Programming Languages & Compilers - Assignment 2

Hari Amoor, NetID: hra25

March 3, 2020

Problem 1: Define the RD data-flow problem for the CFG given and provide the MFP solution.

The data-flow problem for the CFG given is defined with $I = \{a, b, c, x\}$, where I is the Info domain. We approximate the family of functions F that are used to propagate variables defined in I with the MFP solution as follows:

$$\begin{aligned}
\text{IN}_{S0} &= \emptyset \\
\text{IN}_{C6} &= \{a = 1, b = 2, x \text{ as user input}\} \\
\text{IN}_{S1} &= \{a = 1, b = 2, x \text{ as user input}\} \\
\text{IN}_{S2} &= \{a = 1, b = 2, x \text{ as user input}\} \\
\text{IN}_{C7} &= \{a = 1, a = 2, b = 2, b = 10, c = 5, x \text{ as user input}\} \\
\text{IN}_{S3} &= \{a = 2, b = 10, x \text{ as user input}\} \\
\text{IN}_{S4} &= \{a = 1, b = 2, c = 5, x \text{ as user input}\} \\
\text{IN}_{S5} &= \{a = 1, a = 2, b = 2, b = 11, c = 5, x \text{ as user input}\}
\end{aligned}$$

We use the *join* operator, which corresponds to the union of sets, to define the underlying algebraic structure, i.e. the lattice.

Problem 2: Design a data-flow framework MUST-DEF as specified. Give the final MFP solution for the MUST-DEF framework.

We define the data-flow framework for MUST-DEF as follows:

$$\begin{aligned}
\text{INFO}_{\text{IN}}(v) &= \wedge_{p \in \text{PRED}(v)}(\text{INFO}_{\text{OUT}}(p)) \\
\text{INFO}_{\text{OUT}}(v) &= (\text{INFO}_{\text{IN}}(v) \setminus KILL(v)) \cup GEN(v)
\end{aligned}$$

We specify $\wedge = \cap$, the operator for intersection between sets. We provide the MFP solution of the MUST-DEF data-flow problem for *foo* as follows:

$$\begin{aligned}
\text{IN}_{S0} &= \emptyset \\
\text{IN}_{C6} &= \{a, b, x\} \\
\text{IN}_{S2} &= \{a, b, x\} \\
\text{IN}_{S2} &= \{a, b, x\} \\
\text{IN}_{C7} &= \{a, b, x\} \\
\text{IN}_{S3} &= \{a, b, x\} \\
\text{IN}_{S4} &= \{a, b, c, x\} \\
\text{IN}_{S5} &= \{a, b, x\}
\end{aligned}$$

We observe that, in the case that $x \geq 100$, c is not well-defined. Thus, the given procedure is unsafe, as $S5$ uses c without necessarily defining it.

Problem 3: Consider the following.

(a) **Describe how you would compute MAY-USE and MAY-DEF and provide MAY-USE(foo) and MAY-DEF(foo).**

We provide definitions for MAY-USE and MAY-DEF as follows:

$$\text{INFO}_{\text{MAY-USE}}(v) = \wedge_{p \in \text{PRED}(v)} READ(v)$$
$$\text{INFO}_{\text{MAY-DEF}}(v) = \wedge_{p \in \text{PRED}(v)} WRITE(v)$$

Here, READ and WRITE are operators to describe the variables read/written in a block v respectively, and $\wedge = \cup$.

Furthermore, MAY-USE(foo) = MAY-DEF(foo) = $\{a, b, c, x\}$. In particular, all of the variables in the procedure *may be* read or written.

(b) **How would the answer change if you instead used MUST-USE and MUST-DEF?**

As stated previously, in the case that $x \geq 100$, the variable c is read from but never written to or otherwise initialized. Therefore, MUST-USE(foo) = $\{a, b, c, x\}$ and MUST-DEF(foo) = $\{a, b, x\}$.