Problem Sheet 9: Program Proofs
Sample Solutions

Chris Poskitt
ETH Zürich

1 Axiomatic Semantics

i. I propose the axiom:

$$\vdash \{p\}\ \texttt{havoc}(\texttt{x}_0, \ldots, \texttt{x}_n)\ \{\exists x_0^{\text{old}}, \ldots, x_n^{\text{old}}.\ p[x_0^{\text{old}}/x_0, \ldots, x_n^{\text{old}}/x_n]\}$$

Essentially, it is the same as the forward assignment axiom (see Problem Sheet 1), but without conjuncts about the new values of each x_i, since we do not know what they will be after the execution of `havoc`.

ii. Below is a possible program and proof outline:

$\{x \geq 0\}$
$\{x! * 1 = x! \wedge x \geq 0\}$
    $y := 1;$
$\{x! * y = x! \wedge x \geq 0\}$
    $z := x;$
$\{z! * y = x! \wedge z \geq 0\}$
    while $z > 0$ **do**
    $\{z > 0 \wedge z! * y = x! \wedge z \geq 0\}$
    $\{(z-1)! * (y * z) = x! \wedge (z-1) \geq 0\}$
        $y := y * z;$
    $\{(z-1)! * y = x! \wedge (z-1) \geq 0\}$
        $z := z - 1;$
    $\{z! * y = x! \wedge z \geq 0\}$
    end
$\{\neg(z > 0) \wedge z! * y = x! \wedge z \geq 0\}$
$\{y = x!\}$

Observe that the loop invariant $z! * y = x! \wedge z \geq 0$ is key to completing the proof.

iii. A possible inference rule would be:

$$\text{[from-until]}\ \frac{\vdash \{p\}\ A\ \{inv\} \qquad \vdash \{inv \land \neg b\}\ C\ \{inv\}}{\vdash \{p\}\ \texttt{from}\ A\ \texttt{until}\ b\ \texttt{loop}\ C\ \texttt{end}\ \{inv \land b\}}$$

iv. A possible proof outline is the following:

$\{\ n \geq 0\ \}$
 from
  $k := n$
  $found := False$
 $\{\ 0 \leq k \leq n\ \land\ (found \Longrightarrow 1 \leq k \leq n \land A[k] = v)\ \}$
 until *found* **or** $k < 1$ **loop**
  $\{\ 1 \leq k \leq n\ \land\ \neg found\ \land\ (found \Longrightarrow 1 \leq k \leq n \land A[k] = v)\ \}$
  if $A[k]\ = v$ **then**
   $\{\ A[k] = v\ \land\ 1 \leq k \leq n\ \land\ \neg found\ \}$
   $\{\ 0 \leq k \leq n\ \land\ 1 \leq k \leq n\ \land\ A[k] = v\ \}$
   $found := True$
   $\{\ 0 \leq k \leq n\ \land\ (found \Longrightarrow 1 \leq k \leq n \land A[k] = v)\ \}$
  else
   $\{\ A[k] \neq v\ \land\ 1 \leq k \leq n\ \land\ \neg found\ \}$
   $\{\ 1 \leq k \leq n+1\ \land\ (found \Longrightarrow 2 \leq k \leq n+1 \land A[k-1] = v)\ \}$
   $k := k - 1$
   $\{\ 0 \leq k \leq n\ \land\ (found \Longrightarrow 1 \leq k \leq n \land A[k] = v)\ \}$
  end
  $\{\ 0 \leq k \leq n\ \land\ (found \Longrightarrow 1 \leq k \leq n \land A[k] = v)\ \}$
 end
 $\{\ (found \land 1 \leq k \leq n \land A[k] = v)\ \lor\ (\neg found \land k = 0)\ \}$
$\{(found \Longrightarrow 1 \leq k \leq n \land A[k] = v) \land (\neg found \Longrightarrow k < 1)\}$

Again, note the importance of determining a strong enough loop invariant, i.e.

$$0 \leq k \leq n \land (found \Longrightarrow 1 \leq k \leq n \land A[k] = v)$$

for the proof to be able to go through. Note that we can still apply backwards reasoning when the assignment involves a Boolean value (in this case, $found[True/found] = True$).

v. Assume that $\models \{p\}\ P\ \{q\}$ and $\vdash \{\mathrm{WP}[P,q]\}\ P\ \{q\}$. By the definition of $\models$, executing P on a state satisfying p results in a state satisfying q. By definition, $\mathrm{WP}[P,q]$ expresses the weakest requirements on the pre-state for P to establish q; hence p is either equivalent to or stronger than $\mathrm{WP}[P,q]$, and $p \Rightarrow \mathrm{WP}[P,q]$ is valid. Clearly, $q \Rightarrow q$ is also valid, so we can apply the rule of consequence [cons] and derive the result that $\vdash \{p\}\ P\ \{q\}$.

