Programming Languages: Imperative Program Construction
Practicals 11: Separation Logic I

Shin-Cheng Mu

Autumn Term, 2021

1. Let $x, y : Int$ such that $x \neq y$, and let h_0 and h_1 be *singleton* heaps such that $h_0\ x = 0$ and $h_1\ y = 1$. For each of the predicate below, describe what heap it holds of, if any. (For example, $x \mapsto 0$ holds of h when $h = h_0$.)

    1. $x \mapsto 0$.
    2. $y \mapsto 1$.
    3. $(x \mapsto 0) * (y \mapsto 1)$.
    4. $(x \mapsto 0) * (x \mapsto 0)$.
    5. $x \mapsto 0 \lor y \mapsto 1$.
    6. $(x \mapsto 0) * (x \mapsto 0 \lor y \mapsto 1)$.
    7. $(x \mapsto 0 \lor y \mapsto 1) * (x \mapsto 0 \lor y \mapsto 1)$.
    8. $(x \mapsto 0) * (y \mapsto 1) * (x \mapsto 0 \lor y \mapsto 1)$.
    9. $(x \mapsto 0) * True$.
    10. $(x \mapsto 0) * \neg (x \mapsto 0)$.

Solution:

1. $x \mapsto 0$: $h = h_0$.
2. $y \mapsto 1$: $h = h_1$.
3. $(x \mapsto 0) * (y \mapsto 1)$: $h = h_0 \cdot h_1$.
4. $(x \mapsto 0) * (x \mapsto 0)$: *False*.
5. $x \mapsto 0 \lor y \mapsto 1$: $h = h_0 \lor h = h_1$.
6. $(x \mapsto 0) * (x \mapsto 0 \lor y \mapsto 1)$: $h = h_0 \cdot h_1$.
7. $(x \mapsto 0 \lor y \mapsto 1) * (x \mapsto 0 \lor y \mapsto 1)$: $h = h_0 \cdot h_1$.
8. $(x \mapsto 0) * (y \mapsto 1) * (x \mapsto 0 \lor y \mapsto 1)$: *False*
9. $(x \mapsto 0) * True$: $h_0 \subseteq h$.
10. $(x \mapsto 0) * \neg (x \mapsto 0)$: $h_0 \subseteq h$.

2. Prove

$$
\begin{array}{l}
\{(x \mapsto \_) * (y \mapsto \_)\} \\
\textbf{if } y = x + 1 \quad \rightarrow skip \\
\ \mid\ x = y + 1 \quad \rightarrow x := y \\
\ \mid\ |x - y| > 1 \rightarrow free\ x; free\ y \\
\qquad\qquad\qquad\quad x := \textbf{cons}\ (1, 2) \\
\textbf{fi} \\
\{x \mapsto \_, \_\}\ \ .
\end{array}
$$

Solution: If we add some more annotations:

$$
\begin{array}{lll}
\{(x \mapsto \_) * (y \mapsto \_)\} & & \\
\textbf{if } y = x + 1 & \rightarrow & \{((x \mapsto \_) * (y \mapsto \_)) \land y = x + 1\} \\
 & & skip \\
 & & \{x \mapsto \_, \_\} \\
\ \mid\ x = y + 1 & \rightarrow & \{((x \mapsto \_) * (y \mapsto \_)) \land x = y + 1\} \\
 & & x := y \\
 & & \{x \mapsto \_, \_\} \\
\ \mid\ |x - y| > 1 & \rightarrow & \{((x \mapsto \_) * (y \mapsto \_)) \land |x - y| > 1\} \\
 & & free\ x; free\ y \\
 & & x := \textbf{cons}\ (1, 2) \\
 & & \{x \mapsto \_, \_\} \\
\textbf{fi} & & \\
\{x \mapsto \_, \_\}\ \ , & &
\end{array}
$$

the three branches can be considered separately. The first branch is immediate:

$$
\begin{array}{ll}
 & (x \mapsto \_) * (y \mapsto \_) \land y = x + 1 \\
\Rightarrow & (x \mapsto \_) * (x + 1 \mapsto \_) \\
\equiv & x \mapsto \_, \_\ \ .
\end{array}
$$

The second branch:

$$
\begin{array}{ll}
 & wp\ (x := y)\ (x \mapsto \_, \_) \\
\equiv & y \mapsto \_, \_ \\
\equiv & (y \mapsto \_) * (y + 1 \mapsto \_) \\
\Leftarrow & ((x \mapsto \_) * (y \mapsto \_)) \land x = y + 1\ \ .
\end{array}
$$

The third branch can be verified using simple versions of global rules of deallocation and allocation:

$$
\begin{array}{l}
\{((x \mapsto \_) * (y \mapsto \_)) \land |x - y| > 1\} \\
free\ x \\
\{y \mapsto \_\} \\
free\ y \\
\{\textbf{emp}\} \\
x := \textbf{cons}\ (1, 2) \\
\{x \mapsto \_, \_\}\ \ .
\end{array}
$$

3. The following fragment creates a two-element cyclic structure containing relative addresses. Prove its correctness.

$\{\mathbf{emp}\}$
$x := \mathbf{cons}\ (a, a)$
$y := \mathbf{cons}\ (b, b)$
$^*(x + 1) := y - x$
$^*(y + 1) := x - y$
$\{\langle \exists k :: (x \mapsto a, k) * (x + k \mapsto b, -k) \rangle\}$

Hint: k in the existential quantification shall be instantiated to $y - x$.

Solution: One can verify the program using the non-overwriting global rule for allocation and the rule for mutation as below:

$\{\mathbf{emp}\}$
$x := \mathbf{cons}\ (a, a)$
$y := \mathbf{cons}\ (b, b)$
$\{(x \mapsto a, a) * (y \mapsto b, b)\}$
$^*(x + 1) := y - x$
$\{(x \mapsto a, y - x) * (y \mapsto b, b)\}$
$^*(y + 1) := x - y$
$\{(x \mapsto a, y - x) * (y \mapsto b, x - y)\}$

And the last assertion implies $\langle \exists k :: (x \mapsto a, k) * (x + k \mapsto b, -k) \rangle$, when k is instantiated to $y - x$.

Note: we can also use the backwards rule for mutation as its weakest precondition, and reason:

$$
\begin{array}{ll}
 & wp\ (^*(y + 1) := x - y)\ (y \mapsto b, x - y) \\
\equiv & \quad \{\text{ backwards rule for mutation }\} \\
 & (y + 1 \mapsto \_) * ((y + 1 \mapsto x - y) \mathrel{-\!\!*} (y \mapsto b, x - y)) \\
\equiv & \quad \{\text{ expanding abbrevation }\} \\
 & (y + 1 \mapsto \_) * ((y + 1 \mapsto x - y) \mathrel{-\!\!*} ((y + 1 \mapsto x - y) * (y \mapsto b))) \\
\Leftarrow & \quad \{\text{ since } R \Rightarrow (Q \mathrel{-\!\!*} (Q * R)) \} \\
 & (y + 1 \mapsto \_) * (y \mapsto b) \\
\Leftarrow & y \mapsto b, b\ \ .
\end{array}
$$

The Hoare triple

$\{(x \mapsto a, y - x) * (y \mapsto b, b)\}$
$^*(y + 1) := x - y$
$\{(x \mapsto a, y - x) * (y \mapsto b, x - y)\}$

then follows from the frame rule. We can then do similar reasoning with $^*(x + 1) := y - x$ to prove the other Hoare triple.