Exercises

Program Analysis (CO70020)

Sheet 3

Exercise 1 *Consider the following* **while** *program:*

```
[y := 2]^1;
if [z > 1]^2
then [x := 1]^3
else [x := -1]^4;
[y := x * x]^5;
```

Perform a Constant Propagation Analysis CP *for this program.*

Solution We represent the state at a label as

$$\widehat{\sigma} = [x \mapsto \sigma(x), y \mapsto \sigma(y), z \mapsto \sigma(z)] = \begin{array}{|c|c|c|}\hline \mathtt{x} & \mathtt{y} & \mathtt{z} \\ \hline \sigma(x) & \sigma(y) & \sigma(z) \\ \hline \end{array}$$

The transfer functions are given as

$$\begin{aligned}
f_1^{\mathsf{CP}}\left(\begin{array}{|c|c|c|}\hline \mathtt{x} & \mathtt{y} & \mathtt{z} \\ \hline \sigma(x) & \sigma(y) & \sigma(z) \\ \hline \end{array}\right) &= \begin{array}{|c|c|c|}\hline \mathtt{x} & \mathtt{y} & \mathtt{z} \\ \hline \sigma(x) & 2 & \sigma(z) \\ \hline \end{array} \\
f_2^{\mathsf{CP}}\left(\begin{array}{|c|c|c|}\hline \mathtt{x} & \mathtt{y} & \mathtt{z} \\ \hline \sigma(x) & \sigma(y) & \sigma(z) \\ \hline \end{array}\right) &= \begin{array}{|c|c|c|}\hline \mathtt{x} & \mathtt{y} & \mathtt{z} \\ \hline \sigma(x) & \sigma(y) & \sigma(z) \\ \hline \end{array} \\
f_3^{\mathsf{CP}}\left(\begin{array}{|c|c|c|}\hline \mathtt{x} & \mathtt{y} & \mathtt{z} \\ \hline \sigma(x) & \sigma(y) & \sigma(z) \\ \hline \end{array}\right) &= \begin{array}{|c|c|c|}\hline \mathtt{x} & \mathtt{y} & \mathtt{z} \\ \hline 1 & \sigma(y) & \sigma(z) \\ \hline \end{array} \\
f_4^{\mathsf{CP}}\left(\begin{array}{|c|c|c|}\hline \mathtt{x} & \mathtt{y} & \mathtt{z} \\ \hline \sigma(x) & \sigma(y) & \sigma(z) \\ \hline \end{array}\right) &= \begin{array}{|c|c|c|}\hline \mathtt{x} & \mathtt{y} & \mathtt{z} \\ \hline -1 & \sigma(y) & \sigma(z) \\ \hline \end{array} \\
f_5^{\mathsf{CP}}\left(\begin{array}{|c|c|c|}\hline \mathtt{x} & \mathtt{y} & \mathtt{z} \\ \hline \sigma(x) & \sigma(y) & \sigma(z) \\ \hline \end{array}\right) &= \begin{array}{|c|c|c|}\hline \mathtt{x} & \mathtt{y} & \mathtt{z} \\ \hline \sigma(x) & \sigma(x)\widehat{*}\sigma(x) & \sigma(z) \\ \hline \end{array}
\end{aligned}$$

The analysis is then based on the following equations:

$$\begin{aligned}
\mathsf{CP}_{entry}(1) &= \lambda x.\top \\
\mathsf{CP}_{entry}(2) &= \mathsf{CP}_{exit}(1) \\
\mathsf{CP}_{entry}(3) &= \mathsf{CP}_{exit}(2) \\
\mathsf{CP}_{entry}(4) &= \mathsf{CP}_{exit}(2) \\
\mathsf{CP}_{entry}(5) &= \mathsf{CP}_{exit}(3) \sqcup \mathsf{CP}_{exit}(4)
\end{aligned}$$

and

$$\begin{aligned}
\mathsf{CP}_{exit}(1) &= f_1^{\mathsf{CP}}(\mathsf{CP}_{entry}(1)) \\
\mathsf{CP}_{exit}(2) &= f_2^{\mathsf{CP}}(\mathsf{CP}_{entry}(2)) \\
\mathsf{CP}_{exit}(3) &= f_3^{\mathsf{CP}}(\mathsf{CP}_{entry}(3)) \\
\mathsf{CP}_{exit}(4) &= f_4^{\mathsf{CP}}(\mathsf{CP}_{entry}(4)) \\
\mathsf{CP}_{exit}(5) &= f_5^{\mathsf{CP}}(\mathsf{CP}_{entry}(5))
\end{aligned}$$

