ATVA 2023 · 25 October 2023

pymwp: A Static Analyzer Determining Polynomial Growth Bounds

Clément Aubert, Thomas Rubiano, **Neea Rusch** and Thomas Seiller

```
void main(int X1, int X2, int X3)
{
  while(X1 < 10){
     X1 = X2 + X3;
  }
  // X1'  X2'  X3'
}
```

For input variables of an imperative program:

Does there exist a polynomially bounded data-flow relation between variables' **initial** values and **final** values?

That is, $\forall n$, is $X_n \rightsquigarrow X'_n$ polynomially bounded?

```
void main(int X1, int X2, int X3)
{
  while(X1 < 10){
     X1 = X2 + X3;
  }
  // X1'  X2'  X3'
}
```

Yes. Here is a bound:

$$X1' \leq \max(X1, X2+X3)$$

$$\wedge\ X2' \leq X2$$

$$\wedge\ X3' \leq X3$$

mwp-flow analysis[1]

Calculus for resource analysis of imperative programs.

```
while(X1 < 10)
  X1 = X2 + X3
```

0 – no dependency

m – maximal (of linear)

w – weak polynomial

p - polynomial

[1]Neil D. Jones and Lars Kristiansen. “A flow calculus of mwp-bounds for complexity analysis”. In: ACM Trans. Comput. Log. 10.4 (Aug. 2009), 28:1–28:41. doi: 10.1145/1555746.1555752.

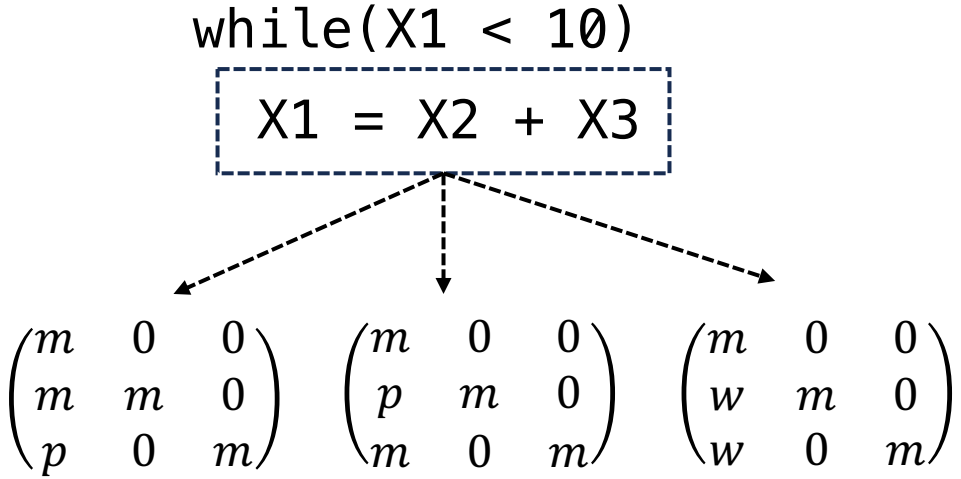
mwp-flow analysis[1]

