Übungen zur Vorlesung
Modern Concurrency Theory
Blatt 6

Prof. Dr. Roland Meyer
Sören van der Wall Abgabe bis 09.01.2022 um 23:59

Aufgabe 6.1 (Concurrent Separation Logic and Race Conditions)

Betrachten Sie die folgenden Codestücke.

```
1: Let A be
2:     atomic
3:         a = [x]
4:
5: Let C be
6:     atomic
7:         [x] = a
8:
```

```
1: Let B be
2:     atomic
3:         b = [x]
4:
5: Let D be
6:     atomic
7:         [x] = b
8:
```

Gibt es eine Resource-Invariant R für die jeweiligen Judgements? Falls ja, geben Sie sie an. Sonst begründen Sie informell, wieso es keine geben kann.

1. $R \vdash \{\, a = 0 \wedge b = 1 * x \mapsto \_ \,\} C \,||\, D \{\, x \mapsto \_ \,\}$
2. $R \vdash \{\, a = 0 \wedge b = 1 * x \mapsto \_ \,\} A; B \{\, a = b \,\}$
3. $R \vdash \{\, a = 0 \wedge b = 1 * x \mapsto \_ \,\} A \,||\, B \{\, a = b \,\}$
4. $R \vdash \{\, a = 0 \wedge b = 1 * x \mapsto \_ \,\} A \,||\, C \{\, x \mapsto \_ \,\}$

Aufgabe 6.2 (Resource Invariants for Locks)

Betrachten Sie die folgende Implementierung eines Read&Write-Locks, wobei $i \in \mathbb{N}$ jeweils unterschiedliche Stackframes und damit unterschiedliche Stackvariablen beschreiben soll.

```
1: Let LOCK_i[l] be
2:     atomic
3:         z_i = [l]
4:         Assume z_i == 0
5:         [l] = 1
6:
```

```
1: Let UNLOCK_i[l] be
2:     atomic
3:         z_i = [l]
4:         Assume z_i == 1
5:         [l] = 0
6:
```

1. Zeigen Sie für eine beliebige Assertion A, dass es eine Resource-Invariante R gibt, sodass

$$R \vdash \{\, emp \,\} \textsc{lock}_i[l] \{\, A \,\} \qquad R \vdash \{\, A \,\} \textsc{unlock}_i[l] \{\, emp \,\}$$

2. Zeigen Sie, dass sich daraus die folgende Beweisregel ableiten lässt:

$$\frac{R \vdash \{\, X * A \,\} C \{\, Y * A \,\}}{R \vdash \{\, X \,\} \text{LOCK}_i[l]; C; \text{UNLOCK}_i[l] \{\, Y \,\}}$$

Hinweis: Intuitiv überträgt ein Lock das Ownership auf einem Teil des States an den Thread, welchem das Lock gehört.

Aufgabe 6.3 (Coarse-Grained Stack)

Betrachten Sie den folgenden Code eines coarse-grained, concurrent Stacks mit den Stackvariablen $\texttt{tos\_adr}, \texttt{tos}_i, \texttt{new\_tos}_i, \texttt{rem}_i, \texttt{next}_i$. Hierbei ist $\text{LOCK}_i[l]$ keine Funktion, sondern ein Makro, also eine textuelle Ersetzung von Code.

```
1: Let PUSH_i[v] be
2:     LOCK_i[l]
3:     tos_i = [tos_adr]
4:     new_tos_i = cons(v, tos_i)
5:     [tos_adr] = new_tos_i
6:     UNLOCK_i[l]
7:
```

```
1: Let POP_i be
2:     LOCK_i[l]
3:     next_i = [tos + 1]
4:     Assume next_i ≠ nil
5:     rem_i = tos
6:     tos = next_i
7:     UNLOCK_i[l]
8:     DISPOSE(rem_i)
9:
```

1. Drücken sie mittels einer Assertion $\mathit{StackRef}(x)$ aus, dass x auf einen Wert y zeigt und sich an Addresse y ein Stack befindet. Sie dürfen über Listen quantifizieren.

2. Zeigen Sie, dass das folgende SL-Tripel gilt:

$$\{\mathit{StackRef}(\texttt{tos\_adr})\} \vdash \{\, \mathit{emp} \,\} \text{PUSH}_1[v] \,||\, \text{POP}_2 \{\, \mathit{emp} \,\}$$

Hinweis: Nutzen Sie die Beweisregel aus der vorherigen Aufgabe, um sich Ownership der Resource-Invariant zu leihen.

Abgabe bis 09.01.2022 um 23:59 unter `https://cloudstorage.tu-braunschweig.de/preparefilelink?folderID=2HcdWaXvj1DZBMNXGWRVX.`