

Written Assignment 11

Solutions

1. (a) $Int \rightarrow Object$
 (b) Undefined.
 (c) $Int \rightarrow Object$
 (d) Undefined.
 (e) $(Object \rightarrow Int) \rightarrow (Int \rightarrow Object)$
2. An informal proof: if $T \leq T'$, then T and T' must have the same “shape” (i.e., different only up to the classes appearing in the types), and the claim follows from the fact that there are only finitely many types of the same shape. We can prove the above more formally by induction. Indeed, we can directly show that $\{T' \mid T \leq T'\}$ is finite by induction on the structure of T . We prove simultaneously that both $UB(T) = \{T' \mid T \leq T'\}$ and $LB(T) = \{T' \mid T' \leq T\}$ are finite.

The base case: Suppose $T = C$ for some class C . Then $UB(T) = \{C' \mid C \leq C'\}$ and $LB(T) = \{C' \mid C' \leq C\}$ are both finite because there are only finitely many classes.

The inductive case: Suppose $T = T_1 \rightarrow T_2$ for some T_1 and T_2 . Let $T' \in UB(T)$. It must be the case that $T' = T'_1 \rightarrow T'_2$ for some T'_1 and T'_2 such that $T'_1 \leq T_1$ and $T_2 \leq T'_2$. Therefore, $UB(T) = \{T'_1 \rightarrow T'_2 \mid T'_1 \in LB(T_1) \wedge T'_2 \in UB(T_2)\}$. By the induction hypothesis, $LB(T_1)$ and $UB(T_2)$ are finite, and so $UB(T)$ is also finite. By a similar argument, $LB(T) = \{T'_1 \rightarrow T'_2 \mid T'_1 \in UB(T_1) \wedge T'_2 \in LB(T_2)\}$ is also finite.

3. One good way to solve a problem like this is to break it into sub-problems:

- When do we need to do GC?
- How long will GC take?
- How long will the the program run between GC's?

Let's start with Stop-and-Copy:

- We need to do GC when From space fills up (remember, we divide the heap in half, into From and To space).
- Stop-and-Copy only bothers with the words in objects that are still live (that is, $s \cdot \frac{n}{2}$ words), but we have to copy them (effectively referencing the copied word as well), so it takes $s \cdot \frac{n}{2} \cdot 2 = sn$ cycles to do GC.
- The program will now continue running until it fills up the new From space. From space can hold $\frac{n}{2}$ words, but $s \cdot \frac{n}{2}$ of those words are filled with objects that were live from the last GC. So the program can run until it creates $\frac{n}{2} - s \cdot \frac{n}{2}$ words, which takes $10 \cdot (\frac{n}{2} - s \cdot \frac{n}{2}) = 5n - 5sn$ cycles.

After this, the machine will do the exact same thing over and over again, spending sn cycles on GC and $5n - 5sn$ cycles running the program, so we can use one iteration (GC + running the program) as a model for the steady state of the program.

One iteration takes $sn + 5n - 5sn = 5n - 4sn$ cycles, so the fraction of total time spent in GC is

$$\frac{sn}{5n - 4sn} = \frac{s}{5 - 4s}$$

Solving the problem for Mark-and-Sweep works the same way:

- Mark-and-Sweep needs to do GC only when the entire heap fills up.

- Mark-and-Sweep touches all the words in the heap during the sweep phase, so this takes n cycles.
- The program will continue running until the heap fills up again. sn of those words are filled with objects that survived the last GC, so the program only gets to create $n - sn$ words, which takes $10 \cdot (n - sn) = 10n - 10sn$ cycles.

Thus, the fraction of time spent in GC, using Mark-and-Sweep, is

$$\frac{n}{n + 10n - 10sn} = \frac{n}{11n - 10sn} = \frac{1}{11 - 10s}$$

Note that neither result depends on n , the size of the heap.