Separation Logic Examples

Viktor Vafeiadis

May 1, 2014

1 Proof outline notation

In the examples, we tend to write the proofs in proof outline mode, by interleaving assertions and program commands. For example, we write:

$$\{\mathbf{emp}\}x := \mathbf{alloc}(0); \{x \mapsto 0\}[x] := 1\{x \mapsto 1\}$$

This is to be read as follows:

- By the Alloc rule, we have: $\{\mathbf{emp}\}x := \mathbf{alloc}(0)\{x \mapsto 0\}$.
- By the Write rule, we have: $\{x \mapsto 0\}[x] := 1\{x \mapsto 1\}$.
- By the Seq rule, we have $\{\mathbf{emp}\}\ x := \mathbf{alloc}(0); [x] := 1\ \{x \mapsto 1\}$.

In the individual steps, we also often freely use the structural rules of separation logic, (Frame, Conseq, Disj, Ex). For example:

$$\{x \mapsto 0\}c := \mathbf{alloc}(0); \{x \mapsto 0 * c \mapsto 0\}$$

Here, we used consequence rule, the frame rule, and the allocation axiom. A typical example using the disjunction rule is:

$$\{x \mapsto 0 \vee x \mapsto 1\}t := [x]\{x \mapsto 0 \wedge t = 0 \vee x \mapsto 1 \wedge t = 1\}$$

When for emphasis we want to show the use of the consequence rule, we write the two assertions one after the other. For example, we could have written our earlier step more explicitly as:

$$\{x \mapsto 0\}\{x \mapsto 0 * \mathbf{emp}\}c := \mathbf{alloc}(0); \{x \mapsto 0 * c \mapsto 0\}$$

For other compound commands, we use similar notation:

$$\begin{array}{c} \{P_1 * P_2\} \\ \left(\begin{array}{c|c} \{P_1\} & \{P_2\} \\ C_1 & C_2 \\ \{Q_1\} & \{Q_2\} \end{array}\right) \\ \{Q_1 * Q_2\} \end{array} \qquad \begin{array}{l} \{P\} \\ \mathbf{atomic}\ \{ \\ \quad \{P * J\} \\ \quad C \\ \quad \{Q * J\} \\ \} \\ \{Q\} \end{array}$$

Finally, we denote applications of the SHARE rule as:

$$\begin{array}{l}\{P * J\}\\ J \vdash \begin{pmatrix}\{P\}\\ C\\ \{Q\}\end{pmatrix}\\ \{Q * J\}\end{array}$$

2 Ownership Transfer

Let $J \stackrel{\text{def}}{=} c \mapsto 0 \vee c \mapsto 1 * x \mapsto 1$.

$$\begin{array}{l}\{\mathbf{emp}\}\\ x := \mathbf{alloc}(0);\\ \{x \mapsto 0\}\\ c := \mathbf{alloc}(0);\\ \{x \mapsto 0 * c \mapsto 0\}\\ \{x \mapsto 0 * \mathbf{emp} * J\}\\ \quad \{x \mapsto 0 * \mathbf{emp}\}\\ J \vdash \left(\begin{array}{l}\{x \mapsto 0\}\\ [x] := 1;\\ \{x \mapsto 1\}\\ \mathbf{atomic}\ \{\\ \quad \{x \mapsto 1 * J\}\\ \quad \{x \mapsto 1 * c \mapsto 0\}\\ \quad [c] := 1;\\ \quad \{x \mapsto 1 * c \mapsto 1\}\\ \quad \{\mathbf{emp} * J\}\\ \}\\ \{\mathbf{emp}\}\end{array} \middle\| \begin{array}{l}\{\mathbf{emp}\}\\ \mathbf{atomic}\ \{\\ \quad \{\mathbf{emp} * J\}\\ \quad \{c \mapsto 0 \vee c \mapsto 1 * x \mapsto 1\}\\ \quad t := [c];\\ \quad \left\{\begin{array}{l}c \mapsto 0 \wedge t = 0 \vee\\ c \mapsto 1 * x \mapsto 1 \wedge t = 1\end{array}\right\}\\ \quad \mathbf{assume}(t = 1);\\ \quad \left\{\left(\begin{array}{l}c \mapsto 0 \wedge t = 0 \vee\\ c \mapsto 1 * x \mapsto 1 \wedge t = 1\end{array}\right) \wedge t = 1\right\}\\ \quad \{c \mapsto 1 * x \mapsto 1\}\\ \quad [c] := 0;\\ \quad \{c \mapsto 0 * x \mapsto 1\}\\ \quad \{x \mapsto 1 * J\}\\ \}\\ \{x \mapsto 1\}\\ t := [x]\\ \{x \mapsto 1 \wedge t = 1\}\end{array}\right)\\ \quad \{\mathbf{emp} * (x \mapsto 1 \wedge t = 1)\}\\ \quad \{x \mapsto 1 \wedge t = 1\}\\ \{(x \mapsto 1 \wedge t = 1) * J\}\\ \{c \mapsto 0 * x \mapsto 1 \wedge t = 1\}\end{array}$$

