

mwp-Analysis Improvement and Implementation: Realizing Implicit Computational Complexity

Clément Aubert¹, Thomas Rubiano², Neea Rusch¹, Thomas Seiller^{2,3}

¹ Augusta University

² LIPN - Laboratoire d'Informatique de Paris-Nord

³ CNRS - Centre National de la Recherche Scientifique

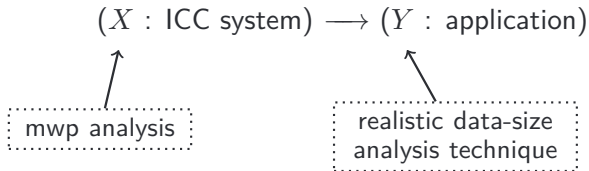
10 March 2023

Earlier: **Implicit Computational Complexity** $R \subseteq L = C$

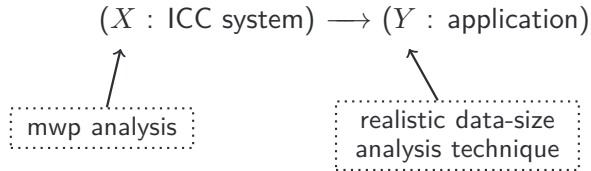
Earlier: **Implicit Computational Complexity** $R \subseteq L = C$

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

Earlier: **Implicit Computational Complexity** $R \subseteq L = C$



Earlier: **Implicit Computational Complexity** $R \subseteq L = C$



mwp-analysis \longrightarrow mwp-analysis' $\xrightarrow{*}$ 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 : $\llbracket C \rrbracket(x_i \rightsquigarrow x'_i)$.

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

$$C'' \equiv \begin{array}{l} X1 := 1; \\ \text{loop } X2 \{ X1 := X1 + X1 \} \end{array}$$

$$\llbracket C' \rrbracket(x_1, x_2, x_3 \rightsquigarrow x'_1, x'_2, x'_3)$$

$$x'_1 \leq 2x_2 + 2x_3$$

$$x'_2 \leq x_2$$

$$x'_3 \leq x_3$$

$$\llbracket C'' \rrbracket(x_1, x_2 \rightsquigarrow x'_1, x'_2)$$

$$x'_1 \leq 2^{x_2}$$

$$x'_2 \leq x_2$$

mwp analysis¹

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

C : program

M : matrix

$\vdash C : M$

¹Neil 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) $X_1 \mid X_2 \mid X_3 \mid \dots$ (aexp) $e + e \mid e * e$ (bexp) $e = e \mid e < e \mid \dots$
 (com) $\text{skip} \mid X := e \mid C;C \mid \text{if } b \text{ then } C \text{ else } C \mid \text{loop } X \{C\} \mid \text{while } b \text{ do } \{C\}$

Dependencies (“flows”) $\xrightarrow{\text{stronger}}$

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

Inference rules

$\frac{}{\vdash_{\text{JK}} X_i : \{i\}^m}$ E1 $\frac{\vdash_{\text{JK}} X_i : V_1 \quad \vdash_{\text{JK}} X_j : V_2}{\vdash_{\text{JK}} X_i * X_j : pV_1 \oplus V_2}$ E3 $\frac{\vdash e : V}{\vdash X_j = e : 1 \stackrel{j}{\leftarrow} V}$ A ...

