Assignment 10 (Solution)

Exercise 1

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

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

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., $x \coloneqq \mathit{newObject}^l$
     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

     $$[\![x := \mathit{newObject}^l]\!](m_p, m_h) = (m_p[x \mapsto \{l\}], m_h).$$

   - comparison of two pointers, e.g., $x = y$
     Solution: We define

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

     where $P \coloneqq m_p(x) \cap m_p(y)$.
   - pointer assignment, e.g., $x \coloneqq y$
     Solution: We define

     $$[\![x \coloneqq y]\!](m_p, m_h) = (m_p[x \mapsto m_p(y)], m_h).$$

- pointer heap store, e.g., $x.f \coloneqq y$
  Solution: We define

$$[\![x.f := y]\!](m_p, m_h) = (m_p, m'_h),$$

  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., $x \coloneqq y.f$
  Solution: We define

$$[\![x \coloneqq y.f]\!](m_p, m_h) = \Big(m_p\Big[x \mapsto \bigcup_{o \in m_p(y)} m_h(o.f)\Big], m_h\Big).$$

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^l_p, a^l_h, b^l_p, and b^l_h be mappings such that $a(l) = (a^l_p, a^l_h)$ and $b(l) = (b^l_p, b^l_h)$. We define

$$a \sqsubseteq b \iff \forall l\colon \big(\forall x\colon a^l_p(x) \subseteq b^l_p(x)\big) \wedge \big(\forall x.f\colon a^l_h(x.f) \subseteq b^l_h(x.f)\big).$$

- the least element $\bot$
  Solution: We define

$$\bot = \lambda l.\ \big(m^\bot_p, m^\bot_h\big),$$

  where $m^\bot_p = \lambda x.\ \emptyset$ and $m^\bot_h = \lambda(x.f).\ \emptyset$.
- the greatest element $\top$
  Solution: We define

$$\top = \lambda l.\ \big(m^\top_p, m^\top_h\big),$$

  where where $m^\top_p = \lambda x.\ \mathsf{AbsObj}$ and $m^\top_h = \lambda(x.f).\ \mathsf{AbsObj}$.
- the meet operation $\sqcap$
  Solution: We define

$$a \sqcap b = \lambda l.\ (m^l_p, m^l_h),$$

  where

$$m^l_p = \lambda x.\ a^l_p(x) \cap b^l_p(x) \quad \text{and}$$
$$m^l_h = \lambda(x.f).\ a^l_h(x.f) \cap b^l_h(x.f).$$

- 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}}
```