recurrences}$$

Assignments and Recurrences

$x := 10$
$y := 10$
while $y > 0$ **do**
  $x := 2 \cdot x + 3$
  $y := {}^{1}\!/_{2} \cdot y - 1$
end while

$$\left.\begin{aligned} x_0 &= 10 \\ y_0 &= 10 \end{aligned}\right\} \quad \text{initial values}$$

$$\left.\begin{aligned} x_{n+1} &= 2 \cdot x_n + 3 \\ y_{n+1} &= 1/2 \cdot y_n - 1 \end{aligned}\right\} \quad \text{recurrences}$$

$$\left.\begin{aligned} x_n &= 2^n \cdot (x_0 + 3) - 3 \\ y_n &= 2^{-n} \cdot (y_0 + 2) - 2 \end{aligned}\right\} \quad \text{closed forms}$$

Assignments and Recurrences

$x := 10$
$y := 10$
while $y > 0$ **do**
  $x := 2 \cdot x + 3$
  $y := {}^{1}\!/_{2} \cdot y - 1$
end while

$$\left.\begin{aligned} x_0 &= 10 \\ y_0 &= 10 \end{aligned}\right\} \quad \text{initial values}$$

$$\left.\begin{aligned} x_{n+1} &= 2 \cdot x_n + 3 \\ y_{n+1} &= 1/2 \cdot y_n - 1 \end{aligned}\right\} \quad \text{recurrences}$$

$$\left.\begin{aligned} x_n &= 2^{n} \cdot (x_0 + 3) - 3 \\ y_n &= 2^{-n} \cdot (y_0 + 2) - 2 \end{aligned}\right\} \quad \text{closed forms}$$

Loop counter elimination

$$x = 2^n \cdot (x_0 + 3) - 3$$
$$y = 2^{-n} \cdot (y_0 + 2) - 2$$

Loop counter elimination

$$x = a \cdot (x_0 + 3) - 3$$
$$y = b \cdot (y_0 + 2) - 2$$

Loop counter elimination

$$x = 2^n \cdot (x_0 + 3) - 3$$
$$y = 2^{-n} \cdot (y_0 + 2) - 2$$

$0 = 2^n \cdot 2^{-n} - 1$ algebraic dependency among $2^n, 2^{-n}$

Loop counter elimination

$$x = a \cdot (x_0 + 3) - 3$$
$$y = b \cdot (y_0 + 2) - 2$$

$0 = a \cdot b - 1$ algebraic dependency among $2^n, 2^{-n}$

Loop counter elimination

$$x = a \cdot (x_0 + 3) - 3$$
$$y = b \cdot (y_0 + 2) - 2$$

$0 = a \cdot b - 1$ algebraic dependency among $2^n, 2^{-n}$

$\mathcal{I} =$

Loop counter elimination

$$x = a \cdot (x_0 + 3) - 3$$
$$y = b \cdot (y_0 + 2) - 2$$

$0 = a \cdot b - 1$ algebraic dependency among $2^n, 2^{-n}$

$$\mathcal{I} = \langle x - a \cdot (x_0 + 3) + 3, y - b \cdot (y_0 + 2) + 2 \rangle$$

Loop counter elimination

$$x = a \cdot (x_0 + 3) - 3$$
$$y = b \cdot (y_0 + 2) - 2$$

$$0 = a \cdot b - 1$$

algebraic dependency among $2^n, 2^{-n}$

$$\mathcal{I} = \langle x - a \cdot (x_0 + 3) + 3, y - b \cdot (y_0 + 2) + 2 \rangle + \langle a \cdot b - 1 \rangle$$

