Problem Sheet 8: Separation Logic
Sample Solutions

Chris Poskitt
ETH Zürich

Starred exercises $(*)$ are more challenging than the others.

1 Separation Logic Assertions

i. (a) Does not hold. Expresses that the heap has exactly one location, obtained by evaluating i; but there are several other locations in the heap.

(b) Does not hold. Expresses that the heap can be split into two disjoint parts: one with the location obtained by evaluating i, and another with the location obtained by evaluating j, with the same contents at both. Not only are there additional locations in the heap, but i and j are evaluated w.r.t. the store to the *same* location (i.e. they do not denote two disjoint locations in the heap).

(c) Holds. The heap can be divided into two disjoint portions: one portion with exactly the location given by evaluating i; the other portion being the rest of it. The former satisfies $i \mapsto z$ and the latter satisfies true.

(d) Holds. The first conjunct is true for the reason in (c); the second is true because the store evaluates i and j to the same location. For the third conjunct, we can certainly split the heap into three disjoint parts: one satisfying $z \mapsto 1$, one satisfying $z+1 \mapsto 2$, and then the remainder of the heap which of course will satisfy true.

(e) Does not hold. It asserts that $i \mapsto x$ and $j \mapsto x'$ are disjoint parts of the heap, but i and j both evaluate to the same location in the heap.

(f) Holds. Every heap can be split into two disjoint parts: one, the original heap (which satisfies true), and the other, the empty heap (which satisfies emp).

ii. (a) p implies $p * p$ is not valid. Counterexample: take p to be $x \mapsto a$. A state satisfying this assertion (i.e. that there is exactly one location) will not satisfy $x \mapsto a * x \mapsto a$.

(b) $p * q$ implies $[(p \wedge q) * \text{true}]$ is not valid. Counterexample: take q to be $\neg p$. A heap can satisfy $p * \neg p$ because p might hold in one disjoint part, and $\neg p$ in another. But there is no part of any heap that will satisfy both p and $\neg p$, and so $[(p \wedge \neg p) * \text{true}]$ cannot be satisfied.

2 Separation Logic Proofs

i. Proof outline using the small axioms, frame rule, consequence rule, and the following derived axiom:

$\vdash \{e \mapsto e'\}\ x := [e]\ \{e \mapsto e' \wedge x = e'\}$

(provided that x does not appear free in e, e').

$\{\text{emp}\}$

$\quad l := \text{cons}(1)$

$\{l \mapsto 1\}$

$\{l \mapsto 1 * \text{emp}\}$

$\quad r := \text{cons}(2, 3)$

$\{l \mapsto 1 * r \mapsto 2, 3\}$

$\quad temp1 := [r + 1]$

$\{l \mapsto 1 * r \mapsto 2, 3 \wedge temp1 = 3\}$

$\quad temp2 := [l]$

$\{l \mapsto 1 * r \mapsto 2, 3 \wedge temp1 = 3 \wedge temp2 = 1\}$

$\quad [l] := temp1$

$\{l \mapsto temp1 * r \mapsto 2, 3 \wedge temp1 = 3 \wedge temp2 = 1\}$

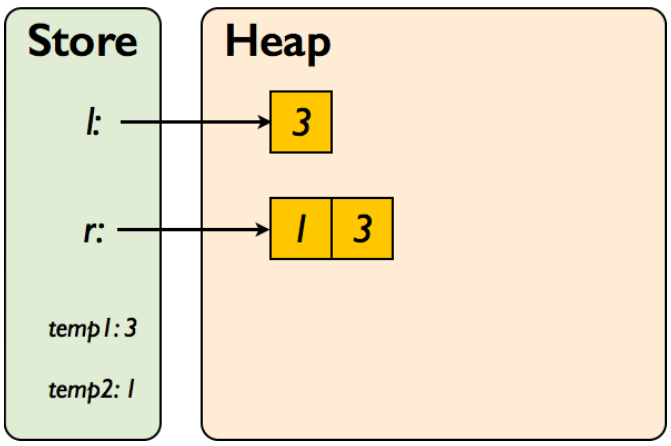
$\quad [r] := temp2$

$\{l \mapsto temp1 * r \mapsto temp2, 3 \wedge temp1 = 3 \wedge temp2 = 1\}$

$\{l \mapsto 3 * r \mapsto 1, 3 \wedge temp1 = 3 \wedge temp2 = 1\}$

$\{l \mapsto 3 * r \mapsto 1, 3\}$

A possible depiction of the post-state is given below:



ii. A possible proof outline:

