WS 2020 / 2021 — 176.12.2020

Übungen zur Vorlesung
Modern Concurrency Theory
Blatt 6

Prof. Dr. Roland Meyer
Sebastian Wolff

Abgabe bis 11.01.2021 um 10:00 Uhr

Aufgabe 6.1 (Separation Logic)

Zeigen Sie, dass Regel (GLOOKV) aus Regel (LLOOKV) abgeleitet werden kann.

Hinweis: Sie müssen die Vor- und Nachbedingungen von (GLOOKV) *äquivalent umschreiben, sodass die „fehlenden" Variablen eingeführt werden. Des Weiteren sind die Regeln* AUX *und* FRAME *hilfreich.*

Aufgabe 6.2 (Merge)

Geben Sie eine Implementierung der Funktion GETLAST an, welche das letzte Element einer nicht leeren Liste liefert. Genauer: vervollständigen Sie folgende Vorlage

```
procedure GETLAST(x)
    local y, ...
    c
    return y
```

und beweisen Sie die Gültigkeit folgender Aussage:

$$\{\, \mathit{list}(x) \,\}\; c \;\{\, \exists \alpha \exists u.\ \mathit{lseg}_{\alpha}(x, y) * \mathit{lseg}_{u}(y, \mathit{nil}) \,\}$$

wobei $\mathit{list}(x) := \exists \alpha.\ \alpha \neq \epsilon * \mathit{lseg}_{\alpha}(x, \mathit{nil})$. Hier erlauben wir Quantoren über Terme α. Benutzen Sie dies, um eine adäquate Schleifeninvariante zu formulieren.

Aufgabe 6.3 (Bi-Abduktion)

Betrachten Sie folgende Implementierung von MERGE

```
procedure MERGE(x, y)
    local z
    z := GETLAST(x)
    [z + 1] := y
    return x
```

Wenden Sie das Verfahren aus der Vorlesung an, um mittels Bi-Abduktion die notwendige Vorbedingungen für MERGE zu berechnen, sodass $\mathit{list}(x)$ eine Nachbedingung ist.

Aufgabe 6.4 (Concurrent Separation Logic I)
Wir definieren eine einfache Lock Implementierung:

$$
\begin{array}{ll}
\begin{array}{l}
\mathbf{lock}_i \equiv \\
\quad \textbf{atomic} \\
\qquad t_i := [l] \\
\qquad \textbf{assume } t_i = 0 \\
\qquad [l] := 1
\end{array}
&
\qquad\qquad
\begin{array}{l}
\mathbf{unlock}_i \equiv \\
\quad \textbf{atomic} \\
\qquad t_i := [l] \\
\qquad \textbf{assume } t_i = 1 \\
\qquad [l] := 0
\end{array}
\end{array}
$$

Sei I eine Assertion und $i \in \mathbb{N}$.

a) Definieren Sie eine Resourcen-Invariante R mit folgender Eigenschaft:

$$R \vdash \{\, emp \,\}\, \mathbf{lock}_i \,\{\, I \,\} \qquad \text{und} \qquad R \vdash \{\, I \,\}\, \mathbf{unlock}_i \,\{\, emp \,\}$$

b) Zeigen Sie, dass folgende Regel sound ist:

$$\frac{R \vdash \{\, P * I \,\}\, C \,\{\, Q * I \,\}}{R \vdash \{\, P \,\}\, \mathbf{lock}_i; C; \mathbf{unlock}_i \,\{\, Q \,\}}$$

Aufgabe 6.5 (Concurrent Separation Logic II)
Betrachten Sie das Programm $c \equiv c_1 \parallel c_2$.

a) Können Sie für

$$\begin{array}{rrcl} & c_1 & \equiv & \textbf{atomic}\ \{\ \ y := [x];\ [x] := y + 1;\ [x+1] := 1\ \ \} \\ \text{und} & c_2 & \equiv & \textbf{atomic}\ \{\ \ z := [x];\ [x] := z + 1;\ [x+2] := 1\ \ \}\ . \end{array}$$

nachweisen, dass c den Heap von $x \mapsto 0, 0, 0$ nach $x \mapsto 2, 1, 1$ überführt? Falls dem so ist, zeigen Sie die Gültigkeit von $J \vdash \{\, A \,\}\, c \,\{\, B \,\}$ für geeignete J, A, B. Andernfalls argumentieren Sie, warum Sie die Gültigkeite nicht mittels Concurrent Separation Logic nachweisen können.

b) Begründen Sie informell für

$$\begin{array}{rrcl} & c_1 & \equiv & \mathbf{lock}_1;\ y := [x];\ [x] := y + 1;\ [x+1] := 1;\ \mathbf{unlock}_1 \\ \text{und} & c_2 & \equiv & \mathbf{lock}_2;\ z := [x];\ [x] := z + 1;\ [x+2] := 1;\ \mathbf{unlock}_2\ . \end{array}$$

was sich im Vergleich zu Aufgabenteil a) an Ihrem Beweis bzw. Argument ändert.

Abgabe bis 11.01.2021 um 10:00 Uhr per Mail an sebastian.wolff@tu-bs.de.