Loop counter elimination

$$x = a \cdot (x_0 + 3) - 3$$
$$y = b \cdot (y_0 + 2) - 2$$

$$0 = a \cdot b - 1$$

algebraic dependency among $2^n, 2^{-n}$

$$\mathcal{I} = \langle x - a \cdot (x_0 + 3) + 3, y - b \cdot (y_0 + 2) + 2 \rangle + \langle a \cdot b - 1 \rangle$$

$$\mathcal{I} \lhd \mathbb{K}[x_0, y_0, x, y, a, b]$$

Loop counter elimination

$$x = a \cdot (x_0 + 3) - 3$$
$$y = b \cdot (y_0 + 2) - 2$$

$$0 = a \cdot b - 1$$ algebraic dependency among $2^n, 2^{-n}$

$$\mathcal{I} = \langle x - a \cdot (x_0 + 3) + 3, y - b \cdot (y_0 + 2) + 2 \rangle + \langle a \cdot b - 1 \rangle$$

$$\mathcal{I} \lhd \mathbb{K}[x_0, y_0, x, y, a, b]$$

$\Rightarrow$ Gröbner bases to the rescue

Loop counter elimination

$$x = a \cdot (x_0 + 3) - 3$$
$$y = b \cdot (y_0 + 2) - 2$$

$0 = a \cdot b - 1$ algebraic dependency among $2^n, 2^{-n}$

$$\mathcal{I} = \langle x - a \cdot (x_0 + 3) + 3, y - b \cdot (y_0 + 2) + 2 \rangle + \langle a \cdot b - 1 \rangle$$
$$\mathcal{I} \lhd \mathbb{K}[x_0, y_0, x, y, a, b]$$

$\Rightarrow$ Gröbner bases to the rescue

$$\mathcal{I} \cap \mathbb{K}[x_0, y_0, x, y]$$

Loop counter elimination

$$x = a \cdot (x_0 + 3) - 3$$
$$y = b \cdot (y_0 + 2) - 2$$
$$0 = a \cdot b - 1$$

algebraic dependency among $2^n, 2^{-n}$

$$\mathcal{I} = \langle x - a \cdot (x_0 + 3) + 3, y - b \cdot (y_0 + 2) + 2 \rangle + \langle a \cdot b - 1 \rangle$$
$$\mathcal{I} \lhd \mathbb{K}[x_0, y_0, x, y, a, b]$$

$\Rightarrow$ Gröbner bases to the rescue

$$\mathcal{I} \cap \mathbb{K}[x_0, y_0, x, y] = \langle -2x + 2x_0 - 3y - xy + 3y_0 + x_0 y_0 \rangle$$

(Extended) P-solvable loops

loops with solvable mappings
(Rodríguez-Carbonell and Kapur [2004])

P-solvable loops (Kovács [2007])

    C-finite sequences

extended P-solvable loops (ISSAC'17)

    subsumes C-finite, rational function and hypergeometric sequences

(Extended) P-solvable loops

loops with solvable mappings
(Rodríguez-Carbonell and Kapur [2004])

P-solvable loops (Kovács [2007])

- C-finite sequences

extended P-solvable loops (ISSAC'17)

- subsumes C-finite, rational function and hypergeometric sequences

$$(\mathcal{C} + \mathcal{A}) \cap \mathbb{K}[X_0, X]$$

Multi-path loops

while b **do**
  if b_1 **then** A_1
    $\vdots$
  if b_m **then** A_m
  while b_{m+1} **do** B_{m+1}
    $\vdots$
  while b_n **do** B_n
end while

Multi-path loops

while b **do**
  if b_1 **then** A_1
    ⋮
  if b_m **then** A_m
  while b_{m+1} **do** B_{m+1}
    ⋮
  while b_n **do** B_n
end while

Multi-path loops

while b **do**
  while $b_1 \wedge f_1$ **do** B_1
    $\vdots$
  while $b_m \wedge f_m$ **do** B_m
  while b_{m+1} **do** B_{m+1}
    $\vdots$
  while b_n **do** B_n
end while

Multi-path loops

while b **do**
  while $b_1 \wedge f_1$ **do** B_1
    $\vdots$
  while $b_m \wedge f_m$ **do** B_m
  while b_{m+1} **do** B_{m+1}
    $\vdots$
  while b_n **do** B_n
end while

Multi-path loops

while b **do**
  while $b_1 \wedge f_1$ **do** B_1
    $\vdots$
  while $b_m \wedge f_m$ **do** B_m
  while b_{m+1} **do** B_{m+1}
    $\vdots$
  while b_n **do** B_n
end while

