COMSM0067: Advanced Topics in Programming Languages
Problem Sheet: Separation Logic — Solutions

The frame rule, small axioms and list segments

What we need

Both problems: frame rule and consequence.

$$\frac{\{P\}\ c\ \{Q\}}{\{P * R\}\ c\ \{Q * R\}}\ \textsc{Frame} \qquad c \text{ modifies no variable free in } R$$

$$\frac{P \Rightarrow P' \qquad \{P'\}\ c\ \{Q'\} \qquad Q' \Rightarrow Q}{\{P\}\ c\ \{Q\}}\ \textsc{Conseq}$$

Problem 1: small axioms on single cells.

lookup	$\{x \mapsto v\}\ \ y := [x]\ \ \{y = v \wedge x \mapsto v\}$ $\quad (y \text{ fresh})$
allocation	$\{\mathsf{emp}\}\ \ x := \mathsf{new}(e)\ \ \{x \mapsto e\}$
mutation	$\{x \mapsto _\}\ \ [x] := e\ \ \{x \mapsto e\}$

Problem 2: list segments, and small axioms on records. A cell is a record $x \mapsto v, w$ with value v (field val) and next-pointer w (field next), and

$$\mathsf{lseg}(x, z)\ \equiv\ (x = z \wedge \mathsf{emp})\ \vee\ (x \neq z \wedge \exists v, w.\ x \mapsto v, w * \mathsf{lseg}(w, z)).$$

lookup	$\{x \mapsto v, w\}\ \ y := [x.\mathsf{next}]\ \ \{y = w \wedge x \mapsto v, w\}$ $\quad (y \text{ fresh})$
disposal	$\{x \mapsto _, _\}\ \ \mathsf{dispose}(x)\ \ \{\mathsf{emp}\}$
assignment	$\{Q[a/x]\}\ \ x := a\ \ \{Q\}$

Solutions

1.

$\{(x \mapsto a) * (y \mapsto b)\}$	
$\quad u := [x];$	lookup; frame $y \mapsto b$
$\{u = a \wedge (x \mapsto a) * (y \mapsto b)\}$	
$\quad v := [y];$	lookup; frame $u = a \wedge x \mapsto a$
$\{u = a \wedge v = b \wedge (x \mapsto a) * (y \mapsto b)\}$	
$\quad z := \mathsf{new}(u + v);$	allocation; frame the whole assertion
$\{u = a \wedge v = b \wedge (x \mapsto a) * (y \mapsto b) * (z \mapsto u + v)\}$	
$\quad [x] := 0$	mutation; frame $(y \mapsto b) * (z \mapsto u + v)$
$\{u = a \wedge v = b \wedge (x \mapsto 0) * (y \mapsto b) * (z \mapsto u + v)\}$	
$\{(x \mapsto 0) * (y \mapsto b) * (z \mapsto a + b)\}$	consequence

Notes on each step.

- *Lookups.* The small axiom needs exactly the one cell being read; the other cell is framed. Side condition: the lookup modifies only u (resp. v), which does not occur in the frame.
- *Allocation.* The small axiom starts from emp, so the *whole* current assertion is the frame: $\mathsf{emp} * R \equiv R$. Side condition: z does not occur in R. The $*$ in the result says the new cell is distinct from the cells at x and y.
- *Mutation.* The small axiom needs $x \mapsto _$, which $x \mapsto a$ provides; the two other cells are framed. Mutation modifies no variable, so the side condition holds trivially.
- *Consequence.* Using $u = a$ and $v = b$, the cell $z \mapsto u + v$ is $z \mapsto a + b$; then we drop the pure facts.

2.

$\{x \neq z \wedge \mathsf{lseg}(x, z)\}$	
$\{x \mapsto v, w * \mathsf{lseg}(w, z)\}$	**unfold** $\mathsf{lseg}(x, z)$; $x \neq z$ rules out $x = z$
$\quad y := [x.\mathsf{next}];$	lookup; frame $\mathsf{lseg}(w, z)$
$\{y = w \wedge x \mapsto v, w * \mathsf{lseg}(w, z)\}$	
$\{x \mapsto v, y * \mathsf{lseg}(y, z)\}$	consequence: substitute $w = y$
$\quad \mathsf{dispose}(x);$	disposal; frame $\mathsf{lseg}(y, z)$
$\{\mathsf{emp} * \mathsf{lseg}(y, z)\} \equiv \{\mathsf{lseg}(y, z)\}$	
$\quad x := y$	assignment: $\mathsf{lseg}(x, z)[y/x] = \mathsf{lseg}(y, z)$
$\{\mathsf{lseg}(x, z)\}$	

Notes on each step.

- *Unfold.* This happens before the first command. The unfolded assertion has $\exists v, w$; we fix v and w for the rest of the proof (existential rule).
- *Lookup.* Only the head cell is read; the rest of the segment, $\mathsf{lseg}(w, z)$, is framed. Side condition: the lookup modifies only y, which does not occur in $\mathsf{lseg}(w, z)$.
- *Disposal.* The head cell is given back to the allocator; $\mathsf{lseg}(y, z)$ is framed. Disposal modifies no variable. Because the postcondition is exact, no cell is leaked.
- *Assignment.* $x := y$ is an ordinary store update, so the Hoare assignment axiom applies: the precondition is the postcondition with y for x. No heap cell is involved, so no frame is needed.

Why the precondition needs $x \neq z$. Without it, $\mathsf{lseg}(x, z)$ may be the empty case $x = z \wedge \mathsf{emp}$, in which the heap has no cell at x. Then $[x.\mathsf{next}]$ faults (a null or dangling dereference), and the triple does not hold: a separation-logic triple promises that the program does not fault. For example, take $x = z$ and the empty heap. With $x \neq z$, the unfolding must take the non-empty case, which supplies the cell $x \mapsto v, w$ that the lookup needs.