

Assignment 10 (Solution)

Exercise 1

Recall from the lecture that the pointer analysis abstract domain is defined as follows:

$$\text{Labs} \rightarrow ((\text{PtrVar} \rightarrow \mathcal{P}(\text{AbsObj})) \times (\text{AbsObj} \times \text{Field} \rightarrow \mathcal{P}(\text{AbsObj})))$$

That is, the abstract domain keeps two mappings at every program label. The first one maps from pointer variables to a set of abstract objects they point to. The second one maps from fields of abstract objects to a set of abstract objects they point to.

1. Write down the formal definition for all abstract transformers that capture the effect of program statements manipulating pointers on the abstract domain:

- object creation, e.g., $x := \text{newObject}^l$

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 . The notation $m[x \mapsto y]$ denotes the map m modified so that x maps to y . We define

$$\llbracket x := \text{newObject}^l \rrbracket(m_p, m_h) = (m_p[x \mapsto \{l\}], m_h).$$

- comparison of two pointers, e.g., $x = y$

Solution: We define

$$\llbracket x = y \rrbracket(m_p, m_h) = (m_p[x \mapsto P, y \mapsto P], m_h),$$

where $P := m_p(x) \cap m_p(y)$.

- pointer assignment, e.g., $x := y$

Solution: We define

$$\llbracket x := y \rrbracket(m_p, m_h) = (m_p[x \mapsto m_p(y)], m_h).$$

- pointer heap store, e.g., $x.f := y$

Solution: We define

$$\llbracket x.f := y \rrbracket(m_p, m_h) = (m_p, m'_h),$$

where

$$m'_h(o.f) = \begin{cases} m_h(o.f) \cup m_p(y) & \text{if } o \in m_p(x) \\ m_h(o.f) & \text{otherwise.} \end{cases}$$

- pointer heap load, e.g., $x := y.f$

Solution: We define

$$\llbracket x := y.f \rrbracket(m_p, m_h) = \left(m_p \left[x \mapsto \bigcup_{o \in m_p(y)} m_h(o.f) \right], m_h \right).$$

2. Formally define the following things for the abstract domain.

- the partial order \sqsubseteq

Solution:

Let a and b two elements of the abstract domain. For all labels l , let a_p^l , a_h^l , b_p^l , and b_h^l be mappings such that $a(l) = (a_p^l, a_h^l)$ and $b(l) = (b_p^l, b_h^l)$. We define

$$a \sqsubseteq b \iff \forall l: (\forall x: a_p^l(x) \subseteq b_p^l(x)) \wedge (\forall x.f: a_h^l(x.f) \subseteq b_h^l(x.f)).$$

- the least element \perp

Solution: We define

$$\perp = \lambda l. (m_p^\perp, m_h^\perp),$$

where $m_p^\perp = \lambda x. \emptyset$ and $m_h^\perp = \lambda(x.f). \emptyset$.

- the greatest element \top

Solution: We define

$$\top = \lambda l. (m_p^\top, m_h^\top),$$

where $m_p^\top = \lambda x. \text{AbsObj}$ and $m_h^\top = \lambda(x.f). \text{AbsObj}$.

- the meet operation \sqcap

Solution: We define

$$a \sqcap b = \lambda l. (m_p^l, m_h^l),$$

where

$$\begin{aligned} m_p^l &= \lambda x. a_p^l(x) \cap b_p^l(x) \quad \text{and} \\ m_h^l &= \lambda(x.f). a_h^l(x.f) \cap b_h^l(x.f). \end{aligned}$$

- the join operation \sqcup

Solution: We define

$$a \sqcup b = \lambda l. (m_p^l, m_h^l),$$

where

$$m_p^l = \lambda x. a_p^l(x) \cup b_p^l(x) \quad \text{and} \\ m_h^l = \lambda(x.f). a_h^l(x.f) \cup b_h^l(x.f).$$

Exercise 2

Consider the following program:

```
0: c = newObject T;
1: t = c;
2: i = 0;
3: while (i < count) {
4:   n = newObject T;
5:   c.f = n;
6:   c = n;
7:   i++;
8: }
9: c.f = t;
10: assert t != n;
```

1. Run the flow-sensitive pointer analysis from the lecture on it.

Solution:

```
0: c = newObject T;
   {t->{}, c->{A0}, n->{}}
1: t = c;
   {t->{A0}, c->{A0}, n->{}}
2: i = 0;
   {t->{A0}, c->{A0}, n->{}}
3: while (i < count) {
   {t->{A0}, c->{A0, A4}, n->{A4}, A0.f->{A4}, A4.f->{A4}}
4:   n = newObject T;
   {t->{A0}, c->{A0, A4}, n->{A4}, A0.f->{A4}, A4.f->{A4}}
5:   c.f = n;
   {t->{A0}, c->{A0, A4}, n->{A4}, A0.f->{A4}, A4.f->{A4}}
6:   c = n;
   {t->{A0}, c->{A4}, n->{A4}, A0.f->{A4}, A4.f->{A4}}
7:   i++;
```

```

        {t->{A0}, c->{A4}, n->{A4}, A0.f->{A4}, A4.f->{A4}}
8: }
        {t->{A0}, c->{A0,A4}, n->{A4}, A0.f->{A4}, A4.f->{A4}}
9: c.f = t;
        {t->{A0}, c->{A0,A4}, n->{A4}, A0.f->{A0,A4}, A4.f->{A0,A4}}
10: assert t != n;

```

2. Can you prove the assertion on line 10 using the results of the analysis?

Solution: No, since variables `t` and `n` could be both `null`.

Exercise 3

Write a program for which the flow-sensitive pointer analysis from the lecture infers the following abstract state at the end of the program:

```
{a->{A0}, b->{A0,A1}, A0.f->{A0}, A1.f->{A0}}
```

Solution:

```

0: a = newObject T;
    {a->{A0}}
1: b = newObject T;
    {a->{A0}, b->{A1}}
2: if (*) {
    {a->{A0}, b->{A1}}
3:   b = a;
    {a->{A0}, b->{A0}}
4: }
    {a->{A0}, b->{A0,A1}}
5: b.f = a;
    {a->{A0}, b->{A0,A1}, A0.f->{A0}, A1.f->{A0}}

```

Exercise 4

Run both the flow-sensitive and the flow-insensitive pointer analysis on the following program:

```

0: a = newObject T;
1: b = a;
2: if (a == b) {
3:   b = newObject T;
4: } else {
5: }

```

Solution:

1. Flow-sensitive pointer analysis:

```
0: a = newObject T;  
   {a->{A0}}  
1: b = a;  
   {a->{A0}, b->{A0}}  
2: if (a == b) {  
   {a->{A0}, b->{A0}}  
3:   b = newObject T;  
   {a->{A0}, b->{A3}}  
4: } else {  
   {a->{A0}, b->{A0}}  
5: }  
   { a->{A0}, b->{A0,A3} }
```

2. Flow-insensitive pointer analysis:

```
{a->{A0}, b->{A0,A3}}
```