Multi-path loops

while ... **do**
  while ... **do** B_1
    ⋮
  while ... **do** B_m
  while ... **do** B_{m+1}
    ⋮
  while ... **do** B_n
end while

Multi-path loops

while ... **do**
  while ... **do** B_1
    $\vdots$
  while ... **do** B_m
  while ... **do** B_{m+1}
    $\vdots$
  while ... **do** B_n
end while

B_1^*

Multi-path loops

while ... **do**
  while ... **do** B_1
    ⋮
  while ... **do** B_m
  while ... **do** B_{m+1}
    ⋮
  while ... **do** B_n
end while

$B_1^*; B_2^*$

Multi-path loops

while … **do**
  while … **do** B_1
    ⋮
  while … **do** B_m
  while … **do** B_{m+1}
    ⋮
  while … **do** B_n
end while

$$B_1^*; B_2^*; \ldots; B_n^*$$

Multi-path loops

while ... **do**
  while ... **do** B_1
    $\vdots$
  while ... **do** B_m
  while ... **do** B_{m+1}
    $\vdots$
  while ... **do** B_n
end while

$$(B_1^*; B_2^*; \ldots; B_n^*)^*$$

Multi-path loops

while ... **do**
  while ... **do** B_1
    ⋮
  while ... **do** B_m
  while ... **do** B_{m+1}
    ⋮
  while ... **do** B_n
end while

$$(B_1^*; B_2^*; \ldots; B_n^*)^*$$

Compute invariant ideal of

$B_1^*; \ldots; B_n^*$

Multi-path loops

while ... **do**
  while ... **do** B_1
    $\vdots$
  while ... **do** B_m
  while ... **do** B_{m+1}
    $\vdots$
  while ... **do** B_n
end while

$$(B_1^*; B_2^*; \ldots; B_n^*)^*$$

Compute invariant ideal of

$B_1^*; \ldots; B_n^*$

$B_1^*; \ldots; B_n^*; B_1^*; \ldots; B_n^*$

Multi-path loops

while ... **do**
  while ... **do** B_1
    ⋮
  while ... **do** B_m
  while ... **do** B_{m+1}
    ⋮
  while ... **do** B_n
end while

$$(B_1^*; B_2^*; \ldots; B_n^*)^*$$

Compute invariant ideal of

$$B_1^*; \ldots; B_n^*$$
$$B_1^*; \ldots; B_n^*; B_1^*; \ldots; B_n^*$$
$$B_1^*; \ldots; B_n^*; B_1^*; \ldots; B_n^*; B_1^*; \ldots; B_n^*$$

Multi-path loops

while ... **do**
  while ... **do** B_1
    $\vdots$
  while ... **do** B_m
  while ... **do** B_{m+1}
    $\vdots$
  while ... **do** B_n
end while

$$(B_1^*; B_2^*; \ldots; B_n^*)^*$$

Compute invariant ideal of

$B_1^*; \ldots; B_n^*$

$B_1^*; \ldots; B_n^*; B_1^*; \ldots; B_n^*$

$B_1^*; \ldots; B_n^*; B_1^*; \ldots; B_n^*; B_1^*; \ldots; B_n^*$

$\vdots$

until a fixed-point is reached

Multi-path loops

while ... **do**
  while ... **do** B_1
    $\vdots$
  while ... **do** B_m
  while ... **do** B_{m+1}
    $\vdots$
  while ... **do** B_n
end while

$$(B_1^*; B_2^*; \ldots; B_n^*)^*$$

Compute invariant ideal of

$B_1^*; \ldots; B_n^*$

$B_1^*; \ldots; B_n^*; B_1^*; \ldots; B_n^*$

$B_1^*; \ldots; B_n^*; B_1^*; \ldots; B_n^*; B_1^*; \ldots; B_n^*$

$\vdots$

until a fixed-point is reached

Fixed-point guaranteed

Multi-path loops

while ... **do**
  while ... **do** B_1
    $\vdots$
  while ... **do** B_m
  while ... **do** B_{m+1}
    $\vdots$
  while ... **do** B_n
end while

