Exercises

Program Analysis (CO70020)

Sheet 5

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

\[
S ::= x := a \mid \text{skip} \mid S_1 ; S_2 \mid \text{if } b \text{ then } S_1 \text{ else } S_2 \mid \text{while } b \text{ do } S \\
   \mid \text{choose } S_1 \mid S_2 \mid \ldots \mid S_n \mid \text{combine } S_1 \mid S_2 \mid \ldots \mid S_n
\]

In the \textit{choose} statement only one of the \( n \geq 1 \) statements \( S_i \) is actually selected to be executed. The \textit{combine} executes all of the \( n \) statements \( S_i \) in some sequence. In both statements the choices are made non-deterministically.

Define a Live Variable Analysis \( \text{LV} \), similar to the one for the simple \textit{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:

\[
S ::= [x := a]^{\ell} \\
| [\text{skip}]^{\ell} \\
| S_1 ; S_2 \\
| \text{if } [b]^{\ell} \text{ then } S_1 \text{ else } S_2 \\
| \text{choose } S_1 \mid S_2 \mid \ldots \mid S_n \\
| \text{combine } S_1 \mid S_2 \mid \ldots \mid S_n \\
| \text{while } [b]^{\ell} \text{ do } S
\]

Initial Labels:

\[
\text{init} : \text{Stmt} \rightarrow \mathcal{P}(\text{Lab})
\]

defined as:

\[
\text{init}([x := a]) = \{\ell\} \\
\text{init}([\text{skip}]) = \{\ell\} \\
\text{init}(S_1 ; S_2) = \text{init}(S_1) \\
\text{init}(\text{if } [b]^{\ell} \text{ then } S_1 \text{ else } S_2) = \{\ell\} \\
\text{init}(\text{choose } S_1 \mid S_2 \mid \ldots \mid S_n) = \bigcup_{i=1}^{n} \text{init}(S_i) \\
\text{init}(\text{combine } S_1 \mid S_2 \mid \ldots \mid S_n) = \bigcup_{i=1}^{n} \text{init}(S_i) \\
\text{init}(\text{while } [b]^{\ell} \text{ do } S) = \{\ell\}
\]
Final Labels:

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

defined as:

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

Flow:

$$\text{flow : Stmt} \rightarrow \mathcal{P}(\text{Lab} \times \text{Lab})$$

defined as:

$$\text{flow}(\text{x := a}^{\ell}) = \emptyset$$
$$\text{flow}(\text{skip}^{\ell}) = \emptyset$$
$$\text{flow}(S_1 ; S_2) = \text{flow}(S_1) \cup \text{flow}(S_2) \cup \{(\ell, \ell') | \ell \in \text{final}(S_1), \ell' \in \text{init}(S_2)\}$$
$$\text{flow}(\text{if [b] then } S_1 \text{ else } S_2) = \text{flow}(S_1) \cup \text{flow}(S_2) \cup \{(\ell, \ell') | \ell' \in \text{init}(S_1)\} \cup \{(\ell, \ell') | \ell' \in \text{init}(S_2)\}$$
$$\text{flow}(\text{choose } S_1 | S_2 | \ldots | S_n) = \bigcup_{i=1}^n \text{flow}(S_i)$$
$$\text{flow}(\text{combine } S_1 | S_2 | \ldots | S_n) = \bigcup_{i=1}^n \text{flow}(S_i) \cup \{(\ell_i, \ell_j) | \ell_i \in \text{final}(S_i), \ell_j \in \text{init}(S_j),
\quad i = 1, \ldots, n \land j = 1, \ldots, n \land i \neq j\}$$
$$\text{flow}(\text{while [b] do } S) = \text{flow}(S) \cup \{(\ell, \text{init}(S))\} \cup \{(\ell', \ell) | \ell' \in \text{final}(S)\}$$

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

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

$$(\text{let } g = (\text{fn } f \Rightarrow (\text{if } f \text{ then } 10 \text{ else } 5))
\text{ in } (g \text{ (fn } y \Rightarrow (y > 2)) ) )$$

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:

$$e = (\text{let } g = (\text{fn } f \Rightarrow (\text{if } f^1 \text{ then } 10^4 \text{ else } 5^5)^5))
\text{ in } (g^8(\text{fn } y \Rightarrow (y^9 > 2^{10^{11}^{12^{13^{14}}}})))$$
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 $n \in \mathbb{N}$:

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

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, $\alpha$, and concretisation, $\gamma$, functions.

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

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

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

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

or if 0 is represented by ‘nothing’:

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

with this $\beta$ is more formally:

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

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

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

Bottom element could be 0, 1 or some undefined \bot.
For abstraction/concretisation we have \( \alpha : \mathcal{P}(\mathbb{N}) \to \mathcal{D} \) and \( \gamma : \mathcal{D} \to \mathcal{P}(\mathbb{N}) \):

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

and

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

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

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

and

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

**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 \( \beta \) for this Sign Analysis. How can one define the corresponding correctness relation \( R_\beta \)? State formally what it means that the transfer functions \( f_\ell \) 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: \( \bot, \top, \leq 0 \) and \( \geq \) not needed for \( \beta \).

Correctness relation:

\[
v R_\beta l \iff \beta(v) \sqsubseteq l
\]

Correctness, as

\[
v_1 R_\beta l_1 \land p \vdash v_1 \leadsto v_2 \Rightarrow v_2 R_\beta f_\ell(l_1)
\]

or maybe also via \( R_\beta \), with \( l_1 \gg l_2 \) with \( f_\ell(l_1) = l_2 \):

\[
v_1 R_\beta l_1 \land p \vdash v_1 \leadsto v_2 \land p \vdash l_1 \gg l_2 \Rightarrow v_2 R_\beta l_2
\]