mwp-bound $\max(\vec{x}, \text{poly}_1(\vec{y})) + \text{poly}_2(\vec{z})$

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

$$\frac{\frac{\frac{}{\vdash_{JK} X2 : \begin{pmatrix} 0 \\ m \\ 0 \end{pmatrix}} E1 \quad \frac{}{\vdash_{JK} X3 : \begin{pmatrix} 0 \\ 0 \\ m \end{pmatrix}} E1}{\vdash_{JK} X2+X3 : \begin{pmatrix} 0 \\ p \\ m \end{pmatrix}} E3}{\vdash_{JK} X1:=X2+X3 : \begin{pmatrix} 0 & 0 & 0 \\ p & m & 0 \\ m & 0 & m \end{pmatrix}} A$$

$$\vdots$$

$$\frac{}{\vdash_{JK} X1:=X1+X1 : \begin{pmatrix} p & 0 & 0 \\ 0 & m & 0 \\ 0 & 0 & m \end{pmatrix}} A$$

$$\vdots$$

$$\frac{}{\vdash_{JK} X1:=X2+X3;X1:=X1+X1 : \begin{pmatrix} 0 & 0 & 0 \\ p & m & 0 \\ p & 0 & m \end{pmatrix}} C$$

$$x'_1 \leq W_1(;;x_2, x_3) \wedge x'_2 \leq W_2(x_2) \wedge x'_3 \leq W_3(x_3)$$

$$C'' \equiv \begin{array}{l} X1 := 1; \\ \text{loop } X2 \{ X1 := X1 + X1 \} \end{array}$$

$$\frac{\frac{\frac{}{\vdash_{\text{JK}} X1 := 1 : \begin{pmatrix} m \\ 0 \end{pmatrix}} \text{E1}}{\vdots}}{\vdash_{\text{JK}} X1 := X1 + X1 : \begin{pmatrix} p & 0 \\ 0 & m \end{pmatrix}} \text{A}$$

$$\vdots$$

$$\times$$

$$\forall i, M_{ii}^* = m \frac{\vdash_{\text{JK}} C : M}{\vdash_{\text{JK}} \text{loop } X_\ell \{C\} : M^* \oplus \{ \overset{p}{\ell} \rightarrow j \mid \exists i, M_{ij}^* = p \}} \text{L}$$

Theorem: Soundness² $\vdash C : M$ implies $\models C : M$

$\vdash C : M$ 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.

²Jones 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

$$\frac{}{\vdash_{JK} \mathbf{Xi} : \{\mathbf{i}^m\}} \text{E1}$$

$$\frac{}{\vdash_{JK} \mathbf{e} : \{\mathbf{i}^w \mid \mathbf{Xi} \in \text{var}(\mathbf{e})\}} \text{E2}$$

$$\frac{\vdash_{JK} \mathbf{Xi} : V_1 \quad \vdash_{JK} \mathbf{Xj} : V_2}{\vdash_{JK} \mathbf{Xi} \star \mathbf{Xj} : pV_1 \oplus V_2} \text{E3}$$

$$\frac{\vdash_{JK} \mathbf{Xi} : V_1 \quad \vdash_{JK} \mathbf{Xj} : V_2}{\vdash_{JK} \mathbf{Xi} \star \mathbf{Xj} : V_1 \oplus pV_2} \text{E4}$$

$\mathbf{X2} + \mathbf{X3}$ has 3 derivations:

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

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

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

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 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) is equal to the unit of the mwp semi-ring, else 0.

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

Comparison

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

Original: 6×6 matrix $\times 3^2$ choices = 324 coefficients

Improved: 6 polynomials + 30 simple values = 66 coefficients

Example 2. $C \equiv X1 := X2 + X3$

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

The Failure Problem

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

Derivation of $X1 := X2 + X2$ yields two matrices: $\begin{pmatrix} 0 & 0 \\ p & m \end{pmatrix}$ and $\begin{pmatrix} 0 & 0 \\ w & m \end{pmatrix}$

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

\Rightarrow derivation $\begin{pmatrix} 0 & 0 \\ p & m \end{pmatrix}$ fails but derivation $\begin{pmatrix} 0 & 0 \\ w & m \end{pmatrix}$ 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., $\infty \times^\infty 0 = \infty$.

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

Apart from ∞ coefficients, the original and adjusted mwp systems agree.

The latter 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?