$\{\text{list}(a :: as, i)\}$
$\{\exists j.\ i \mapsto a, j * \text{list}(as, j)\}$
$\quad\{i \mapsto a, j * \text{list}(as, j)\}$
$\quad\text{dispose}(i)$
$\quad\{i + 1 \mapsto j * \text{list}(as, j)\}$
$\quad k := [i + 1]$
$\quad\{i + 1 \mapsto j * \text{list}(as, j) \wedge k = j\}$
$\quad\text{dispose}(i + 1)$
$\quad\{\text{list}(as, j) \wedge k = j\}$
$\{\exists j.\ \text{list}(as, j) \wedge k = j\}$
$\{\text{list}(as, k)\}$
$\quad i := k$
$\{\text{list}(as, i)\}$

iii. (∗) Here is a possible procedure for CopyTree:

```
procedure CopyTree(p,q)
  if isAtom(p) then
     q := p;
  else
     local p1, p2, q1, q2;
     p1 := [p];
     p2 := [p+1];
     CopyTree(p1,q1);
     CopyTree(p2,q2);
     q := cons(q1,q2);
  end
end
```

We focus on the most crucial part of the program, i.e. the code in the else block. The following proof outline uses the small axioms, frame rule, rule of consequence, as well as:

$$\vdash \{\text{tree}(t_1, \tau)\}\ \text{CopyTree}(t_1, t_2)\ \{\text{tree}(t_1, \tau) * \text{tree}(t_2, \tau)\}$$

as an inductively assumed axiom (for recursive calls of the CopyTree procedure).

$\{\text{tree}(p, \tau)\}$

$\{\exists x, y, \tau_1, \tau_2.\ p \mapsto x, y * \text{tree}(x, \tau_1) * \text{tree}(y, \tau_2) \wedge \tau = \langle \tau_1, \tau_2 \rangle\}$

$p1 := [p]$

$\{\exists y, \tau_1, \tau_2.\ p \mapsto p1, y * \text{tree}(p1, \tau_1) * \text{tree}(y, \tau_2) \wedge \tau = \langle \tau_1, \tau_2 \rangle\}$

$p2 := [p + 1]$

$\{\exists \tau_1, \tau_2.\ p \mapsto p1, p2 * \text{tree}(p1, \tau_1) * \text{tree}(p2, \tau_2) \wedge \tau = \langle \tau_1, \tau_2 \rangle\}$

$\text{CopyTree}(p1, q1)$

$\{\exists \tau_1, \tau_2.\ p \mapsto p1, p2 * \text{tree}(p1, \tau_1) * \text{tree}(q1, \tau_1) * \text{tree}(p2, \tau_2) \wedge \tau = \langle \tau_1, \tau_2 \rangle\}$

$\text{CopyTree}(p2, q2)$

$\{\exists \tau_1, \tau_2.\ p \mapsto p1, p2 * \text{tree}(p1, \tau_1) * \text{tree}(q1, \tau_1) * \text{tree}(p2, \tau_2) * \text{tree}(q2, \tau_2) \wedge \tau = \langle \tau_1, \tau_2 \rangle\}$

$q := \text{cons}(q1, q2)$

$\{\exists \tau_1, \tau_2.\ p \mapsto p1, p2 * \text{tree}(p1, \tau_1) * \text{tree}(q1, \tau_1) * \text{tree}(p2, \tau_2) * \text{tree}(q2, \tau_2) * q \mapsto q1, q2 \wedge \tau = \langle \tau_1, \tau_2 \rangle\}$

$\{\exists \tau_1, \tau_2.\ p \mapsto p1, p2 * \text{tree}(p1, \tau_1) * \text{tree}(p2, \tau_2) * q \mapsto q1, q2 * \text{tree}(q1, \tau_1) * \text{tree}(q2, \tau_2) \wedge \tau = \langle \tau_1, \tau_2 \rangle\}$

$\{\text{tree}(p, \tau) * \text{tree}(q, \tau)\}$