3 Fractional Permissions

Fractional permissions allow multiple concurrent readers of a memory cell. For example:

$$
\begin{array}{c}
\{x \mapsto 5\} \\
\left\{x \overset{0.3}{\mapsto} 5 * x \overset{0.7}{\mapsto} 5\right\} \\
\left(\begin{array}{l|l}
\left\{x \overset{0.3}{\mapsto} 5\right\} & \left\{x \overset{0.7}{\mapsto} 5\right\} \\
t := [x]; & u := [x]; \\
\left\{x \overset{0.3}{\mapsto} 5 \wedge t = 5\right\} & \left\{x \overset{0.7}{\mapsto} 5 \wedge u = 5\right\}
\end{array}\right) \\
\left\{(x \overset{0.3}{\mapsto} 5 \wedge t = 5) * (x \overset{0.7}{\mapsto} 5 \wedge u = 5)\right\} \\
\{x \mapsto 5 \wedge t = 5 \wedge u = 5\}
\end{array}
$$

Using fractional permissions, we can also reason about a system where one distinguished thread writes to $[x]$ atomically and knows its value, while other threads simply read from $[x]$ atomically, but cannot assert anything about its value. The trick is to give half of the permission to the resource invariant and keep the other half in the writing thread. Here is an illustration:

$$
\begin{array}{l}
\{x \mapsto 5\} \\
\left\{x \overset{0.4}{\mapsto} 6 * x \overset{0.6}{\mapsto} -\right\} \\
\quad \left\{x \overset{0.4}{\mapsto} 5\right\} \\
x \overset{0.6}{\mapsto} - \vdash \left(\begin{array}{l|l}
\left\{x \overset{0.4}{\mapsto} 5\right\} & \\
t := [x]; & \\
\left\{x \overset{0.4}{\mapsto} 5 \wedge t = 5\right\} & \{\mathbf{emp}\} \\
\mathbf{atomic}\ \{ & \mathbf{atomic}\ \{ \\
\quad \left\{(x \overset{0.4}{\mapsto} 5 \wedge t = 5) * x \overset{0.6}{\mapsto} -\right\} & \quad \left\{x \overset{0.6}{\mapsto} -\right\} \\
\quad \{x \mapsto 5 \wedge t = 5\} & \quad u := [x]; \\
\quad [x] := t + 1; & \quad \left\{x \overset{0.6}{\mapsto} -\right\} \\
\quad \left\{x \mapsto 6 \wedge t = 5\right\} & \} \\
\quad \left\{x \overset{0.4}{\mapsto} 6 * x \overset{0.6}{\mapsto} -\right\} & \{\mathbf{emp}\} \\
\} & \\
\left\{x \overset{0.4}{\mapsto} 6\right\} &
\end{array}\right) \\
\quad \left\{x \overset{0.4}{\mapsto} 6\right\} \\
\left\{x \overset{0.4}{\mapsto} 6 * x \overset{0.6}{\mapsto} -\right\} \\
\{x \mapsto 6\}
\end{array}
$$