Exercise 10

Exercise 1

Give two programs that are output equivalent (i.e. for the same initial state they result in the same final state) under the concrete domain, and they are not output equivalent under the interval domain.

Solution:

Program P_1

```
1     x := 2
2     sq_x := x*x
3     i = 0
```

Program P_2

```
1     x := 2
2     sq_x := 0
3     i := x
4     while (i > 0) do
5       sq_x := sq_x + x
6       i := i - 1
```

The programs are equivalent because for any initial state the programs' final state is $\{x \mapsto 2, sq\_x \mapsto 4, i \mapsto 0\}$.

If we compute P_1's least fixed point under the interval domain, at the end of P_1 we have $\{x \mapsto [2,2], sq\_x \mapsto [4,4], i \mapsto [0,0]\}$.

For P_2 we have $\{x \mapsto [2,2], sq\_x \mapsto [0,4], i \mapsto [0,0]\}$.

Exercise 2

1. The pointer analysis abstract domain is defined over the elements:

$$\mathsf{Lab} \to ((\mathsf{PtrVar} \to \mathcal{P}(\mathsf{AbsObj})) \times (\mathsf{AbsObj} \times \mathsf{Field} \to \mathcal{P}(\mathsf{AbsObj})))$$

   Define:

   (a) the partial order $\sqsubseteq$

   Solution:

   $m_1 \sqsubseteq m_2 \leftrightarrow m_1(l)(p) \subseteq m_2(l)(p) \wedge m_1(l)(a.f) \subseteq m_2(l)(a.f)$

   (b) the least ($\bot$) and greatest ($\top$) elements

   Solution:

   $\bot(l) = (m_p^\bot, m_h^\bot)$ where $\begin{cases} m_p^\bot = \lambda p.\ \emptyset \\ m_h^\bot = \lambda(a.f).\ \emptyset \end{cases}$

$$\top(l) = (m_p^\top, m_h^\top) \text{ where } \begin{cases} m_p^\top = \lambda p.\ \mathsf{AbsObj} \\ m_h^\top = \lambda(a.f).\ \mathsf{AbsObj} \end{cases}$$

(c) the meet $\sqcap$

Solution:

$$(m_1 \sqcap m_2)l = (m_p, m_h) \text{ where } \begin{cases} m_p = \lambda p.\ m_1(l)(p) \cap m_2(l)(p) \\ m_h = \lambda(a.f).\ m_1(l)(a.f) \cap m_2(l)(a.f) \end{cases}$$

(d) the join $\sqcup$

Solution:

$$(m_1 \sqcup m_2)l = (m_p, m_h) \text{ where } \begin{cases} m_p = \lambda p.\ m_1(l)(p) \cup m_2(l(p) \\ m_h = \lambda(a.f).\ m_1(l)(a.f) \cup m_2(l)(a.f) \end{cases}$$

2. The abstraction function α maps sets of states to pointer maps:

$$\mathcal{P}\Sigma \to \Big(\mathsf{Lab} \to \big[(\mathsf{PtrVar} \to \mathcal{P}\mathsf{AbsObj}) \times (\mathsf{AbsObj} \times \mathsf{Field} \to \mathcal{P}\mathsf{AbsObj})\big]\Big)$$

and the abstraction function γ maps pointer maps to sets of states:

$$\Big(\mathsf{Lab} \to \big[(\mathsf{PtrVar} \to \mathcal{P}\mathsf{AbsObj}) \times (\mathsf{AbsObj} \times \mathsf{Field} \to \mathcal{P}\mathsf{AbsObj})\big]\Big) \to \mathcal{P}\Sigma$$

Here the states are defined as $\sigma = (l_\sigma, \langle \rho_\sigma, r_\sigma, h_\sigma \rangle) \in \Sigma$ where

$l_\sigma \in \mathsf{Lab}$

$\rho_\sigma : \mathsf{Var} \to \mathbb{Z}$,

$r_\sigma : \mathsf{PtrVar} \to (\mathsf{Obj} \cup \{null\})$

$h_\sigma : \mathsf{Obj} \times \mathsf{Field} \to (\mathsf{Obj} \cup \{null\} \cup \mathbb{Z})$

(a) Define α and γ.

Solution:

Given a concrete object o, we write $loc(o)$ for the location (i.e. label) where o is allocated.