The solutions are given by:

$$\mathsf{CP}_{entry}(1) = \begin{array}{|c|c|c|}\hline \mathtt{x} & \mathtt{y} & \mathtt{z} \\ \hline \top & \top & \top \\ \hline \end{array}$$

$$\mathsf{CP}_{entry}(2) = \begin{array}{|c|c|c|}\hline \mathtt{x} & \mathtt{y} & \mathtt{z} \\ \hline \top & 2 & \top \\ \hline \end{array}$$

$$\mathsf{CP}_{entry}(3) = \begin{array}{|c|c|c|}\hline \mathtt{x} & \mathtt{y} & \mathtt{z} \\ \hline \top & 2 & \top \\ \hline \end{array}$$

$$\mathsf{CP}_{entry}(4) = \begin{array}{|c|c|c|}\hline \mathtt{x} & \mathtt{y} & \mathtt{z} \\ \hline \top & 2 & \top \\ \hline \end{array}$$

$$\mathsf{CP}_{entry}(5) = \begin{array}{|c|c|c|}\hline \mathtt{x} & \mathtt{y} & \mathtt{z} \\ \hline \top & 2 & \top \\ \hline \end{array}$$

and

$$\mathsf{CP}_{exit}(1) = \begin{array}{|c|c|c|}\hline \mathtt{x} & \mathtt{y} & \mathtt{z} \\ \hline \top & 2 & \top \\ \hline \end{array}$$

$$\mathsf{CP}_{exit}(2) = \begin{array}{|c|c|c|}\hline \mathtt{x} & \mathtt{y} & \mathtt{z} \\ \hline \top & 2 & \top \\ \hline \end{array}$$

$$\mathsf{CP}_{exit}(3) = \begin{array}{|c|c|c|}\hline \mathtt{x} & \mathtt{y} & \mathtt{z} \\ \hline 1 & 2 & \top \\ \hline \end{array}$$

$$\mathsf{CP}_{exit}(4) = \begin{array}{|c|c|c|}\hline \mathtt{x} & \mathtt{y} & \mathtt{z} \\ \hline -1 & 2 & \top \\ \hline \end{array}$$

$$\mathsf{CP}_{exit}(5) = \begin{array}{|c|c|c|}\hline \mathtt{x} & \mathtt{y} & \mathtt{z} \\ \hline \top & \top & \top \\ \hline \end{array}$$

Exercise 2 *Consider the following simple imperative language with statements:*

$$S \quad ::= \quad \textbf{skip} \;\Big|\; x := a \;\Big|\; S_1 \; ; \; S_2 \;\Big|\; \textbf{if } b \textbf{ then } S_1 \textbf{ else } S_2 \;\Big|\; \textbf{while } b \textbf{ do } S$$

Define an instance of the monotone framework (similar to the Constant Propagation Analysis CP*) which performs a usage analysis* Use *of expressions.*

For each program point, the Use Analysis will determine the minimum and maximum number of program points the value of an arithmetic expression will be used when leaving this program point and before any variables in the expressions get redefined.

Assume that labelling, flow (flow) and reverse flow (flow^R), as well as initial and final labels are defined as usual. Let $\mathbf{AExp}_\star$ be the set of arithmetic sub-expressions in a program S. The lattice of abstract states is given by:

$$\widehat{\mathbf{State}} = (\mathbf{AExp}_\star \to \mathbf{N}_\infty \times \mathbf{N}_\infty)$$

with $\mathbf{N}_\infty = \{0, 1, 2, \ldots\} \cup \infty$. These record (at every label) the number of times an expression might be used and the number of times it is guaranteed to be used. Define a least upper bound operator $\sqcup$ and $\sqsubseteq$ on $\widehat{\mathbf{State}}$ and identify the $\perp$ and $\top$ elements. Use $\widehat{\mathbf{State}}$ to define a Use *Analysis like a monotone framework instance, i.e. specify the flow F, the extreme labels E, their initialization ι and the transfer functions. Derive the data-flow equations for the following program (after labelling it)*

```
y := a*b
while (x < a * b) do (
    x := a+b;
    y := a+b );
x := a*b;
```

