mwp-Analysis Improvement and Implementation: Realizing Implicit Computational Complexity

Clément Aubert[1], Thomas Rubiano[2], <u>Neea Rusch</u>[1], Thomas Seiller[2,3]

[1] Augusta University
[2] LIPN - Laboratoire d'Informatique de Paris-Nord
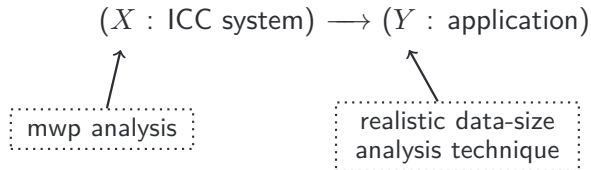[3] CNRS - Centre National de la Recherche Scientifique

10 March 2023

Earlier: **I**mplicit **C**omputational **C**omplexity $R \subseteq L = C$
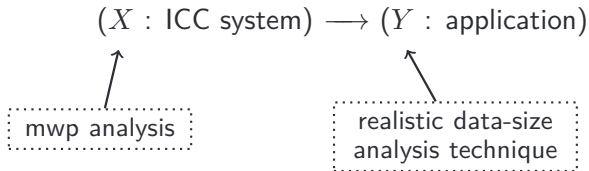
Earlier: **I**mplicit **C**omputational **C**omplexity $R \subseteq L = C$

$$(X : \text{ICC system}) \longrightarrow (Y : \text{application})$$

Earlier: **I**mplicit **C**omputational **C**omplexity $R \subseteq L = C$

$$(X : \text{ICC system}) \longrightarrow (Y : \text{application})$$

mwp analysis

realistic data-size
analysis technique

Earlier: **I**mplicit **C**omputational **C**omplexity $\qquad R \subseteq L = C$

$$(X : \text{ICC system}) \longrightarrow (Y : \text{application})$$

mwp analysis

realistic data-size analysis technique

mwp-analysis $\longrightarrow$ mwp-analysis$'$ $\overset{*}{\longrightarrow}$ realistic analysis

> The goal is to discover a polynomially bounded data-flow relation between command C, initial values x_i, and final values x_i': $[\![C]\!](x_i \leadsto x_i')$.

$C' \equiv$ `X1 := X2 + X3;`
　　　`X1 := X1 + X1`

$[\![C']\!](x_1, x_2, x_3 \leadsto x_1', x_2', x_3')$

$x_1' \leq 2x_2 + 2x_3$

$x_2' \leq x_2$

$x_3' \leq x_3$

$C'' \equiv$ `X1 := 1;`
　　　`loop X2 {X1 := X1 + X1}`

$[\![C'']\!](x_1, x_2 \leadsto x_1', x_2')$

$x_1' \leq 2^{x_2}$

$x_2' \leq x_2$

mwp analysis[1]

Method for certifying that values computed by a deterministic imperative program will be bounded by polynomials in the program's inputs.

$$\texttt{C} : \text{program} \qquad\qquad M : \text{matrix} \qquad\qquad \vdash \texttt{C} : M$$

---

[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.

Language

(var) $X_1 \mid X_2 \mid X_3 \mid \ldots$ (aexp)   `e + e | e * e`   (bexp)   `e = e | e < e | ...`

(com)   `skip | X := e | C;C | if b then C else C | loop X {C} | while b do {C}`

Dependencies ("flows") $\xrightarrow{\text{stronger}}$

0 : no dependency m : maximal w : weak polynomial p : polynomial

Inference rules   **mwp-bound**

$$\frac{}{\vdash_{\text{JK}} \mathtt{Xi} : \{^m_i\}} \;\; \text{E1} \qquad \frac{\vdash_{\text{JK}} \mathtt{Xi} : V_1 \quad \vdash_{\text{JK}} \mathtt{Xj} : V_2}{\vdash_{\text{JK}} \mathtt{Xi \star Xj} : pV_1 \oplus V_2} \;\; \text{E3} \qquad \frac{\vdash \mathtt{e} : V}{\vdash \mathtt{Xj = e} : 1 \xleftarrow{\mathtt{j}} V} \;\; \text{A} \ldots \qquad \max(\vec{x}, poly_1(\vec{y})) + poly_2(\vec{z})$$

