Exercises

Program Analysis (CO70020)

Sheet 5

Exercise 1 *Consider the following imperative language with statements of the form:*

$$
\begin{array}{rcl}
S & ::= & \texttt{x := a} \;\Big|\; \textbf{skip} \;\Big|\; S_1 \;;\; S_2 \;\Big|\; \textbf{if } b \textbf{ then } S_1 \textbf{ else } S_2 \;\Big|\; \textbf{while } b \textbf{ do } S \\
 & & \Big|\; \textbf{choose } S_1 \mid S_2 \mid \ldots \mid S_n \;\Big|\; \textbf{combine } S_1 \mid S_2 \mid \ldots \mid S_n
\end{array}
$$

In the **choose** *statement only one of the $n \geq 1$ statements S_i is actually selected to be executed. The* **combine** *executes all of the n statements S_i in some sequence. In both statements the choices are made non-deterministicly.*

Define a Live Variable Analysis LV*, similar to the one for the simple* **while** *language, for this extended language. Define an appropriate labelling for statements/blocks and give a definition for the flow flow (together with init and final).*

Solution Labelling:

$$
\begin{array}{rcl}
S & ::= & [\texttt{x := a}]^\ell \\
 & | & [\textbf{skip}]^\ell \\
 & | & S_1 \;;\; S_2 \\
 & | & \textbf{if } [b]^\ell \textbf{ then } S_1 \textbf{ else } S_2 \\
 & | & \textbf{choose } S_1 \mid S_2 \mid \ldots \mid S_n \\
 & | & \textbf{combine } S_1 \mid S_2 \mid \ldots \mid S_n \\
 & | & \textbf{while } [b]^\ell \textbf{ do } S
\end{array}
$$

Initial Labels:

$$\text{init} : \textbf{Stmt} \to \mathcal{P}(\textbf{Lab})$$

defined as:

$$
\begin{array}{rcl}
\text{init}([\texttt{x := a}]^\ell) & = & \{\ell\} \\
\text{init}([\textbf{skip}]^\ell) & = & \{\ell\} \\
\text{init}(S_1 \;;\; S_2) & = & \text{init}(S_1) \\
\text{init}(\textbf{if } [b]^\ell \textbf{ then } S_1 \textbf{ else } S_2) & = & \{\ell\} \\
\text{init}(\textbf{choose } S_1 \mid S_2 \mid \ldots \mid S_n) & = & \bigcup_{i=1}^n \text{init}(S_i) \\
\text{init}(\textbf{combine } S_1 \mid S_2 \mid \ldots \mid S_n) & = & \bigcup_{i=1}^n \text{init}(S_i) \\
\text{init}(\textbf{while } [b]^\ell \textbf{ do } S) & = & \{\ell\}
\end{array}
$$

Final Labels:

$$\text{final} : \mathbf{Stmt} \to \mathcal{P}(\mathbf{Lab})$$

defined as:

$$\begin{array}{rcl}
\text{final}([\mathtt{x\ :=\ a}]^\ell) & = & \{\ell\} \\
\text{final}([\mathbf{skip}]^\ell) & = & \{\ell\} \\
\text{final}(S_1\ ;\ S_2) & = & \text{final}(S_2) \\
\text{final}(\mathbf{if}\ \ [b]^\ell\ \mathbf{then}\ \ S_1\ \mathbf{else}\ \ S_2) & = & \text{final}(S_1) \cup \text{final}(S_2) \\
\text{final}(\mathbf{choose}\ S_1 \mid S_2 \mid \ \ldots \mid S_n) & = & \bigcup_{i=1}^n \text{final}(S_i) \\
\text{final}(\mathbf{combine}\ S_1 \mid S_2 \mid \ \ldots \mid S_n) & = & \bigcup_{i=1}^n \text{final}(S_i) \\
\text{final}(\mathbf{while}\ [b]^\ell\ \mathbf{do}\ \ S) & = & \{\ell\}
\end{array}$$

