

# Assignment 10

## 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.,  $x := \text{newObject}^l$
  - comparison of two pointers, e.g.,  $x = y$
  - pointer assignment, e.g.,  $x := y$
  - pointer heap store, e.g.,  $x.f := y$
  - pointer heap load, e.g.,  $x := y.f$
2. Formally define the following things for the abstract domain.
  - the partial order  $\sqsubseteq$
  - the least element  $\perp$
  - the greatest element  $\top$
  - the meet operation  $\sqcap$
  - the join operation  $\sqcup$

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

## 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:

$\{a \rightarrow \{A0\}, b \rightarrow \{A0, A1\}, A0.f \rightarrow \{A0\}, A1.f \rightarrow \{A0\}\}$

## 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:  }
```