$$\dfrac{\dfrac{}{\vdash_{\text{JK}} \text{X2} : \left(\begin{smallmatrix} 0 \\ m \\ 0 \end{smallmatrix}\right)} \ \text{E1} \quad \dfrac{}{\vdash_{\text{JK}} \text{X3} : \left(\begin{smallmatrix} 0 \\ 0 \\ m \end{smallmatrix}\right)} \ \text{E1}}{\dfrac{\vdash_{\text{JK}} \text{X2+X3} : \left(\begin{smallmatrix} 0 \\ p \\ m \end{smallmatrix}\right)}{\vdash_{\text{JK}} \text{X1:=X2+X3} : \left(\begin{smallmatrix} 0 & 0 & 0 \\ p & m & 0 \\ m & 0 & m \end{smallmatrix}\right)} \ \text{A}} \ \text{E3}$$

$$\vdots$$

$$\dfrac{}{\vdash_{\text{JK}} \text{X1:=X1+X1} : \left(\begin{smallmatrix} p & 0 & 0 \\ 0 & m & 0 \\ 0 & 0 & m \end{smallmatrix}\right)} \ \text{A}$$

$$\vdots$$

$$C' \equiv \begin{array}{l} \text{X1} \ := \ \text{X2} \ + \ \text{X3}; \\ \text{X1} \ := \ \text{X1} \ + \ \text{X1} \end{array}$$

$$\dfrac{}{\vdash_{\text{JK}} \text{X1:=X2+X3;X1:=X1+X1} : \left(\begin{smallmatrix} 0 & 0 & 0 \\ p & m & 0 \\ p & 0 & m \end{smallmatrix}\right)} \ \text{C}$$

$$x_1' \leq W_1(;; x_2, x_3) \ \wedge \ x_2' \leq W_2(x_2) \ \wedge \ x_3' \leq W_3(x_3)$$

$$\dfrac{}{\vdash_{\text{JK}} \text{X1:=1} : \left(\begin{smallmatrix} m \\ 0 \end{smallmatrix} \right)} \ \text{E1}$$

$$\vdots$$

$$C'' \equiv \begin{array}{l} \text{X1 := 1;} \\ \text{loop X2 \{X1 := X1 + X1\}} \end{array} \qquad\qquad \dfrac{}{\vdash_{\text{JK}} \text{X1:=X1+X1} : \left(\begin{smallmatrix} p & 0 \\ 0 & m \end{smallmatrix} \right)} \ \text{A}$$

$$\vdots$$

$$\times$$

$$\forall i, M_{ii}^* = m \ \dfrac{\vdash_{\text{JK}} \text{C} : M}{\vdash_{\text{JK}} \texttt{loop X1 \{C\}} : M^* \oplus \{ \begin{smallmatrix} p \\ 1 \end{smallmatrix} \to j \mid \exists i, M_{ij}^* = p \}} \ \text{L}$$

Theorem: Soundness[2]

$\vdash$ C : M implies $\vDash$ C : M

$\vdash$ C : M means the calculus *assigns* the matrix M to command C.

Relation $\vdash$ C : M holds iff there exists a derivation in the calculus.

Command C is *derivable* if the calculus assigns at least one matrix to it.

---

[2] Jones and Kristiansen, "A flow calculus of *mwp*-bounds for complexity analysis", p. 11.

mwp overview

Properties
Compositional, language-agnostic, multi-variate result, focus on value
growth, avoids termination and iteration-bounds analysis, . . .

Open questions
Richer languages? Expressiveness?

Challenges
Nondeterminism, derivation failure.

Nondeterminism

$$\frac{}{\vdash_{\mathrm{JK}} \mathtt{Xi} : \{^m_i\}} \ \mathsf{E1}$$

$$\frac{}{\vdash_{\mathrm{JK}} \mathtt{e} : \{^w_i | \ \mathtt{Xi} \in \mathrm{var}(\mathtt{e})\}} \ \mathsf{E2}$$

$$\frac{\vdash_{\mathrm{JK}} \mathtt{Xi} : V_1 \quad \vdash_{\mathrm{JK}} \mathtt{Xj} : V_2}{\vdash_{\mathrm{JK}} \mathtt{Xi \star Xj} : pV_1 \oplus V_2} \ \mathsf{E3}$$

$$\frac{\vdash_{\mathrm{JK}} \mathtt{Xi} : V_1 \quad \vdash_{\mathrm{JK}} \mathtt{Xj} : V_2}{\vdash_{\mathrm{JK}} \mathtt{Xi \star Xj} : V_1 \oplus pV_2} \ \mathsf{E4}$$

X2 + X3 has 3 derivations:

by (E2) $\quad \begin{pmatrix} 0 \\ w \\ w \end{pmatrix}$

by (E1) and (E3) $\quad \begin{pmatrix} 0 \\ p \\ m \end{pmatrix}$

by (E1) and (E4) $\quad \begin{pmatrix} 0 \\ m \\ p \end{pmatrix}$

In general: n choices yields 3^n derivations.