For every $S \subseteq \Sigma$ and $l \in \mathsf{Lab}$ we have $\alpha(S)(l) = (m_p, m_h)$ where

$$m_p = \lambda p.\ \{loc(o \in \mathsf{Obj}) \mid \exists \sigma \in S, l_\sigma = l, o = r_\sigma(p)\}$$
$$m_h = \lambda(a.f).\ \{loc(o \in \mathsf{Obj}) \mid \exists \sigma \in S, l_\sigma = l, \exists o' \in \mathsf{Obj}, loc(o') = a, h_\sigma(o', f) = o\}$$

We have $\gamma(m) = \{\sigma \in \Sigma \mid m(l_\sigma) = (m_p, m_h), r_\sigma \in \gamma_p(m_p), h_\sigma \in \gamma_h(m_h)\}$ where

$$\gamma_p(m_p) = \{r \mid \forall p \in \mathsf{PtrVar}, loc(r(p)) \in m_p(p)\}$$
$$\gamma_h(m_h) = \{h \mid \forall o \in \mathsf{Obj}, \forall f \in \mathsf{Field}, loc(h(o, f)) \in m_h(loc(o), f)\}$$

(b) Apply α on the following concrete set of states.

To avoid clutter, we do not write the function ρ in states. The initialization locations for the objects are $loc(o_1) = a_1$, $loc(o_2) = a_1$, and $loc(o_3) = a_2$.

$$\begin{aligned} S = \{ \quad & (1, \{p \mapsto o_1, q \mapsto o_2\}, \{o_1.k \mapsto o_3\}), \\ & (2, \{p \mapsto o_2, q \mapsto o_3\}, \{o_1.k \mapsto o_2\}), \\ & (2, \{p \mapsto o_1, q \mapsto o_1\}, \{o_1.k \mapsto o_2\}), \\ & (3, \{p \mapsto o_1, q \mapsto o_3\}, \{o_1.k \mapsto o_1\}) \quad \} \end{aligned}$$

(c) Apply γ on

$m = \{1 \mapsto (\{p \mapsto \{a_1, a_2\}, q \mapsto \{a_2\}\}, \{a_1.k \mapsto \{a_2\}\})\}$

Solution:

$$\begin{aligned}
\alpha(S) = \{ \quad & 1 \mapsto \{p \mapsto \{a_1\}, q \mapsto \{a_1\}\}, \{a_1.k \mapsto \{a_2\}\} \\
& 2 \mapsto \{p \mapsto \{a_1\}, q \mapsto \{a_1, a_2\}\}, \{a_1.k \mapsto \{a_1\}\}), \\
& 3 \mapsto \{p \mapsto \{a_1\}, q \mapsto \{a_2\}\}, \{a_1.k \mapsto \{a_1\}\}) \qquad \}
\end{aligned}$$

$$\begin{aligned}
\gamma(m) = \{ \quad & (1, \{p \mapsto o_1, q \mapsto o_3\}, \{o_1.k \mapsto o_3\}), \\
& (1, \{p \mapsto o_2, q \mapsto o_3\}, \{o_1.k \mapsto o_3\}), \\
& (1, \{p \mapsto o_3, q \mapsto o_3\}, \{o_1.k \mapsto o_3\}), \\
& (1, \{p \mapsto o_1, q \mapsto o_3\}, \{o_2.k \mapsto o_3\}), \\
& (1, \{p \mapsto o_2, q \mapsto o_3\}, \{o_2.k \mapsto o_3\}), \\
& (1, \{p \mapsto o_3, q \mapsto o_3\}, \{o_2.k \mapsto o_3\}), \quad \}
\end{aligned}$$

Exercise 3

In the lecture you have seen the abstract transformer for *pointer heap store* $p.f := q$. Define the remaining abstract transformers for the interval domain:

(object creation)	$p := newObject^l$
(compare two pointers)	$p = q$
(pointer assignment)	$p := q$
(pointer heap load)	$p := q.f$

Solution:

Let (m_p, m_h) be a tuple where m_p is a pointer map and m_h is a heap map, pointed to by some label l.

Object creation:
$[\![p := newObject^l]\!](m_p, m_h) = (m_p[p \mapsto l], m_h)$

Comparing two pointers:
$[\![p = q]\!](m_p, m_h) = (m_p[p \mapsto m_p(p) \cap m_p(q), q \mapsto m_p(p) \cap m_p(q)], m_h)$

Pointer assignment:
$[\![p := q]\!](m_p, m_h) = (m_p[p \mapsto m_p(q)], m_h)$

Pointer heap load:
$[\![p := q.f]\!](m_p, m_h) = (m_p[p \mapsto m_h(q.f)], m_h)$