I Introduction

I.1 mwp-Bounds Analysis

The mwp-flow analysis [1] is a static analysis technique for evaluating resource usage of imperative programs. It analyzes input variables' value growth, and aims to discover a polynomially bounded data-flow relation between variables *initial* values and *final* values. When all variables are bounded by polynomials in inputs, the analysis succeeds, i.e., a program is derivable in the underlying calculus. If one or more variables is not bounded, such bound cannot be established. The soundness theorem of the analysis guarantees that if a derivation exists, the program's value growth is polynomially bounded in inputs. Furthermore, as a sound technique, it offers a computational method to certify programs data size growth properties at runtime.

To develop intuition of what mwp-flow analysis computes, consider the following example. Let $\mathtt{C}' \equiv$ `X1:=X2+X3; X1:=X1+X1` and $\mathtt{C}'' \equiv$ `X1:=1; loop X2 {X1:=X1+X1}` be imperative programs with standard operational semantics. For each variable $\mathtt{X}_i$, let x_i denote its initial value and x_i' its final value.

- Program $\mathtt{C}'$. Observe by inspection that variable `X1`'s final value is $x_1' \leq 2x_2 + 2x_3$. Variables `X2` and `X3` do not change and therefore are bounded by their initial values $x_2' \leq x_2$ and $x_3' \leq x_3$. We conclude all variables have a polynomial growth bound, and the program has the property of interest.
- Program $\mathtt{C}''$. Since variable `X2` does not change, its growth bound is $x_2' \leq x_2$. However, `X1` grows exponentially with bound $x_1' \leq 2^{x_2}$. We conclude program does not have a polynomial growth bound.

Instead of manual inspection, the mwp-flow analysis provides an *automatable* technique to distinguish programs that have polynomial growth bounds. Internally is works my applying inference rules to program commands and tracking variable dependencies by coefficients (or *flows*). A detailed description of the system is provided in the original work by Jones and Kristiansen [1] and refined by Aubert et. al [2]. The remainder of this section elaborates on the central notions and terminology to understand the mechanics of the analysis, as it relates to the publications in **??**.

II References

[1] Neil D. Jones and Lars Kristiansen. “A flow calculus of *mwp*-bounds for complexity analysis”. In: *ACM Transactions on Computational Logic* 10.4 (Aug. 2009), 28:1–28:41. DOI: 10.1145/1555746.1555752.

[2] Clément Aubert, Thomas Rubiano, Neea Rusch, and Thomas Seiller. “mwp-Analysis Improvement and Implementation: Realizing Implicit Computational Complexity”. In: *FSCD 2022*. Vol. 228. LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022, 26:1–26:23. DOI: 10.4230/LIPIcs.FSCD.2022.26.

$$\frac{}{\texttt{Xi} : \{^{m}_{\texttt{i}}\}}\ \text{E1} \qquad \frac{}{\texttt{e} : \{^{w}_{\texttt{i}} \mid \texttt{Xi} \in \text{var}(\texttt{e})\}}\ \text{E2} \qquad \star \in \{+,-\}\ \frac{\texttt{Xi} : V_1 \quad \texttt{Xj} : V_2}{\texttt{Xi}\star\texttt{Xj} : pV_1 \oplus V_2}\ \text{E3}$$

$$\star \in \{+,-\}\ \frac{\texttt{Xi} : V_1 \quad \texttt{Xj} : V_2}{\texttt{Xi}\star\texttt{Xj} : V_1 \oplus pV_2}\ \text{E4} \qquad \frac{\texttt{e} : V}{\texttt{Xj = e} : \mathbf{1} \xleftarrow{\texttt{j}} V}\ \text{A} \qquad \frac{\texttt{C1} : M_1 \quad \texttt{C2} : M_2}{\texttt{C1; C2} : M_1 \otimes M_2}\ \text{C}$$

$$\frac{\texttt{C1} : M_1 \quad \texttt{C2} : M_2}{\textbf{\texttt{if}}\ \texttt{b}\ \textbf{\texttt{then}}\ \texttt{C1}\ \textbf{\texttt{else}}\ \texttt{C2} : M_1 \oplus M_2}\ \text{I}$$

$$\forall i, M^*_{ii} = m\ \frac{\texttt{C} : M}{\textbf{\texttt{loop}}\ \texttt{X}_\ell\texttt{\{C\}} : M^* \oplus \{^{p}_{\ell} \rightarrow j \mid \exists i, M^*_{ij} = p\}}\ \text{L}$$

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

Figure 1: mwp-bounds flow analysis inference rules

III Appendices