PLSV, Mock Test, 2011

Question 1

The *lseg* predicate describes a segment of a singly-linked list. It is defined to be the least predicate satisfying the following equation:

$$lseg(E, F) \Leftrightarrow (E = F \wedge emp) \vee (E \neq F \wedge \exists x.\, E \mapsto x * lseg(x, F)).$$

Nodes do not contain values, only a pointer to the next node. A complete list starting at location E can hence be defined as $lseg(E, null)$. Consider the following routine, which takes two call-by-reference parameters:

```
append_list(x,y) {
  if (x=null) {
    x := y;
  } else {
    t := x;
    u := [t];
    while (u!=null) {
      t := u;
      u := [t];
    };
    [t] := y;
  }
}
```

Construct a proof, using Separation Logic, of the following specification:

$$\{lseg(\mathtt{x}, null) * lseg(\mathtt{y}, null)\}$$
$$\mathtt{append\_list(x,y)}$$
$$\{lseg(\mathtt{x}, null)\}$$

Your proof should be sufficiently detailed to be convincing, although proofs of straightforward arithmetical or logical facts may be elided. You may assume the following lemma without proof, but be sure to state clearly where you use it:

$$lseg(x, y) * lseg(y, z) \Rightarrow lseg(x, z).$$

[20 marks]

Question 2

(a) For each of the following specifications, either construct a proof using Concurrent Separation Logic, or explain why such a proof cannot exist.

1. $\{\mathtt{x} \mapsto \_\}$ `[x] := 5 || [x] := 5` $\{\mathtt{x} \mapsto 5\}$
2. $\{\mathtt{x} \mapsto \_ \wedge f \neq g\}$
   ```
   if f = 1 then [x] := 5
   ||
   if g = 1 then [x] := 5
   ```
   $\{\mathtt{x} \mapsto \_\}$

[8 marks]

(b) For each of the following specifications, either construct a proof using the Rely-Guarantee method, or explain why such a proof cannot exist. You may add auxiliary state to the programs as necessary.

1. $\{\mathtt{x} = 0\}$
   ```
   x := x + 3 || x := 5
   ```
   $\{\mathtt{x} = 8 \vee \mathtt{x} = 5\}$
2. $\{\mathtt{x} = 0\}$
   ```
   if x=0 then x := x + 1
   ||
   if x=0 then x := x + 2
   ||
   if x=0 then x := x + 1
   ```
   $\{\mathtt{x} \in \{1, 2, 3, 4\}\}$

[12 marks]

Question 3

(a) Which of the following separation logic formulae are valid, which are unsatisfiable, and which are neither? Give examples, counterexamples and formal proofs as appropriate.

(i) $A \Rightarrow A * A$

(ii) $x \mapsto 2 \wedge y \mapsto 2$

(iii) $x \mapsto 2 * x \mapsto 2$

(iv) $(A * B) \vee (A * C) \Rightarrow A * (B \vee C)$

(v) $(A * B) \wedge (A * C) \Rightarrow A * (B \wedge C)$

(vi) $(A \Rightarrow B) \Rightarrow (A * C \Rightarrow B * C)$

[12 marks]

(b) You have been asked to verify a concurrent program using either:

- Concurrent Separation Logic,
- the Owicki-Gries method, or
- the Rely-Guarantee method.

Explain the factors that could influence your decision about which verification method to use. [8 marks]

Question 4

For each of the following specifications, either prove it correct, or explain why it is invalid.

1. $\{even(\mathtt{x})\}$ `x := x + 1` $\{odd(\mathtt{x})\}$
2. $\{true\}$ `while true do skip` $\{false\}$
3. $\{\mathtt{x} = \mathtt{y} \wedge \mathtt{x} \mapsto \mathtt{z}\}$ `x := [x]` $\{\mathtt{y} \mapsto \mathtt{z} \wedge \mathtt{x} = \mathtt{z}\}$
4. $\{true\}$
```
r := x;
d := 0;
while r >= y do
  r := r - y;
  d := d + 1
```
$\{\mathtt{x} = (\mathtt{d} \times \mathtt{y}) + \mathtt{r} \wedge \mathtt{r} < \mathtt{y}\}$
5. $\{list(\mathtt{x})\}$
```
h := new();
t := h;
while (true) {
  p := t
  v := [x+1];
  [p+1] := v;
  x := [x];
  if (x!=0) {
    t := new;
    [p] := t;
  }
}
[p] := 0;
```
$\{list(\mathtt{x}) * list(\mathtt{h})\}$

