CSC 410F: Assignment 4

Due on October 30, 2017

Harman Sran **Ryan Downes** **Michael Li**

October 30, 2017

Problem 1

(a)

Show that Live Variable Analysis is a bit vector framework.

Proof. A bit vector framework satisfies precisely:

1. The analysis has a property space of the form:

$$L = (\mathcal{P}(D), \sqcap) \tag{1}$$

   Where D is a finite set and $\sqcap$ is either $\cup$ or $\cap$.

2. The set of all transfer functions is of the form:

$$\mathcal{F} = \{f : \mathcal{P}(D) \to \mathcal{P}(D) \mid \exists G, K \in D : \forall S \subseteq D, f(S) = (S \cap K) \cup G\} \tag{2}$$

The domain in LVA is the set of all labels crossed with all variables in the program. Both these pieces are finite, thus the domain of LVA is finite:

$$Domain = Lab \times Var \tag{3}$$

The meet operator, $\sqcap$, used in LVA is $\cup$ (set union). Combined with the above domain, we have that LVA satisfies property (1) of a bit vector framework.

We know that transformer functions in LVA, at some node l, are of the form:

$$f(l) = (l \setminus write(l)) \cup read(l) \tag{4}$$

We can rewrite the transformer function as:

$$f(l) = (l \cap (write(l))^c) \cup read(l) \tag{5}$$

Where $(write(l))^c$ is the set of variables that is the set complement of $write(l)$. This manipulation let's us rewrite LVA's transformer function to satisfy property (2) of a bit vector framework.

With both properties satisfied, we have that LVA is a bit vector framework. ■

(b)

Show that all bit vector frameworks are distributive frameworks.

Proof. We start by showing that a bit vector framework is monotone; that is, a monotone framework must consist of:

1. A complete lattice L, that satisfied the Ascending Chain Condition, and we write $\sqcup$ for the least upper bound operator.
2. A set $\mathcal{F}$ of monotone functions from L to L that contain the identify function and that is closed under function composition.

The first requirement follows from the first property of a BVF (1) - we have that a BVF must be a lattice with a finite domain D of form $L = (\mathcal{P}(D), \sqcap)$.

The set $\mathcal{F}$ of a BVF can be shown to be monotone; we have that the identify function is contained in $\mathcal{F}$ by taking $G = \emptyset \in D$ and $K = D \in D$, we obtain $f(S) = S$. Closure under function composition also holds (let $g, f \in \mathcal{F}$):

$$f(g(S)) = (((S \cap K_g) \cup G_g) \cap K_f) \cup G_f) \tag{6}$$
$$= (S \cap (K_g \cup K_f)) \cup ((G_g \cap K_f) \cup G_f) \tag{7}$$

Substituting, we have $f(g(S)) = (S \cap K_h) \cup G_h$ where $K_h = K_g \cup K_f$ and $G_h = (G_g \cap K_f) \cup G_f$. Then we have that BVFs are monotone frameworks.

Finally, to prove that BVFs are distributive frameworks consider $f \in \mathcal{F}$ such that $f(S) = (S \cap K) \cup G$:

$$f(S \sqcup S') = ((S \sqcup S') \cap K) \cup G) \tag{8}$$
$$= ((S \cap K) \sqcup (S' \cap K)) \cup G \tag{9}$$
$$= ((S \cap K) \cup G) \sqcup ((S' \cap K) \cup G) \tag{10}$$
$$= f(S) \sqcup f(S') \tag{11}$$

Then we have that BVFs are distributive frameworks. ■

(c)

Provide an example of a simple distributive framework that is not a bit vector framework to prove (by counterexample) that the converse of (b) is not true.

Constant propagation is an example of a distributive framework since its transfer function is:

$$v_1 \bigsqcup v_2 = \begin{cases} v_1 & \text{if } v_1 = v_2 \\ \text{T} & \text{else} \end{cases} \tag{12}$$

and since it's meet function is not union nor intersection it fails the first criteria of a bit vector framework.

Problem 2

(a)

Provide a precise formal definition of Partially Available Expressions.

First we define the domain and range of $kill_{PAE}$, gen_{PAE}, PAE_{entry}, and PAE_{exit}:

$$kill_{PAE}, gen_{PAE} : Blocks_* \longrightarrow \mathcal{P}(AExp_*) \tag{13}$$
$$PAE_{entry}, PAE_{exit} : Lab_* \longrightarrow \mathcal{P}(AExp_*) \tag{14}$$

Next, we define the functions:

$$kill_{PAE}([x := a]^l) = \{a' \in AExp_* \mid x \in FV(a')\} \tag{15}$$
$$kill_{PAE}([skip]^l) = \emptyset \tag{16}$$
$$kill_{PAE}([b]^l) = \emptyset \tag{17}$$

$$gen_{PAE}([x := a]^l) = \{a' \in AExp_* \mid x \notin FV(a')\} \tag{18}$$
$$gen_{PAE}([skip]^l) = \emptyset \tag{19}$$
$$gen_{PAE}([b]^l) = AExp(b) \tag{20}$$

$$PAE_{entry}(l) = \begin{cases} \emptyset & \text{if } l = inti(S_*) \\ \bigcup\{PAE_{exit}(l') \mid (l', l) \in flow(S_*)\} & \text{else} \end{cases} \tag{21}$$

$$PAE_{exit}(l) = \left\{(PAE_{entry}(l) \setminus kill_{PAE}(B^l)) \bigcup gen_{PAE}(B^l) \quad \text{where } B^l \in blocks(S_*) \right. \tag{22}$$

Problem 2 continued on next page...

(c)

Define a data flow analysis for Placement Possible Analysis.

First we define the domain and range of the $kill_{PPA}$, gen_{PPA}, PPA_{entry}, and PPA_{exit} functions.

$$kill_{PPA}, gen_{PPA} : Blocks_* \longrightarrow \mathcal{P}(AExp_*) \tag{23}$$
$$PPA_{entry}, PPA_{exit} : Lab_* \longrightarrow \mathcal{P}(AExp_*) \tag{24}$$

Next, we define the functions themselves:

$$kill_{PPA}([x := a]^l) = \{a' \in AExp_* \mid x \in FV(a')\} \tag{25}$$
$$kill_{PPA}([skip]^l) = \emptyset \tag{26}$$
$$kill_{PPA}([b]^l) = \emptyset \tag{27}$$

$$gen_{PPA}([x := a]^l) = AExp(a) \tag{28}$$
$$gen_{PPA}([skip]^l) = \emptyset \tag{29}$$
$$gen_{PPA}([b]^l) = AExp(b) \tag{30}$$

$$PPA_{exit}(l) = \begin{cases} \emptyset & \text{if } l = final(S_*) \\ \bigcup\{PPA_{entry}(l') \mid (l', l) \in flow^R(S_*)\} & \text{else} \end{cases} \tag{31}$$

$$PPA_{entry}(l) = \Big\{(PPA_{exit}(l) \setminus kill_{PPA}(B^l)) \bigcup gen_{PPA}(B^l) \quad \text{where } B^l \in blocks(S_*) \tag{32}$$

Altogether, we have:

Our proporty space is $= \mathcal{P}(AExp_*)$

Our transfer functions are $= PPA_{exit}, PPA_{entry}$

And the initalization information is specified in $= PPA_{exit}$

(d)

$$insert_l = \bigcup PPA_{entry}(l) \tag{33}$$
$$delete_l = \bigcap PPA_{exit}(l) \tag{34}$$

Note that these sets correspond to the examples on the handout.