Note: this property is called *relative completeness*, i.e. all valid triples can be proven in the Hoare logic, relative to the existence of an oracle for deciding the validity of implications (such as those in [cons]).

2 Separation Logic

i. There are instances of s, h and p such that the state satisfies the first assertion. For example,

$$s, h \models x \mapsto x * \neg x \mapsto x$$

if $s(x) = 5$, $h(5) = 5$, and h is defined for no other values. However, $x = y * \neg(x = y)$ is not satisfiable since x, y denote values in the store, which is heap-independent.

ii. (a) Satisfies.

(b) Does not satisfy (the heap only contains two locations).

(c) Does not satisfy (the heap contains more than one location).

(d) Satisfies. The variables x and y are indeed evaluated to the same location by the store. The second conjunct expresses that there is a location in the heap determined by evaluating y (clearly true).

(e) Satisfies.

iii. A proof outline is given below:

$\{\mathrm{emp}\}$

$\quad x := \mathrm{cons}(5, 9);$

$\{x \mapsto 5, 9\}$

$\quad y := \mathrm{cons}(6, 7);$

$\{x \mapsto 5, 9 * y \mapsto 6, 7\}$

$\{\exists x^{\mathrm{old}}.\ x \mapsto 5, 9 * y \mapsto 6, 7 \wedge x^{\mathrm{old}} = x\}$

$\quad x := [x];$

$\{\exists x^{\mathrm{old}}.\ x^{\mathrm{old}} \mapsto 5, 9 * y \mapsto 6, 7 \wedge x = 5\}$

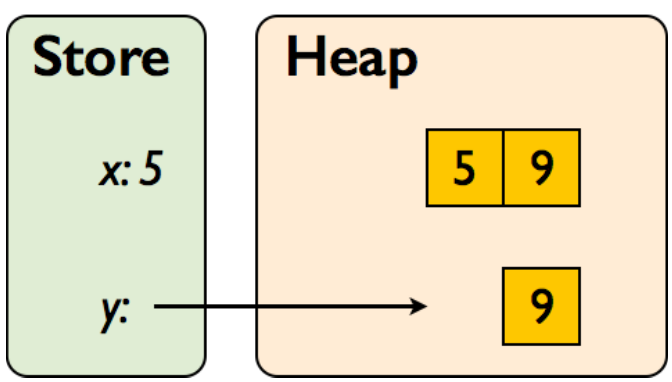
$\quad [y + 1] := 9;$

$\{\exists x^{\mathrm{old}}.\ x^{\mathrm{old}} \mapsto 5, 9 * y \mapsto 6, 9 \wedge x = 5\}$

$\quad \mathrm{dispose}(y);$

$\{\exists x^{\mathrm{old}}.\ x^{\mathrm{old}} \mapsto 5, 9 * y + 1 \mapsto 9 \wedge x = 5\}$

and a depiction of the final state:



iv. A proof outline is given below:

$\{tree((1, t),\, i)\}$

$\{\exists l, r.\ i \mapsto l, r * tree(1,\, l) * tree(t,\, r)\}$

$\quad x := [i];$

$\{\exists r.\ i \mapsto x, r * tree(1,\, x) * tree(t,\, r)\}$

$\quad [i] := 2;$

$\{\exists r.\ i \mapsto 2, r * tree(1,\, x) * tree(t,\, r)\}$

$\quad y := [i+1];$

$\{i \mapsto 2, y * tree(1,\, x) * tree(t,\, y)\}$

$\quad$ **dispose** i;

$\{(i+1) \mapsto y * tree(1,\, x) * tree(t,\, y)\}$

$\{(i+1) \mapsto y * x \mapsto 1 * tree(t,\, y)\}$

$\quad$ **dispose** x;

$\{(i+1) \mapsto y * tree(t,\, y)\}$

$\quad$ **dispose** $(i+1)$;

$\{tree(t,\, y)\}$