Flow:

$$\mathit{flow} : \mathbf{Stmt} \to \mathcal{P}(\mathbf{Lab} \times \mathbf{Lab})$$

defined as:

$$\begin{array}{rcl}
\mathit{flow}([\mathtt{x\ :=\ a}]^\ell) & = & \emptyset \\
\mathit{flow}([\mathbf{skip}]^\ell) & = & \emptyset \\
\mathit{flow}(S_1\ ;\ S_2) & = & \mathit{flow}(S_1) \cup \mathit{flow}(S_2) \cup \\
 & & \{(\ell, \ell') \mid \ell \in \text{final}(S_1), \ell' \in \text{init}(S_2)\} \\
\mathit{flow}(\mathbf{if}\ \ [b]^\ell\ \mathbf{then}\ \ S_1\ \mathbf{else}\ \ S_2) & = & \mathit{flow}(S_1) \cup \mathit{flow}(S_2) \cup \\
 & & \{(\ell, \ell') \mid \ell' \in \text{init}(S_1)\} \cup \\
 & & \{(\ell, \ell') \mid \ell' \in \text{init}(S_2)\} \\
\mathit{flow}(\mathbf{choose}\ S_1 \mid S_2 \mid \ \ldots \mid S_n) & = & \bigcup_{i=1}^n \mathit{flow}(S_i) \\
\mathit{flow}(\mathbf{combine}\ S_1 \mid S_2 \mid \ \ldots \mid S_n) & = & \bigcup_{i=1}^n \mathit{flow}(S_i) \cup \\
 & & \{(\ell_i, \ell_j) \mid \ell_i \in \text{final}(S_i), \ell_j \in \text{init}(S_j), \\
 & & i = 1, \ldots, n \wedge j = 1, \ldots, n \wedge i \neq j\} \\
\mathit{flow}(\mathbf{while}\ [b]^\ell\ \mathbf{do}\ \ S) & = & \mathit{flow}(S) \cup \{(\ell, \text{init}(S))\} \cup \\
 & & \{(\ell', \ell) \mid \ell' \in \text{final}(S)\}
\end{array}$$

There is no change in the local transfer functions ($\mathit{kill}_{\mathsf{LV}}$ and $\mathit{gen}_{\mathsf{LV}}$) as we have the same blocks as in the original language.

Exercise 2 *Consider the following expression from which labels have been stripped:*

$$\begin{array}{l}
(\mathtt{let}\ g = (\mathtt{fn}\ f\ \mathtt{=>}\ (\mathtt{if}\ (f\ 3)\ \mathtt{then}\ 10\ \mathtt{else}\ 5)) \\
\ \ \mathtt{in}\ (g\ \ (\mathtt{fn}\ y\ \mathtt{=>}(y > 2))\ )\ )
\end{array}$$

Label the expression and give a brief and informal description of its execution: what does it evaluate to?

Write down the constraints for a 0-CFA and provide the least solution that satisfies the constraints.

Solution Labelled program:

$$\begin{array}{rl}
e = & (\mathtt{let}\ g = (\mathtt{fn}\ f\ \mathtt{=>}\ (\mathtt{if}\ (f^1\ 3^2)^3 \mathtt{then}\ 10^4\ \mathtt{else}\ 5^5)^5)^6 \\
 & \ \ \mathtt{in}\ (g^8 (\mathtt{fn}\ y\ \mathtt{=>}(y^9 > 2^{10})^{11})^{12})^{13})^{14}
\end{array}$$

Let $f_6 = \texttt{fn}\ f\ \texttt{=>}\ e_6$, $f_{11} = \texttt{fn}\ y\ \texttt{=>}\ e_{11}$.

$$\begin{array}{l}
\{C(7) \subseteq r(g), C(13 \subseteq C(14), \{f_6\} \subseteq C(7), \\
C(4) \subseteq C(6), C(5) \subseteq C(6), r(f) \subseteq C(1), \\
\{f_6\} \subseteq C(1) \Rightarrow C(2) \subseteq r(f), \{f_{11}\} \subseteq C(1) \Rightarrow C(2) \subseteq r(y), \\
\{f_6\} \subseteq C(1) \Rightarrow C(6) \subseteq C(3), \{f_{11}\} \subseteq C(1) \Rightarrow C(11) \subseteq C(3), \\
r(g) \subseteq C(8), \{f_{11}\} \subseteq C(12), r(y) \subseteq C(9), \\
\{f_6\} \subseteq C(8) \Rightarrow C(12) \subseteq r(f), \{f_{11}\} \subseteq C(8) \Rightarrow C(12) \subseteq r(y), \\
\{f_6\} \subseteq C(8) \Rightarrow C(6) \subseteq C(13), \{f_{11}\} \subseteq C(8) \Rightarrow C(11) \subseteq C(13)
\end{array}$$

Solution: $C(1) = C(12) = r(f) = \{f_{11}\}$, $C(7) = C(8) = r(g) = \{f_6\}$. The rest is the empty set.

Exercise 3 *Consider the following extraction function for $n \in \mathbb{N}$:*

$$\beta(n) = \begin{cases} \min \textit{ bits to represent } n & \textit{if } n < 2^8 \\ \textbf{overflow} & \textit{otherwise} \end{cases}$$

which allows for a Bit-Size analysis for "small" integers via Abstract Interpretation.