Is this a may or a must analysis? How is it related to the VB *analysis? Discuss whether $\widehat{\mathbf{State}}$ fulfills the Ascending/Descending Chain Conditions, and whether the* Use *Analysis is computable, or how it can be made computable.*

Solution Describe the property "usage of an expression" by a pair, the first component is the minimal usage number, second component is maximum usage number. Denote the projections of first and second element in a pair by

$$(x, y)|_1 = x \text{ and } (x, y)|_2 = y.$$

Bottom:

$$\perp = \lambda e.(\infty, 0) \text{ with } e \in \mathbf{AExp}_\star$$

Top:

$$\top = \lambda e.(0, \infty) \text{ with } e \in \mathbf{AExp}_\star$$

Order:

$$\sigma_1 \sqsubseteq \sigma_2 \text{ iff } \sigma_1(e)|_1 \geq \sigma_2(e)|_1 \text{ and } \sigma_1(e)|_2 \leq \sigma_2(e)|_2 \quad \forall e \in \mathbf{AExp}_\star$$

Least Upper Bound:

$$(\sigma_1 \sqcup \sigma_2)(e) = (\min(\sigma_1(e)|_1, \sigma_2(e)|_1), \max(\sigma_1(e)|_2, \sigma_2(e)|_2)) \text{ with } e \in \mathbf{AExp}_\star$$

This is a backward analysis, i.e. $F = \mathit{flow}^R$.

The extremal labels $\mathbf{E}$ are therefore $\mathit{final}(S)$, and the initialization is $\iota(\ell) = \lambda e.(0, 0)$. Use the following notation: $(x, y) + 1 = (x + 1, y + 1)$ for pairs.

Transfer functions:

$$[\texttt{x:=}a]^\ell: \quad f_\ell(\sigma) = \begin{cases} f_\ell(\sigma)(e) = \sigma(e)+1 & \text{if } e \in \mathbf{AExp}(a) \text{ and } x \notin FV(e) \\ f_\ell(\sigma)(e) = (0,0) & \text{if } x \in FV(e) \\ f_\ell(\sigma)(e) = \sigma(e) & \text{otherwise} \end{cases}$$

$$[\mathbf{skip}]^\ell: \quad f_\ell(\sigma) = \sigma$$

$$[b]^\ell: \quad f_\ell(\sigma) = \begin{cases} f_\ell(\sigma)(e) = \sigma(e)+1 & \text{if } e \in \mathbf{AExp}(b) \\ f_\ell(\sigma)(e) = \sigma(e) & \text{otherwise} \end{cases}$$

These are similar to the VB analysis.

Labelling:

```
[y := a*b]^1
while [(x < a * b)]^2 do (
    [x := a+b]^3;
    [y := a+b]^4);
[x := a*b]^5;
```

$$\begin{aligned}
\mathsf{Use}_{entry}(1) &= f_1(\mathsf{Use}_{exit}(1)) \\
\mathsf{Use}_{entry}(2) &= f_2(\mathsf{Use}_{exit}(2)) \\
\mathsf{Use}_{entry}(3) &= f_3(\mathsf{Use}_{exit}(3)) \\
\mathsf{Use}_{entry}(4) &= f_4(\mathsf{Use}_{exit}(4)) \\
\mathsf{Use}_{entry}(5) &= f_5(\mathsf{Use}_{exit}(5))
\end{aligned}$$

or more explicitely

$$\begin{aligned}
\mathsf{Use}_{entry}(1) &= \mathsf{Use}_{exit}(1)[(a*b) \mapsto \mathsf{Use}_{exit}(1)(a*b)+1] \\
\mathsf{Use}_{entry}(2) &= \mathsf{Use}_{exit}(2)[(a*b) \mapsto \mathsf{Use}_{exit}(2)(a*b)+1] \\
\mathsf{Use}_{entry}(3) &= \mathsf{Use}_{exit}(3)[(a+b) \mapsto \mathsf{Use}_{exit}(3)(a+b)+1] \\
\mathsf{Use}_{entry}(4) &= \mathsf{Use}_{exit}(4)[(a+b) \mapsto \mathsf{Use}_{exit}(4)(a+b)+1] \\
\mathsf{Use}_{entry}(5) &= \mathsf{Use}_{exit}(4)[(a*b) \mapsto \mathsf{Use}_{exit}(5)(a*b)+1]
\end{aligned}$$