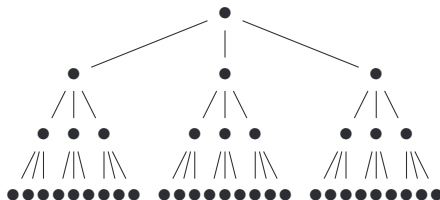
Apart from ∞ coefficients, the original and adjusted mwp systems agree.

The latter 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? → Delta graph
2. If yes, what is the concrete mwp-bound? → Efficient evaluation
3. If no, where does failure occur? → from matrix

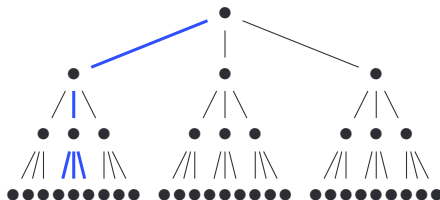
Delta graph enables decoupling computation of *existence* of bounds and computing its values.



Nodes: $n = (\delta(i, j)_1, \dots, \delta(i, j)_n)$

e.g., $n_1 = ((0, 1), (0, 2), (0, 3), (0, 4))$
 $n_2 = ((0, 1), (0, 2), (1, 3), (0, 4))$

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



$$n_1 = ((0, 1), (0, 2))$$

$$n_2 = ((0, 1), (1, 2))$$

$$n_3 = ((0, 1), (2, 2), (0, 3))$$

$$n_4 = ((0, 1), (2, 2), (1, 3))$$

$$n_5 = ((0, 1), (2, 2), (2, 3))$$

Evaluation

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

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

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.

$$[(2,1)] \times [(0,0), (2,2)] \times [(1,1)]$$

$$\text{initially: } [[0,1,2], [0,1,2], [0,1,2]]$$

$$(2,1)(0,0)(1,1) \Rightarrow [[1,2], [0], [0,1,2]]$$

$$(2,1)(2,2)(1,1) \Rightarrow [[0,1,2], [0], [0,1]]$$

4. Result is a disjunction of choice vectors.

$$[[1,2], [0], [0,1,2]] \vee [[0,1,2], [0], [0,1]]$$

Compositionality

Compositional analysis enables computing result once then reusing the result 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.

Cf. Carboneaux et al. "Compositional certified resource bounds" DOI: 10.1145/2737924.2737955

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 $M_f^1, M_f^2, \dots, M_f^k$.

$$\frac{}{\vdash \mathbf{Xi} = \mathbf{F}(\mathbf{X1}, \dots, \mathbf{Xn}) : 1 \stackrel{i}{\leftarrow} ((M_f^1)\delta(0, c) \oplus \dots \oplus (M_f^k)\delta(0, c)\delta(k, c))} \mathbf{F}$$

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

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

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



Formalization – formally verifying the original mwp-analysis in Coq cf. “Certifying Complexity Analysis” at CoqPL’23.

doi.org/10.4230/LIPIcs.FSCD.2022.26

 github.com/statycc

Original Inference Rules

$$\frac{}{\vdash_{\text{JK}} \mathbf{Xi} : \{-i^m\}} \text{E1}$$

$$\frac{\vdash_{\text{JK}} \mathbf{C1} : M_1 \quad \vdash_{\text{JK}} \mathbf{C2} : M_2}{\vdash_{\text{JK}} \mathbf{if\ b\ then\ C1\ else\ C2} : M_1 \oplus M_2} \text{I}$$

$$\frac{}{\vdash_{\text{JK}} \mathbf{e} : \{i^w \mid \mathbf{Xi} \in \text{var}(\mathbf{e})\}} \text{E2}$$

$$\frac{\vdash_{\text{JK}} \mathbf{Xi} : V_1 \quad \vdash_{\text{JK}} \mathbf{Xj} : V_2}{\vdash_{\text{JK}} \mathbf{Xi} \star \mathbf{Xj} : pV_1 \oplus V_2} \text{E3}$$