[20 marks]

Question 5

(a) The conjunction rule is:

$$\frac{\{P_1\}\,\mathtt{C}\,\{Q_1\} \quad \{P_2\}\,\mathtt{C}\,\{Q_2\}}{\{P_1 \wedge P_2\}\,\mathtt{C}\,\{Q_1 \wedge Q_2\}}$$

Prove the conjunction rule is sound for sequential Hoare logic. That is, show:

$$\models \{P_1\}\,\mathtt{C}\,\{Q_1\} \;\wedge\; \models \{P_2\}\,\mathtt{C}\,\{Q_2\} \implies \models \{P_1 \wedge P_2\}\,\mathtt{C}\,\{Q_1 \wedge Q_2\}$$

[10 marks]

(b) The disjunction rule is:

$$\frac{\{P_1\}\,\mathtt{C}\,\{Q_1\} \quad \{P_2\}\,\mathtt{C}\,\{Q_2\}}{\{P_1 \vee P_2\}\,\mathtt{C}\,\{Q_1 \vee Q_2\}}$$

Prove the disjunction rule is sound for sequential Hoare logic. [10 marks]

Question 6

Consider the following concurrent program.

```
makezero(struct listnode *h) {
  cont := 1;
  x := h;
  resource r1 in {
   (
    while(cont == 1 ) {
      with r1 when true in {
        if(x != null) {
          resource r2 in {
           (
            with r2 when true [x.val] := 0
           ||
            with r2 when true [x.val] := 0
           )
          }
          x := [x.nxt];
        } else { cont := 0 }
      }
    }
   ||
    with r1 when true in {
      if (x != null) {
        [x.val] := 0;
        x := [x.nxt];
      }
    }
   )
  }
}
```

This program assumes that `h` points to a singly-linked list of nodes defined like so:

```
struct listnode { int val; struct listnode *nxt; };
```

The *lseg* predicate describes a segment of a singly-linked list. It is defined to be the least predicate satisfying the following equation:

$$\begin{array}{l} \mathit{lseg}(E, F) \iff \\ \quad (E = F \wedge \mathit{emp}) \vee (E \neq F \wedge (\exists x.\, E.\mathit{val} \mapsto \_ * E.\mathit{nxt} \mapsto x * \mathit{lseg}(x, F)) \end{array}$$

The notation "$E.\mathit{fieldname} \mapsto F$" can be understood as "$E + n \mapsto F$", where n is the offset of the field *fieldname*. For list nodes we define $E.\mathit{val}$ and $E.\mathit{nxt}$ as $E + 0$ and $E + 1$ respectively.

Prove the following specification for `makezero`:

$$\{\mathit{lseg}(\mathtt{h}, \mathit{null})\} \quad \mathtt{makezero(h)} \quad \{\mathit{lseg}(\mathtt{h}, \mathit{null})\}$$

Your proof should be sufficiently detailed to be convincing, although proofs of straightforward arithmetical or logical facts may be elided. You may assume the following joining lemma without proof, but be sure to state clearly where you use it:

$$\mathit{lseg}(x, y) * \mathit{lseg}(y, z) \Rightarrow \mathit{lseg}(x, z)$$

Your proof should include definitions of the resource invariants $I(\texttt{r1})$ and $I(\texttt{r2})$ associated with the locks. [20 marks]

Bonus question! The predicates *onelist* and *zerolist* are defined as follows:

$$\mathit{onelist}(E, F) \iff (E = F \wedge \mathit{emp}) \vee (E \neq F \wedge (\exists x.\, E.\mathit{val} \mapsto 1 * E.\mathit{nxt} \mapsto x * \mathit{onelist}(x, F)))$$

$$\mathit{zerolist}(E, F) \iff (E = F \wedge \mathit{emp}) \vee (E \neq F \wedge (\exists x.\, E.\mathit{val} \mapsto 0 * E.\mathit{nxt} \mapsto x * \mathit{zerolist}(x, F)))$$

Prove the following specification for `makezero`:

$$\{\mathit{onelist}(\texttt{h}, \mathit{null})\} \quad \texttt{makezero(h)} \quad \{\mathit{zerolist}(\texttt{h}, \mathit{null})\}$$

You may assume the *onelist* and *zerolist* equivalents of the joining lemma.