Universität Karlsruhe (TH)
Lehrstuhl für Programmierparadigmen

Theorembeweiser und ihre Anwendungen SS 2009 http://pp.info.uni-karlsruhe.de/
Übungsleiter: Daniel Wasserrab wasserra@ipd.info.uni-karlsruhe.de

Übungsblatt 4 Besprechung: 19.05.2009

Rekursive Datenstrukturen

In dieser Übung soll eine rekursive Datenstruktur für Binärbäume erstellt werden. Außerdem sollen Funktionen über Binärbäume definiert und Aussagen darüber gezeigt werden. Dazu dürfen Sie auch automatische Taktiken, z.B. *auto* verwenden. Und denken Sie daran: *Recursion is proved by induction!*.

Als erstes definieren Sie den Datentypen für Binärbäume. Sowohl Blätter als auch innere Knoten speichern Information. Der Typ der Information soll beliebig sein, also arbeiten sie mit Typparameter $'a$.

datatype $'a\ tree = \ldots$

Definieren Sie jetzt die Funktionen *preOrder*, *postOrder* und *inOrder*, welche einen $'a\ tree$ in der entsprechenden Ordnung durchlaufen:

consts
$preOrder :: 'a\ tree \Rightarrow 'a\ list$
$postOrder :: 'a\ tree \Rightarrow 'a\ list$
$inOrder :: 'a\ tree \Rightarrow 'a\ list$

Als nächstes definieren Sie eine Funktion *mirror*, welche das Spiegelbild eines $'a\ tree$ zurückgibt.

consts $mirror :: 'a\ tree \Rightarrow 'a\ tree$

Seien *xOrder* und *yOrder* Verfahren zum Durchlaufen von Bäumen, beliebig ausgewählt aus *preOrder*, *postOrder* und *inOrder*. Formulieren und zeigen Sie alle gültigen Eigenschaften der Art $xOrder\ (mirror\ xt) = rev\ (yOrder\ xt)$.

Definieren Sie die Funktionen *root*, *leftmost* und *rightmost*, welche die Wurzel, das äußerst links bzw. das äußerst rechts gelegene Element zurückgeben.

consts
$root :: 'a\ tree \Rightarrow 'a$
$leftmost :: 'a\ tree \Rightarrow 'a$
$rightmost :: 'a\ tree \Rightarrow 'a$

Zum Schluss beweisen Sie noch folgende Theoreme oder zeigen ein Gegenbeispiel (dazu kann man u.a. *quickcheck* verwenden). Es kann nötig sein, erst bestimmte Hilfslemmas zu beweisen.

theorem $last\ (inOrder\ xt) = rightmost\ xt$
theorem $hd\ (inOrder\ xt) = leftmost\ xt$
theorem $hd\ (preOrder\ xt) = last\ (postOrder\ xt)$
theorem $hd\ (preOrder\ xt) = root\ xt$
theorem $hd\ (inOrder\ xt) = root\ xt$
theorem $last\ (postOrder\ xt) = root\ xt$