Many nice properties

- Sound syntactic calculus
- Compositional …

Challenges

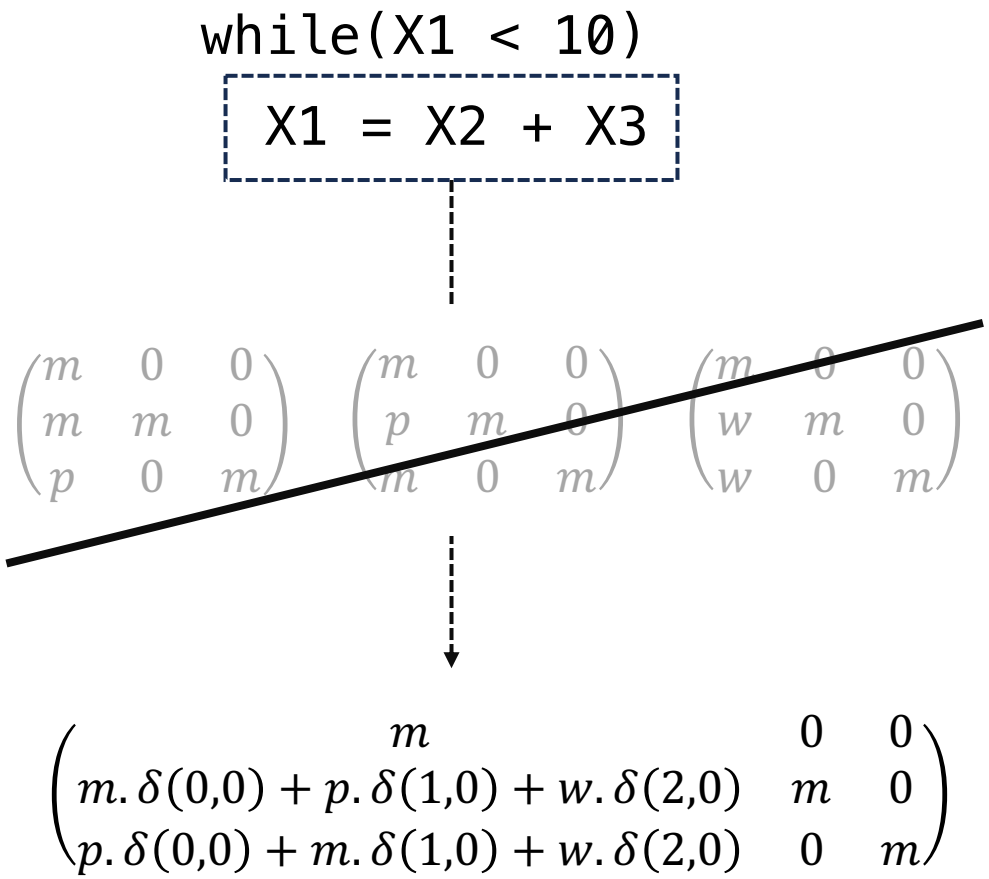
- Non-determinism
- Derivation failure



[1]Neil D. Jones and Lars Kristiansen. “A flow calculus of mwp-bounds for complexity analysis”. In: ACM Trans. Comput. Log. 10.4 (Aug. 2009), 28:1–28:41. doi: 10.1145/1555746.1555752.

Automating mwp

Handling non-determinism



Aubert, Clément, et. al. “mwp-Analysis Improvement and Implementation: Realizing Implicit Computational Complexity”. In: FSCD 2022. Vol. 228. LIPIcs, 2022, 26:1–26:23. doi: 10.4230/LIPIcs.FSCD.2022.26.

Automating mwp

Handling derivation failure

0 – no dependency

m – maximal (of linear)

w – weak polynomial

p - polynomial

∞ - non-polynomial (failure)

```
while(X1 < 10)
   X1 = X2 + X3
```

$$\begin{pmatrix} m + \infty.\delta(0,0) + \infty.\delta(1,0) & 0 & 0 \\ \infty.\delta(0,0) + \infty.\delta(1,0) + w.\delta(2,0) & m & 0 \\ \infty.\delta(0,0) + \infty.\delta(1,0) + w.\delta(2,0) & 0 & m \end{pmatrix}$$

Aubert, Clément, et. al. “mwp-Analysis Improvement and Implementation: Realizing Implicit Computational Complexity”. In: FSCD 2022. Vol. 228. LIPIcs, 2022, 26:1–26:23. doi: 10.4230/LIPIcs.FSCD.2022.26.

```
void main(int X1, int X2, int X3)
{
  while(X1 < 10){
     X1 = X2 + X3;
  }
  // X1'  X2'  X3'
}
```

We were here

$$\begin{pmatrix} m + \infty.\delta(0,0) + \infty.\delta(1,0) & 0 & 0 \\ \infty.\delta(0,0) + \infty.\delta(1,0) + w.\delta(2,0) & m & 0 \\ \infty.\delta(0,0) + \infty.\delta(1,0) + w.\delta(2,0) & 0 & m \end{pmatrix}$$

But we *actually* want

$X1' \leq \max(X1, X2+X3)$
$\wedge\ X2' \leq X2\ \wedge\ X3' \leq X3$

Obtaining bounds compactly and efficiently

$$\begin{pmatrix} m + \infty.\delta(0,0) + \infty.\delta(1,0) & 0 & 0 \\ \infty.\delta(0,0) + \infty.\delta(1,0) + w.\delta(2,0) & m & 0 \\ \infty.\delta(0,0) + \infty.\delta(1,0) + w.\delta(2,0) & 0 & m \end{pmatrix}$$

Find the derivation paths that lead to ∞, then simplify.

[{~~∅~~,1,2}, {0,1,2}, {0,1,2}]

[{2}, {0,1,2}, {0,1,2}]

Build "choice vectors".

```
void main(int X1)
{
  while(X1 < 10){
     X1 = X1 * X1;
  }
}
```

When derivation fails

Problematic flows: X1 ➔ X1

pymwp is an automatic static analyzer for C code,
to determine if inputs' value growth is polynomially-bounded.

run in terminal

```
pip install pymwp

pymwp file.c  [ARGS]
```

run in browser

`statycc.github.io/pymwp/demo`

source code + docs: **`statycc/pymwp`**