Improvement

Idea: internalize the choices as functions from choices to coefficients.

If a coefficient depends on a choice, represent as 3 elements (think $\{0, 1, 2\}^n$)

If independent, represented as a single element.

We also define basic functions $\delta(i, j)$ where i is a value, and j is index of the domain. If j^{th} input is equal to i, then (i, j) will be equal to the unit of the mwp semi-ring, and 0 otherwise.

$$\star \in \{+, -\} \ \frac{}{\vdash \texttt{Xi} \star \texttt{Xj} : (0 \mapsto \{\underset{\texttt{i}}{m}, \underset{\texttt{j}}{p}\}) \oplus (1 \mapsto \{\underset{\texttt{i}}{p}, \underset{\texttt{j}}{m}\}) \oplus (2 \mapsto \{\underset{\texttt{i}}{w}, \underset{\texttt{j}}{w}\})} \ \mathsf{E}^A$$

Comparison

Example 1. A 6-variable program with 2 assignments:

Original: $(6 \times 6) \times 3^2 = 324$ coefficients
Improved: 6 polynomials $+$ 30 simple values $= 66$ coefficients

Example 2. `C ≡ X1 := X2 + X3`

$$\begin{pmatrix} 0 & 0 & 0 \\ w & m & 0 \\ w & 0 & m \end{pmatrix} \quad \begin{pmatrix} 0 & 0 & 0 \\ p & m & 0 \\ m & 0 & m \end{pmatrix} \quad \begin{pmatrix} 0 & 0 & 0 \\ m & m & 0 \\ p & 0 & m \end{pmatrix} \quad \Rightarrow \quad \begin{pmatrix} 0 & 0 & 0 \\ m\delta(0,0)+p\delta(1,0)+w\delta(2,0) & m & 0 \\ p\delta(0,0)+m\delta(1,0)+w\delta(2,0) & 0 & m \end{pmatrix}$$

The Failure Problem

$$C \equiv \texttt{while(b){X1:=X2+X2;}}$$

Derivation of X1:=X2+X2 yields 2 matrices: $\left(\begin{smallmatrix} 0 & 0 \\ p & m \end{smallmatrix}\right)$ and $\left(\begin{smallmatrix} 0 & 0 \\ w & m \end{smallmatrix}\right)$

$$\forall i, M_{ii}^* = m \text{ and } \forall i, j, M_{ij}^* \neq p \ \frac{\vdash_{\text{JK}} \texttt{C} : M}{\vdash_{\text{JK}} \texttt{while b do \{C\}} : M^*} \ \text{W}$$

$\Rightarrow$ derivation $\left(\begin{smallmatrix} 0 & 0 \\ p & m \end{smallmatrix}\right)$ fails, but derivation $\left(\begin{smallmatrix} 0 & 0 \\ w & m \end{smallmatrix}\right)$ succeeds.

Representing Failure

Idea: We introduce ∞ flow to represent non-polynomial dependencies.

$$\{0, m, w, p, \infty\}$$

Every derivation can be completed without restarts.

Captures localized information about where failure occurs.

Once failure is introduced, it cannot be erased i.e., $\infty \times^\infty 0 = \infty$.

$$C \equiv \texttt{while(b)\{X1:=X2+X2;\}} \qquad \begin{pmatrix} m+\infty\delta(0,0)+\infty\delta(1,0) & 0 \\ \infty\delta(0,0)+\infty\delta(1,0)+w\delta(2,0) & m \end{pmatrix}$$

Apart from ∞ coefficients, the original and adjusted mwp systems agree.
The latter also provides a tractable technique: better proof-search strategy,
fine-grained feedback, etc.

Asking more specific questions, for some program C:

1. Does a bound exists?

2. If yes, what is the concrete mwp-bound?

3. If no, where does failure occur?

Apart from ∞ coefficients, the original and adjusted mwp systems agree. The latter also provides a tractable technique: better proof-search strategy, fine-grained feedback, etc.

