Übung zur Vorlesung

Automatisierte Logik und Programmierung

Prof. Chr. Kreitz

Universität Potsdam, Theoretische Informatik — Sommersemester 2004

Blatt 4 — Abgabetermin: —

Aufgabe 4.1 (Entwurfsentscheidungen bei Divide & Conquer)

Gegeben sei die bekannte Spezifikation des Sortierproblems

`FUNCTION sort(L:Seq(ℤ)):Seq(ℤ)  RETURNS S  SUCH THAT rearranges(L,S) ∧ ordered(S)`

Bei der Synthese des Mergesort-Algorithmus haben wir mit der Wahl der binären Dekompositionsfunktion `ListSplit` mit $\mathtt{O}_D\mathtt{(L,(L_1,L_2))} \mathrel{\hat{=}} \mathtt{L=L_1 \circ L_2}$ begonnen und hierzu passend die Kompositionsfunktion generiert. Leiten Sie informal andere Sortieralgorithmen ab, indem Sie

4.1–a als Dekompositionsfunktion die Zerlegung `FirstRest` mit $\mathtt{O}_D\mathtt{(L,(a,L'))} \mathrel{\hat{=}} \mathtt{L=a.L'}$ wählen und hierzu passend eine Kompositionsfunktion generieren.

4.1–b als Kompositionsfunktion die Listenerzeugungsfunktion `cons` mit $\mathtt{O}_C\mathtt{((a',S'),S)} \mathrel{\hat{=}} \mathtt{S=a'.S'}$ wählen und hierzu passend eine Dekompositionsfunktion generieren.

4.1–c als Kompositionsfunktion die Listenerzeugungsfunktion `append` mit $\mathtt{O}_C\mathtt{((S_1,S_2),S)} \mathrel{\hat{=}} \mathtt{S=S_1 \circ S_2}$ wählen und hierzu passend eine Dekompositionsfunktion generieren.

Erklären Sie die entstehenden Anforderungen an die Komponenten des Divide & Conquer Algorithmus, und geben Sie dann die entsprechenden Komponenten und den Sortieralgorithmus an. Geben Sie eine kurze Begründung für die Korrektheit Ihrer Wahl. Eine formale Herleitung ist nicht erforderlich.

Aufgabe 4.2 (Synthese von Divide & Conquer Algorithmen)

Erzeugen Sie mithilfe der formalen Synthesestrategie für Divide & Conquer Algorithmen den Quicksort-Algorithmus für das Sortieren von Listen ganzer Zahlen. Welche Lemmata benötigt die Strategie, um die Komponenten herzuleiten?

Hinweise: Wie im Falle des Mergesort-Algorithmus der Vorlesung muß man in zwei Phasen vorgehen, da der Quicksort-Algorithmus eine nichttriviale Dekomposition besitzt. Berücksichtigen Sie auch, daß Quicksort in einem gewissem Sinne invers zu Mergesort arbeitet, also die Dekomposition invers zur Komposition von Mergesort operiert, während die Komposition (invers zur Dekomposition von Mergesort) verhältnismäßig einfach ist und als Ausgangspunkt genommen werden sollte.

Aufgabe 4.3 (Lokalsuche)

Synthetisieren Sie informal den Simplex Algorithmus als Lokalsuch–Algorithmus

```
FUNCTION Simplex(A:Matrix[n,m], b:  Vector[m], c:Vector[n]):Vector[n]
  RETURNS x  SUCH THAT A * x ≤ b ∧ ∀y:Vector[n].(A*y ≤ b ⇒  c*y≤c*x)
```

Wie könnten die Nachbarschaftsstruktur und geeignete Filter aussehen?