$$(B_1^*; B_2^*; \ldots; B_n^*)^*$$

Compute invariant ideal of

$B_1^*; \ldots; B_n^*$

$B_1^*; \ldots; B_n^*; B_1^*; \ldots; B_n^*$

$B_1^*; \ldots; B_n^*; B_1^*; \ldots; B_n^*; B_1^*; \ldots; B_n^*$

$\vdots$

until a fixed-point is reached

Fixed-point guaranteed

Bound: $k+1$
for k loop vars

Merging loops

B_1^*

B_2^*

Merging loops

initial values
B_1^*

initial values
B_2^*

Merging loops

initial values
B_1^*
final values

initial values
B_2^*
final values

Merging loops

initial values
B_1^*
final values
initial values
B_2^*
final values

Merging loops

initial values
B_1^*
final values

initial values
B_2^*
final values

$$B_1^* \begin{cases} x_1 & = 2^m \cdot (x_0 + 3) - 3 \\ y_1 & = 2^{-m} \cdot (y_0 + 2) - 2 \end{cases}$$

$$B_2^* \begin{cases} x_2 & = 2^n \cdot (x_1 + 3) - 3 \\ y_2 & = 2^{-n} \cdot (y_1 + 2) - 2 \end{cases}$$

Merging loops

initial values
B_1^*
final values

initial values
B_2^*
final values

$$\mathcal{C}_1 \begin{cases} x_1 & = a \cdot (x_0 + 3) - 3 \\ y_1 & = b \cdot (y_0 + 2) - 2 \end{cases}$$

$$\mathcal{C}_2 \begin{cases} x_2 & = c \cdot (x_1 + 3) - 3 \\ y_2 & = d \cdot (y_1 + 2) - 2 \end{cases}$$

Merging loops

initial values
B_1^*
final values
initial values
B_2^*
final values

$$\mathcal{C}_1 \begin{cases} x_1 &= a \cdot (x_0 + 3) - 3 \\ y_1 &= b \cdot (y_0 + 2) - 2 \end{cases}$$

$$\mathcal{A}_1 \begin{cases} 0 &= a \cdot b - 1 \end{cases}$$

$$\mathcal{C}_2 \begin{cases} x_2 &= c \cdot (x_1 + 3) - 3 \\ y_2 &= d \cdot (y_1 + 2) - 2 \end{cases}$$

$$\mathcal{A}_2 \begin{cases} 0 &= c \cdot d - 1 \end{cases}$$

Merging loops

initial values
B_1^*
final values
initial values
B_2^*
final values

$$\mathcal{C}_1 \begin{cases} x_1 & = a \cdot (x_0 + 3) - 3 \\ y_1 & = b \cdot (y_0 + 2) - 2 \end{cases}$$

$$\mathcal{A}_1 \begin{cases} 0 & = a \cdot b - 1 \end{cases}$$

$$\mathcal{C}_2 \begin{cases} x_2 & = c \cdot (x_1 + 3) - 3 \\ y_2 & = d \cdot (y_1 + 2) - 2 \end{cases}$$

$$\mathcal{A}_2 \begin{cases} 0 & = c \cdot d - 1 \end{cases}$$

$$(\mathcal{C}_1 + \mathcal{A}_1 + \mathcal{C}_2 + \mathcal{A}_2) \cap \mathbb{K}[x_0, y_0, x_2, y_2]$$

Merging loops

initial values
B_1^*
final values
initial values
B_2^*
final values

$$\mathcal{C}_1 \begin{cases} x_1 & = a \cdot (x_0 + 3) - 3 \\ y_1 & = b \cdot (y_0 + 2) - 2 \end{cases}$$

$$\mathcal{A}_1 \begin{cases} 0 & = a \cdot b - 1 \end{cases}$$

$$\mathcal{C}_2 \begin{cases} x_2 & = c \cdot (x_1 + 3) - 3 \\ y_2 & = d \cdot (y_1 + 2) - 2 \end{cases}$$

$$\mathcal{A}_2 \begin{cases} 0 & = c \cdot d - 1 \end{cases}$$

$$(\mathcal{C}_1 + \mathcal{A}_1 + \mathcal{C}_2 + \mathcal{A}_2) \cap \mathbb{K}[x_0, y_0, x_2, y_2]$$

$$= (\underbrace{\mathcal{I}_1}_{\text{invariant ideal of first loop}} + \mathcal{C}_2 + \mathcal{A}_2) \cap \mathbb{K}[x_0, y_0, x_2, y_2]$$

