Universität Bremen

Korrekte Software: Grundlagen und Methoden SoSe 2020

Serge Autexier
Christoph Lüth

7. Übungsblatt

Ausgabe: 25.06.20
Abgabe: 02.07.20

Dieses Übungsblatt ist ein PDF-Formular. Bitte in einem PDF-Viewer Ihrer Wahl ausfüllen, abspeichern, und an die Veranstalter mailen.

Gruppe: 1 ☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐ 6 ☐ 7 ☐ 8 ☐ 9 ☐ 10 ☐ 11 ☐ 12 ☐

Name: Matrikelnummer:

Name: Matrikelnummer:

Name: Matrikelnummer:

7.1 *Suche nach dem größten Element*

Das folgende Programm aus der Vorlesung berechnet den Index r des größten Elementes des nicht-leeren Feldes a.

```
1  int n;
2  int a[];
3  int i, n, r;
4
5  /** { 0 < n }
6  i= 0;
7  r= 0;
8  while (i< n)
9     /** inv (∀ j. 0<= j< i ⟶ a[j]<= a[r])
10              && 0 <= i <= n && 0 <= r < n; */ {
11    if (a[r] < a[i]) {
12      r= i;
13      }
14    else {
15      }
16    i= i+1;
17    }
18 /** { (∀ j. 0 <= j< n⟶ a[j] <= a[r]) && 0 <= r< n } */
```

Berechnen Sie die stärkste Nachbedingung und die resultierenden Verifikationsbedingungen nach dem bekannten Muster.

1. Stärkste Nachbedingung (ASP):

2. Verifikationsbedingungen (SVC):

7.2 *Spezifikation*

Zeichenketten sind in C einfach Felder von **char**. Wir können sie zu Sequenzen von Zeichen, oder Zeichenketten, abstrahieren, so wie wir Feldern von ganzen Zahlen zu Sequenzen (Listen) von ganzen Zahlen abstrahiert haben.

In dieser Übung wollen wir ine nützliche Funktion auf Zeichenketten spezifizieren. Dazu definieren wir ein *Wort* als eine Zeichenkette, die kein Leerzeichen[1] enthält.

Als Beispiel für rekursive Funktionsdefinitionen können die Funktionen aus der Vorlesung dienen, oder folgende Funktion, welche aus einer Liste von Listen eine lange Liste macht:

$$concat([]) == []$$
$$concat(xs : xss) == xs{+}{+}(concat(xss))$$

Wir spezifizieren jetzt eine Funktion, die eine Zeichenkette in eine Liste von Wörtern zerlegt.

(1) Spezifizieren Sie ein Prädikat nospace(w), welches wahr ist gdw. das Argument w kein Leerzeichen enthält (also ein Wort ist). Nutzen Sie dazu das Prädikat is_space(c), welches wahr ist, wenn das Argument c ein Leerzeichen ist:

(2) Spezifizieren Sie eine Funktion remspace(s), welche alle Leerzeichen aus einer Zeichenkette entfernt:

(3) Spezifzieren Sie, dass das Ergebnis von remspace(s) keine Leerzeichen enthält und damit ein Wort ist:

(4) Spezifizieren Sie damit eine Funktion words(w), welche aus einer Zeichenketten (Listen von Zeichen w) eine Liste von Zeichenketten ws erzeugt, so dass:

- alle Elemente von ws Wörter sind, und
- sie verkettet daselbe ergeben wie w ohne die Leerzeichen.

Hinweis:

Nutzen Sie die in der Vorlesung vorgestellten Funktionen $++$, : etc. auf Sequenzen; ggf. können Sie auch weitere Hilfsfunktionen definieren. Beachten Sie, dass Ihre Definitionen nicht alle rekursiv bzw. ausführbar zu sein brauchen.

Die Funktionen concat und words gibt es mit der hier intendierten Funktionalität auch in Haskell.

[1] "white space", also Leerzeichen, Tabulator oder Zeilenvorschub