mwp-Analysis Improvement and Implementation: Realizing Implicit Computational Complexity

Clément Aubert[1], Thomas Rubiano[2], Neea Rusch[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 $R \subseteq L = C$

$(X$: ICC system$) \longrightarrow (Y$: application$)$

mwp analysis

realistic data-size analysis technique

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

$(X$: ICC system$) \longrightarrow (Y$: application$)$

mwp analysis

realistic data-size analysis technique

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: $[\![\mathtt{C}]\!](x_i \rightsquigarrow x'_i)$.

```
C′ ≡ X1 := X2 + X3;
     X1 := X1 + X1
```

$[\![\mathtt{C}']\!](x_1, x_2, x_3 \rightsquigarrow x'_1, x'_2, x'_3)$

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

$x'_2 \leq x_2$

$x'_3 \leq x_3$

```
C″ ≡ X1 := 1;
     loop X2 {X1 := X1 + X1}
```

$[\![\mathtt{C}'']\!](x_1, x_2 \rightsquigarrow x'_1, x'_2)$

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

$x'_2 \leq x_2$

mwp Analysis[1]

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

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

[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) $\texttt{X}_1 \mid \texttt{X}_2 \mid \texttt{X}_3 \mid \dots$ (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

$$\frac{}{\vdash_{\text{JK}} \texttt{Xi} : \{^{m}_{\texttt{i}}\}} \text{ E1} \qquad \frac{\vdash_{\text{JK}} \texttt{Xi} : V_1 \quad \vdash_{\text{JK}} \texttt{Xj} : V_2}{\vdash_{\text{JK}} \texttt{Xi}\star\texttt{Xj} : pV_1 \oplus V_2} \text{ E3} \qquad \frac{\vdash \texttt{e} : V}{\vdash \texttt{Xj = e} : 1 \overset{\texttt{j}}{\leftarrow} V} \text{ A} \quad \dots$$

mwp-bound $\max(\vec{x}, poly_1(\vec{y})) + poly_2(\vec{z})$

$$\mathtt{C}' \equiv \mathtt{X1\ :=\ X2\ +\ X3;}$$
$$\mathtt{X1\ :=\ X1\ +\ X1}$$

$$\dfrac{\dfrac{\dfrac{}{\vdash_{\mathrm{JK}} \mathtt{X2} : \begin{pmatrix} 0 \\ m \\ 0 \end{pmatrix}}\,\mathsf{E1} \qquad \dfrac{}{\vdash_{\mathrm{JK}} \mathtt{X3} : \begin{pmatrix} 0 \\ 0 \\ m \end{pmatrix}}\,\mathsf{E1}}{\vdash_{\mathrm{JK}} \mathtt{X2+X3} : \begin{pmatrix} 0 \\ p \\ m \end{pmatrix}}\,\mathsf{E3}}{\vdash_{\mathrm{JK}} \mathtt{X1:=X2+X3} : \begin{pmatrix} 0 & 0 & 0 \\ p & m & 0 \\ m & 0 & m \end{pmatrix}}\,\mathsf{A}$$

$$\vdots$$

$$\dfrac{}{\vdash_{\mathrm{JK}} \mathtt{X1:=X1+X1} : \begin{pmatrix} p & 0 & 0 \\ 0 & m & 0 \\ 0 & 0 & m \end{pmatrix}}\,\mathsf{A}$$

$$\vdots$$

$$\dfrac{}{\vdash_{\mathrm{JK}} \mathtt{X1:=X2+X3;X1:=X1+X1} : \begin{pmatrix} 0 & 0 & 0 \\ p & m & 0 \\ p & 0 & m \end{pmatrix}}\,\mathsf{C}$$

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

```
C'' ≡ X1 := 1;
      loop X2 {X1 := X1 + X1}
```

$$\dfrac{}{\vdash_{\text{JK}} \texttt{X1:=1} : \begin{pmatrix} m \\ 0 \end{pmatrix}}\ \text{E1}$$

$$\vdots$$

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

$$\vdots$$

$$\times$$

$$\forall i, M^*_{ii} = m \quad \dfrac{\vdash_{\text{JK}} \texttt{C} : M}{\vdash_{\text{JK}} \texttt{loop X}_\ell\ \texttt{\{C\}} : M^* \oplus \{^p_\ell \!\to j \mid \exists i, M^*_{ij} = p\}}\ \text{L}$$

Theorem: Soundness[2]

$\vdash \texttt{C} : M$ implies $\vDash \texttt{C} : M$

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

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

Command $\texttt{C}$ 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

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

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

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

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

`X2 + X3` has 3 derivations:

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

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

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

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.

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

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. $\texttt{C} \equiv \texttt{X1 := X2 + X3}$

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

The Failure Problem

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

Derivation of `X1:=X2+X2` yields two 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 $\equiv$ `while(b){X1:=X2+X2}`

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

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?

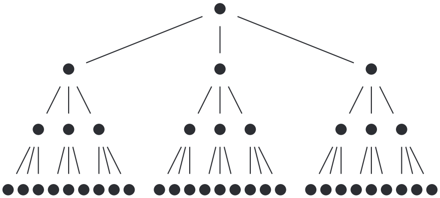
Apart from ∞ coefficients, the original and adjusted mwp systems agree.

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

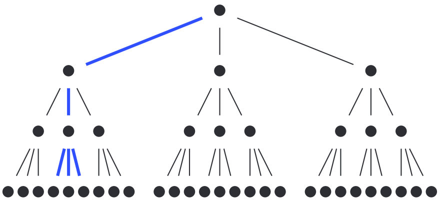
Delta graph enables decoupling computation of *existence* of bounds and computing its values.



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

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.



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

$((0,1),(0,2))$
$((0,1),(1,2))$
$((0,1),(2,2))$

$((0,1))$

Evaluation

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

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

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

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)]`

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. Carbonneaux 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 $M_f^1, M_f^2, \ldots, M_f^k$.

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

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

mwp-analysis $\longrightarrow$ mwp-analysis′ $\xrightarrow{*}$ 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

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

$$\frac{}{\vdash_{\text{JK}} \texttt{e} : \{{}_{\texttt{i}}^{w} \mid \texttt{Xi} \in \text{var}(\texttt{e})\}}\ \textsf{E2} \qquad \frac{\vdash_{\text{JK}} \texttt{Xi} : V_1 \quad \vdash_{\text{JK}} \texttt{Xj} : V_2}{\vdash_{\text{JK}} \texttt{Xi}\star\texttt{Xj} : pV_1 \oplus V_2}\ \textsf{E3} \qquad \frac{\vdash_{\text{JK}} \texttt{Xi} : V_1 \quad \vdash_{\text{JK}} \texttt{Xj} : V_2}{\vdash_{\text{JK}} \texttt{Xi}\star\texttt{Xj} : V_1 \oplus pV_2}\ \textsf{E4}$$

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

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

Deterministic Inference Rules

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

$$\frac{}{\vdash \mathtt{Xi\ *\ Xj} : \{{}^{w}_{\mathtt{i}}, {}^{w}_{\mathtt{j}}\}}\ \mathsf{E}^{\mathrm{M}} \qquad \frac{}{\vdash \mathtt{Xi} : \{{}^{m}_{\mathtt{i}}\}}\ \mathsf{E}^{\mathrm{S}} \qquad \frac{\vdash \mathtt{e} : V}{\vdash \mathtt{Xj\ =\ e} : 1 \xleftarrow{\mathtt{j}} V}\ \mathsf{A}$$

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

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

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

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