$$\frac{\vdash_{\text{JK}} \mathbf{Xi} : V_1 \quad \vdash_{\text{JK}} \mathbf{Xj} : V_2}{\vdash_{\text{JK}} \mathbf{Xi} \star \mathbf{Xj} : V_1 \oplus pV_2} \text{E4}$$

$$\frac{\vdash_{\text{JK}} \mathbf{e} : V}{\vdash_{\text{JK}} \mathbf{Xj} = \mathbf{e} : 1 \stackrel{j}{\leftarrow} V} \text{A}$$

$$\forall i, M_{ii}^* = m \quad \frac{\vdash_{\text{JK}} \mathbf{C} : M}{\vdash_{\text{JK}} \mathbf{loop\ } X_\ell \{\mathbf{C}\} : M^* \oplus \{_\ell^p \rightarrow j \mid \exists i, M_{ij}^* = p\}} \text{L}$$

$$\frac{\vdash_{\text{JK}} \mathbf{C1} : M_1 \quad \vdash_{\text{JK}} \mathbf{C2} : M_2}{\vdash_{\text{JK}} \mathbf{C1}; \mathbf{C2} : M_1 \otimes M_2} \text{C}$$

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

Deterministic Inference Rules

$$\begin{array}{c}
 \star \in \{+, -\} \frac{}{\vdash \mathbf{Xi} \star \mathbf{Xj} : (0 \mapsto \{^m_i, ^p_j\}) \oplus (1 \mapsto \{^p_i, ^m_j\}) \oplus (2 \mapsto \{^w_i, ^w_j\})} \mathbf{E}^A \\
 \\
 \frac{}{\vdash \mathbf{Xi} * \mathbf{Xj} : \{^w_i, ^w_j\}} \mathbf{E}^M \qquad \frac{}{\vdash \mathbf{Xi} : \{^m_i\}} \mathbf{E}^S \qquad \frac{\vdash \mathbf{e} : V}{\vdash \mathbf{Xj} = \mathbf{e} : 1 \stackrel{j}{\leftarrow} V} \mathbf{A} \\
 \\
 \frac{\vdash \mathbf{C1} : M_1 \quad \vdash \mathbf{C2} : M_2}{\vdash \mathbf{C1}; \mathbf{C2} : M_1 \otimes M_2} \mathbf{C} \qquad \frac{\vdash \mathbf{C1} : M_1 \quad \vdash \mathbf{C2} : M_2}{\vdash \mathbf{if} \ \mathbf{b} \ \mathbf{then} \ \mathbf{C1} \ \mathbf{else} \ \mathbf{C2} : M_1 \oplus M_2} \mathbf{I} \\
 \\
 \frac{\vdash \mathbf{C} : M}{\vdash \mathbf{loop} \ \mathbf{X1} \ \{\mathbf{C}\} : M^* \oplus \{^{\infty}_j \rightarrow j \mid M^*_{jj} \neq m\} \oplus \{^p_1 \rightarrow j \mid \exists i, M^*_{ij} = p\}} \mathbf{L}^{\infty} \\
 \\
 \frac{\vdash \mathbf{C} : M}{\vdash \mathbf{while} \ \mathbf{b} \ \mathbf{do} \ \{\mathbf{C}\} : M^* \oplus \{^{\infty}_j \rightarrow j \mid M^*_{jj} \neq m\} \oplus \{^{\infty}_i \rightarrow j \mid M^*_{ij} = p\}} \mathbf{W}^{\infty} \\
 \\
 \frac{}{\vdash \mathbf{Xi} = \mathbf{F}(\mathbf{X1}, \dots, \mathbf{Xn}) : 1 \stackrel{i}{\leftarrow} ((M^1_f)\delta(0, c) \oplus \dots \oplus (M^k_f)\delta(0, c)\delta(k, c))} \mathbf{F}
 \end{array}$$