Describe the (abstract) property lattice and the concrete and abstract domain (incl. ordering and least upper bound operation). Furthermore, define the abstraction, α, and concretisation, γ, functions.

Construct formally the abstraction (in the sense of Abstract Interpretation) of the doubling and square function, i.e. $f^\#$ and $g^\#$ for

$$f(n) = 2 \times n \quad \textit{and} \quad g(n) = n^2$$

Solution Arguably even for 0 we need at least one bit, so with normal order "$\leq = \sqsubseteq$" on $\mathbb{N}$

$$1 \sqsubseteq 2 \sqsubseteq \ldots \sqsubseteq 8 \sqsubseteq \textbf{overflow}$$

or if 0 is represented by 'nothing':

$$0 \sqsubseteq 2 \sqsubseteq \ldots \sqsubseteq 8 \sqsubseteq \textbf{overflow}$$

with this β is more formally:

$$\beta(n) = \begin{cases} 1 \text{ or } 0 & \text{for } n = 0 \\ k & \text{for } 1 \leq 2^{k-1} \leq n < 2^k \wedge n < 2^8 \\ \textbf{overflow} & \text{otherwise} \end{cases}$$

and $\mathcal{D} = \{1, \ldots, 8, \textbf{overflow}\}$ (or maybe $\mathcal{D} = \{1, \ldots, 8, \textbf{overflow}\}$). The least upper bound is essentially the maximum:

$$k_1 \sqcup k_2 = \beta(n) = \begin{cases} \max(k_1, k_2) & \text{for } \max(k_1, k_2) \leq 8 \\ \textbf{overflow} & \text{otherwise} \end{cases}$$

Bottom element could be 0, 1 or some undefined $\perp$.

For abstraction/concretisation we have $\alpha : \mathcal{P}(\mathbb{N}) \to \mathcal{D}$ and $\gamma : \mathcal{D} \to \mathcal{P}(\mathbb{N})$:

$$\alpha(N) = \begin{cases} 1 & \text{for } N \subseteq \{0,1\} \\ k & \text{for } N \subseteq \{2^{k-1}, \dots, 2^k - 1\} \\ \textbf{overflow} & \text{otherwise} \end{cases}$$

and

$$\gamma(k) = \begin{cases} \{0,1\} & \text{for } k = 1 \\ \{2^{k-1}, \dots, 2^k - 1\} & \text{for } k = 2, \dots, 8 \\ \mathbb{N} & \text{otherwise} \end{cases}$$

Construct the abstract versions using induced abstraction ($n \in \mathcal{D}$):

$$f^{\#}(n) = \alpha \circ f \circ \gamma(n) = \begin{cases} n+1 & \text{if } n < 8 \\ \textbf{overflow} & \text{overflow} \end{cases}$$

and

$$g^{\#}(n) = \alpha \circ g \circ \gamma(n) = \begin{cases} 2 \times n & \text{if } n < 4 \\ \textbf{overflow} & \text{overflow} \end{cases}$$

Exercise 4 *Consider a Sign Analysis for the imperative* WHILE *language. That is: We are interested in the* **sign** *of variables, i.e. whether we can guarantee that for a given program point and a variable x (at least) one of the following properties holds: $x = 0$, $x < 0$, $x > 0$, $x \leq 0$ and $x \geq 0$.*

Define a representation function β for this Sign Analysis. How can one define the corresponding correctness relation R_β? State formally what it means that the transfer functions f_ℓ for all labels are fulfilling the correctness condition.

Solution Representation function $\beta : \mathbb{Z} \to S$

$$\beta(x) = \begin{cases} = 0 & \text{if } x = 0 \\ < 0 & \text{if } x < 0 \\ > 0 & \text{if } x > 0 \end{cases}$$

Note: $\bot$, $\top$, ≤ 0 and $\geq$ not needed for β.

Correctness relation:

$$v \; R_\beta \; l \quad \text{iff} \quad \beta(v) \sqsubseteq l$$

Correctness, as

$$v_1 \; R_\beta \; l_1 \;\wedge\; p \vdash v_1 \rightsquigarrow v_2 \quad \Rightarrow \quad v_2 \; R_\beta \; f_\ell(l_1)$$

or maybe also via R_β, with $l_1 \rhd l_2$ with $f_\ell(l_1) = l_2$:

$$v_1 \; R_\beta \; l_1 \;\wedge\; p \vdash v_1 \rightsquigarrow v_2 \;\wedge\; p \vdash l_1 \rhd l_2 \quad \Rightarrow \quad v_2 \; R_\beta \; l_2$$