$$\begin{aligned}
\mathsf{Use}_{exit}(1) &= \mathsf{Use}_{entry}(2) \\
\mathsf{Use}_{exit}(2) &= \mathsf{Use}_{entry}(3) \sqcup \mathsf{Use}_{entry}(5) \\
\mathsf{Use}_{exit}(3) &= \mathsf{Use}_{entry}(4) \\
\mathsf{Use}_{exit}(4) &= \mathsf{Use}_{entry}(2) \\
\mathsf{Use}_{exit}(5) &= [(a*b) \mapsto (0,0), (a+b) \mapsto (0,0)]
\end{aligned}$$

This a monotone framework instance so may/must does not really makes sense. However, one can see it as a combination of a must and a may analysis.

If the minimal number of usages is zero then we can conclude that an expression is not very busy at the exit from a program point, otherwise it is very busy.

The lattice $\widehat{\textbf{State}}$ does not fulfill the ACC/DCC but we only need

$$\widehat{\textbf{State}} = (\textbf{AExp}_\star \to |\textbf{Lab}_\star| \times |\textbf{Lab}_\star|)$$

if we keep also information about which labels are already recorded and avoid counting them twice. This results in a finite lattice and thus fulfills the ACC/DCC, therefore the MFP solutions can then always be computed.

Exercise 3 *Consider the following program:*

$$[\texttt{x:=1}]^1; [\texttt{x:=x-1}]^2; [\texttt{x:=2}]^3$$

Clearly `x` *is dead at the exits from 2 and 3. But* `x` *is live at the exit of 1 even though its only use is to calculate a new value for a variable that turns out to be dead.*

> *We shall say that a variable is a* faint variable *if it is dead or if it is only used to calculate new values for faint variables; otherwise it is* strongly live*.*

In the example `x` *is faint at the exits from 1, 2 and 3.*
Define a Data Flow Analysis that detects Strongly Live variables.

Solution (Sketch) Two alternative approaches:

1. Base the analysis on the Live Variables Analysis. The function gen_{LV} must be changed to take an additional input — a set of strongly live variables:

$$gen_{\mathsf{LV}} : (\textbf{Blocks}_\star \times \mathcal{P}(\textbf{Var}_\star)) \to \mathcal{P}(\textbf{Var}_\star)$$

$$\begin{array}{lcl} gen_{\mathsf{LV}}([x := a]^\ell, X) & = & \begin{cases} FV(a) & \text{if } x \in X \\ \emptyset & \text{otherwise} \end{cases} \\ gen_{\mathsf{LV}}([\texttt{skip}]^\ell, X) & = & \emptyset \\ gen_{\mathsf{LV}}([b]^\ell, X) & = & FV(b) \end{array}$$

and also LV_{entry}:

$$\mathsf{LV}_{entry}(\ell) = (\mathsf{LV}_{exit}(\ell) \backslash kill_{\mathsf{LV}}(B^\ell)) \cup gen_{\mathsf{LV}}(B^\ell, \mathsf{LV}_{exit}(\ell))$$

2. As a monotone framework with transfer functions:

$$\begin{array}{lcl} f_{[x\,:=\,a]^\ell} X & = & \begin{cases} (X \backslash \{x\}) \cup FV(a) & \text{if } x \in X \\ X & \text{otherwise} \end{cases} \\ f_{[\texttt{skip}]^\ell} X & = & X \\ f_{[b]^\ell} X & = & X \cup FV(b) \end{array}$$

Exercise 4 *Consider the following Fun program:*

$$\begin{array}{l}(\texttt{let } f = (\texttt{fn } z \texttt{ => } 1) \texttt{ in} \\ \quad (((\texttt{fn } x \texttt{ => } x\ x)(\texttt{fn } y \texttt{ => } y))\ f))\end{array}$$

Label the program and give a brief and informal description of its execution: what's the result? Evaluate the expression formally using the eval function (from the lecture). For every step specify the environment ρ.

