Übung zur Vorlesung

Automatisierte Logik und Programmierung

Prof. Chr. Kreitz

Universität Potsdam, Theoretische Informatik — Sommersemester 2006

Blatt 4 — Abgabetermin: —

Aufgabe 4.1

In der Vorlesung haben wir die Grundbegriffe der Programmsynthese semi-formal beschrieben. Für eine formale Programmsynthese innerhalb eines Beweissystems müssen diese Konzepte formalisiert werden. Formalisieren Sie die folgenden Begriffe innerhalb der CTT.

4.1–a Die Klasse aller Spezifikationen als ein Datentyp `SPECIFICATIONS`

4.1–b Die Klasse aller Programme als ein Datentyp `PROGRAMS`

4.1–c Programmkorrektheit als ein "Prädikat" p `ist korrekt` *(beachten Sie Terminierung!)*

4.1–d Erfüllbarkeit von Spezifikationen als ein "Prädikat" *spec* `ist erfüllbar`

4.1–e Die Notation für Programme
`FUNCTION` f`(x:D):R WHERE I(x) RETURNS y SUCH THAT O(x,y) =` `body(x)`

Aufgabe 4.2 (Synthese durch Transformationen)

Gegeben sei die folgende Spezifikation des Sortierproblems

`FUNCTION` *sort*`(L:Seq(`$\mathbb{Z}$`)):Seq(`$\mathbb{Z}$`)  WHERE true`
`  RETURNS S  SUCH THAT rearranges(L,S)` $\wedge$ `ordered(S)`

wobei das Prädikat `ordered` wie folgt definiert sei

`ordered(`S`)` $\equiv$ $\forall$`i`$\in$`{1..|`S`|-1}.`S`[i]`$\leq$$S$`[i+1]`

Synthetisieren Sie ein Sortierprogramm durch Anwendung von Transformationen.
Stellen Sie dazu eine Spezifikationsformel auf und transformieren Sie die Ausgabebedingung durch Äquivalenzumformungen bis ein rekursives Sortierprogramm entsteht. Begründen Sie, warum das erzeugte Programm terminiert.

Hinweise: Je nach Art der Umformungen können sehr unterschiedliche Algorithmen erzeugt werden. Stellen Sie die nötigen Lemmata, die nicht bereits im Appendix B des Skriptes zu finden sind, separat auf, ohne sie formal zu beweisen.

Aufgabe 4.3 (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.