Asking more specific questions, for some program `C`:

1. Does a bound exists? $\longrightarrow$ Delta graph

2. If yes, what is the concrete mwp-bound? $\longrightarrow$ Efficient evaluation

3. If no, where does failure occur? $\longrightarrow$ from matrix

Delta graph enables decoupling computation of *existence* of bounds and computing its values.
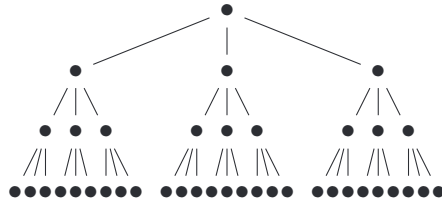
Delta graph enables decoupling computation of *existence* of bounds and computing its values.
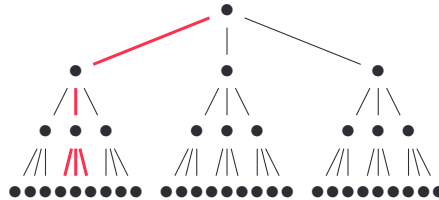
Evaluation

$$\texttt{C} \equiv \texttt{while(b)\{X1:=X2+X2;\}} \qquad \begin{pmatrix} m+\infty\delta(0,0)+\infty\delta(1,0) & 0 \\ \infty\delta(0,0)+\infty\delta(1,0)+w\delta(2,0) & m \end{pmatrix}$$

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

Matrix size depends on number of variables V^2 and n assignments introduces 3^n choices.

Challenge: How to *efficiently* determine and represent valid choices?

1. Construct a set S of all the δ values with ∞ coefficient in the matrix.

```
S = {[(0,0),(2,1)],
     [(1,0),(2,1)],
     [(2,0),(2,1)],
     [(0,0),(2,2)],
     [(0,0),(1,1)],
     [(1,1)]}
```

$\Rightarrow$

```
S = {[(2,1)],
     [(0,0),(2,2)],
     [(1,1)]}
```

2. Simplify S.

3. Construct choice vectors.

`[(2,1)] × [(0,0),(2,2)] × [(1,1)]`

initially: `[[0,1,2],[0,1,2],[0,1,2]]`

`(2,1)(0,0)(1,1) ⇒ [[1,2],[0],[0,1,2]]`
`(2,1)(2,2)(1,1) ⇒ [[0,1,2],[0],[0,1]]`

4. Result is a disjunction of choice vectors.

`[[1,2],[0],[0,1,2]] ∨ [[0,1,2],[0],[0,1]]`

E.g. (1 0 0) and (2 0 1) are valid derivation choices.

Compositionality

Compositional analysis enables computing result once then reusing the result it in the future.

- Analysis can be performed on *parts* of source code.

- It is possible to analyze a function, then save the result.

- Previously analyzed result can be reused at next execution.

- Expensive computation needs to be carried out once.

Extending the Syntax

Let f be a function with one output value,

1. Find the assignments (choices) for which no ∞-coefficients appear.

2. Project the resulting matrices to keep only the vector representing the corresponding mwp-bound of the output value, w.r.t. the input values of f.

3. Obtain k possible mwp-certificates $M_f^1, M_f^2, \ldots, M_f^k$.

$$\frac{}{\vdash \ \texttt{Xi = F(X1,..., Xn)} : 1 \xleftarrow{\texttt{i}} ((M_f^1)\delta(0,c) \oplus \cdots \oplus (M_f^k)\delta(0,c)\delta(k,c))} \ \text{F}$$

Implementation: pymwp

A prototype static analyzer for a subset of C99 programs.

Source code and demo: statycc.github.io/pymwp/demo

Usage

```
pymwp /path/to/file.c [ARGS]
```

- Install: pip install pymwp
- List of args: pymwp --help

Summary

$$\text{mwp-analysis} \longrightarrow \text{mwp-analysis}' \xrightarrow{\ *\ } \text{realistic analysis}$$

Main result
 Lightweight, fast, practical data-size analysis focused on input value *growth*.

Key adjustments and enhancements
 Adjusted mathematical framework (deterministic rules, internalized failure);
 separating computation phases, function analysis, concrete implementation.

Limitations
 Even richer syntax (arrays, pointers, . . .); comparative evaluation.

Next steps

Formalization – formally verifying the original mwp-analysis in Coq cf. "Certifying Complexity Analysis" at CoqPL'23.

Extending current system – further improvements, richer syntax, etc.