Solution

$$\begin{array}{l}(\texttt{let } (f = \texttt{fn } z \texttt{ => } 1^0)^1 \texttt{ in} \\ \quad (((\texttt{fn } x \texttt{ => } (x^2\ x^3)^4)^5 (\texttt{fn } y \texttt{ => } y^6)^7)^8\ f^9)^{10})^{11}\end{array}$$

It evaluates to $\texttt{fn } z \texttt{ => } 1$.

Shorthand notation:

$$\begin{array}{rcl}fz &=& (\texttt{fn } z \texttt{ => } 1^0)^1 \\ fx &=& (\texttt{fn } x \texttt{ => } (x^2\ x^3)^4)^5 \\ fy &=& (\texttt{fn } y \texttt{ => } y^6)^7 \\ t &=& (((\texttt{fn } x \texttt{ => } (x^2\ x^3)^4)^5 (\texttt{fn } y \texttt{ => } y^6)^7)^8\ f^9)^{10} \\ p &=& (\texttt{let } (f = \texttt{fn } z \texttt{ => } 1^0)^1 \texttt{ in} \\ && \quad (((\texttt{fn } x \texttt{ => } (x^2\ x^3)^4)^5 (\texttt{fn } y \texttt{ => } y^6)^7)^8\ f^9)^{10})^{11}\end{array}$$

Use $\rho_0 = [f \mapsto \perp, x \mapsto \perp, y \mapsto \perp, z \mapsto \perp]$.
Use $\rho_1 = [f \mapsto [fz, \rho_0], x \mapsto \perp, y \mapsto \perp, z \mapsto \perp]$.
Use $\rho_2 = [f \mapsto [fz, \rho_0], x \mapsto [fy, \rho_1], y \mapsto \perp, z \mapsto \perp]$.
Use $\rho_3 = [f \mapsto [fz, \rho_0], x \mapsto [fy, \rho_1], y \mapsto [fy, \rho_2], z \mapsto \perp]$.
Use $\rho_4 = [f \mapsto [fz, \rho_0], x \mapsto [fy, \rho_1], y \mapsto [fz, \rho_3], z \mapsto \perp]$.
Compute:

$$\text{eval}(\rho_0, p) = \text{eval}(\rho_0[f \mapsto v_1], ((fx\ fy)^8\ f^9)^{10}) = \text{eval}(\rho_1, ((fx\ fy)^8\ f^9)^{10})$$

as $v_1 = \text{eval}(\rho_0, fz) = [fz, \rho_0]$.

To compute $\text{eval}(\rho_1, ((fx\ fy)^6\ f^9)^{10}$ we need $\text{eval}(\rho_1, (fx\ fy)^8)$ and $\text{eval}(\rho_1, f^9)$.

On the one hand:

$$\text{eval}(\rho_1, f^9) = \rho_1(f) = [fz, \rho_0]$$

on the other hand:

$$\text{eval}(\rho_1, (fx\ fy)^8) = \text{eval}(\rho_2, (x^2\ x^3)^4)$$

as $\text{eval}(\rho_1, fx) = [fx, \rho_1] = [\texttt{fn } x \texttt{ => } (x^2\ x^3)^4, \rho_1]$ and $\text{eval}(\rho_1, fy) = [fy, \varrho_1]$).

Then we have:

$$\text{eval}(\rho_2, (x^2\ x^3)^4) = \text{eval}(\rho_3, fy) = [fy, \rho_2]$$

as $\text{eval}(\rho_2, x^2) = \text{eval}(\rho_2, x^3) = [fy, \rho_1] = [\texttt{fn}\ y\texttt{=>}y, \rho_1]$).

Therefore:

$$\text{eval}(\rho_1, ((fx\ fy)^8\ f^9)^{10}) = \text{eval}(\rho_3, (fy\ f^9)^{10}) = \text{eval}(\rho_4, y^6)$$

and with $\text{eval}(\rho_3, fy) = [fy, \rho_1]$) and $\text{eval}(\rho_3, f) = [fz, \rho_0]$ we get finally

$$\text{eval}(\rho_4, y^6) = \rho_4(y) = fz = (\texttt{fn}\ z\ \texttt{=>}\ 1^0)^1.$$