Conclusion

- Fixed-point computation

Conclusion

- Fixed-point computation
- most probably terminates

Conclusion

- Fixed-point computation
- most probably terminates
- bound on the number of the iterations

Conclusion

- Fixed-point computation
- most probably terminates
- bound on the number of the iterations
- combination with other algorithms

Conclusion

- Fixed-point computation
- most probably terminates
- bound on the number of the iterations
- combination with other algorithms

ALIGATOR Mathematica package
Available at `https://ahumenberger.github.io/aligator/`

(Extended) P-solvable loops

Definition (Kovács [2007])

A loop with assignments only is called P-solvable if the closed forms of its recursively changed variables $x_1, \dots, x_m$ are of the form

$$x_i(n) = p_{i,1}(n)\theta_1^n + \cdots + p_{i,s}(n)\theta_s^n$$

(Extended) P-solvable loops

Definition (Kovács [2007])

A loop with assignments only is called P-solvable if the closed forms of its recursively changed variables $x_1, \ldots, x_m$ are of the form

$$x_i(n) = p_{i,1}(n)\theta_1^n + \cdots + p_{i,s}(n)\theta_s^n$$

(Extended) P-solvable loops

Definition (Kovács [2007])

A loop with assignments only is called P-solvable if the closed forms of its recursively changed variables $x_1, \ldots, x_m$ are of the form

$$x_i(n) = p_{i,1}(n)\theta_1^n + \cdots + p_{i,s}(n)\theta_s^n$$

(Extended) P-solvable loops

Definition (Kovács [2007])

A loop with assignments only is called P-solvable if the closed forms of its recursively changed variables $x_1, \ldots, x_m$ are of the form

$$x_i(n) = p_{i,1}(n)\theta_1^n + \cdots + p_{i,s}(n)\theta_s^n$$

Definition (ISSAC'17)

A loop with assignments only is called extended P-solvable if the closed forms of its recursively changed variables $x_1, \ldots, x_m$ are of the form

$$v_i(n) = \qquad p_{i,k}(n, \theta_1^n, \ldots, \theta_s^n)$$

(Extended) P-solvable loops

Definition (Kovács [2007])

A loop with assignments only is called P-solvable if the closed forms of its recursively changed variables $x_1, \dots, x_m$ are of the form

$$x_i(n) = p_{i,1}(n)\theta_1^n + \cdots + p_{i,s}(n)\theta_s^n$$

Definition (ISSAC'17)

A loop with assignments only is called extended P-solvable if the closed forms of its recursively changed variables $x_1, \dots, x_m$ are of the form

$$v_i(n) = \qquad p_{i,k}(n, \theta_1^n, \dots, \theta_s^n)((n+\zeta_1)^{\underline{n}})^{k_1} \cdots ((n+\zeta_\ell)^{\underline{n}})^{k_\ell}$$

(Extended) P-solvable loops

Definition (Kovács [2007])

A loop with assignments only is called P-solvable if the closed forms of its recursively changed variables $x_1, \ldots, x_m$ are of the form

$$x_i(n) = p_{i,1}(n)\theta_1^n + \cdots + p_{i,s}(n)\theta_s^n$$

Definition (ISSAC'17)

A loop with assignments only is called extended P-solvable if the closed forms of its recursively changed variables $x_1, \ldots, x_m$ are of the form

$$v_i(n) = \sum_{k \in \mathbb{Z}^\ell} p_{i,k}(n, \theta_1^n, \ldots, \theta_s^n)((n+\zeta_1)^{\underline{n}})^{k_1} \cdots ((n+\zeta_\ell)^{\underline{n}})^{k_\ell}$$