Other directions – other ICC-based applications, e.g., optimizations.

doi.org/10.4230/LIPIcs.FSCD.2022.26 $\mathbf{\Omega}$ github.com/statycc

Original Inference Rules

$$\frac{}{\vdash_{\text{JK}} \text{Xi} : \{\_\text{i}^m\}} \text{ E1} \qquad\qquad \frac{\vdash_{\text{JK}} \text{C1} : M_1 \quad \vdash_{\text{JK}} \text{C2} : M_2}{\vdash_{\text{JK}} \text{if b then C1 else C2} : M_1 \oplus M_2} \text{ I}$$

$$\frac{}{\vdash_{\text{JK}} \text{e} : \{^w_i | \text{Xi} \in \text{var}(\text{e})\}} \text{ E2} \qquad \frac{\vdash_{\text{JK}} \text{Xi} : V_1 \quad \vdash_{\text{JK}} \text{Xj} : V_2}{\vdash_{\text{JK}} \text{Xi}\star\text{Xj} : pV_1 \oplus V_2} \text{ E3} \qquad \frac{\vdash_{\text{JK}} \text{Xi} : V_1 \quad \vdash_{\text{JK}} \text{Xj} : V_2}{\vdash_{\text{JK}} \text{Xi}\star\text{Xj} : V_1 \oplus pV_2} \text{ E4}$$

$$\frac{\vdash_{\text{JK}} \text{e} : V}{\vdash_{\text{JK}} \text{Xj = e} : 1 \xleftarrow{\text{j}} V} \text{ A} \qquad \forall i, M^*_{ii} = m \;\; \frac{\vdash_{\text{JK}} \text{C} : M}{\vdash_{\text{JK}} \text{loop X}_\ell\{\text{C}\} : M^* \oplus \{^p_\ell\rightarrow j \mid \exists i, M^*_{ij} = p\}} \text{ L}$$

$$\frac{\vdash_{\text{JK}} \text{C1} : M_1 \quad \vdash_{\text{JK}} \text{C2} : M_2}{\vdash_{\text{JK}} \text{C1; C2} : M_1 \otimes M_2} \text{ C} \qquad \forall i, M^*_{ii} = m \text{ and } \forall i,j, M^*_{ij} \neq p \;\; \frac{\vdash_{\text{JK}} \text{C} : M}{\vdash_{\text{JK}} \text{while b do } \{\text{C}\} : M^*} \text{ W}$$

Deterministic Inference Rules

$$\star \in \{+,-\} \quad \frac{}{\vdash \texttt{Xi}\star\texttt{Xj} : (0 \mapsto \{{}^{m}_{i}, {}^{p}_{j}\}) \oplus (1 \mapsto \{{}^{p}_{i}, {}^{m}_{j}\}) \oplus (2 \mapsto \{{}^{w}_{i}, {}^{w}_{j}\})} \; \mathsf{E^A}$$

$$\frac{}{\vdash \texttt{Xi} * \texttt{Xj} : \{{}^{w}_{i}, {}^{w}_{j}\}} \; \mathsf{E^M} \qquad\qquad \frac{}{\vdash \texttt{Xi} : \{{}^{m}_{i}\}} \; \mathsf{E^S} \qquad\qquad \frac{\vdash \texttt{e} : V}{\vdash \texttt{Xj = e} : 1 \overset{j}{\leftarrow} V} \; \mathsf{A}$$

$$\frac{\vdash \texttt{C1} : M_1 \quad \vdash \texttt{C2} : M_2}{\vdash \texttt{C1; C2} : M_1 \otimes M_2} \; \mathsf{C} \qquad\qquad \frac{\vdash \texttt{C1} : M_1 \quad \vdash \texttt{C2} : M_2}{\vdash \texttt{if b then C1 else C2} : M_1 \oplus M_2} \; \mathsf{I}$$

$$\frac{\vdash \texttt{C} : M}{\vdash \texttt{loop Xl \{C\}} : M^* \oplus \{{}^{\infty}_{j} \to j \mid M^*_{jj} \neq m\} \oplus \{{}^{p}_{\mathtt{l}} \to j \mid \exists i, M^*_{ij} = p\}} \; \mathsf{L^\infty}$$

$$\frac{\vdash \texttt{C} : M}{\vdash \texttt{while b do \{C\}} : M^* \oplus \{{}^{\infty}_{j} \to j \mid M^*_{jj} \neq m\} \oplus \{{}^{\infty}_{i} \to j \mid M^*_{ij} = p\}} \; \mathsf{W^\infty}$$

$$\frac{}{\vdash \texttt{Xi = F(X1,..., Xn)} : 1 \overset{i}{\leftarrow} ((M_f^1)\delta(0,c) \oplus \cdots \oplus (M_f^k)\delta(0,c)\delta(k,c))} \; \mathsf{F}$$