2. Übungsblatt

Ausgabe: 17.11.05
Abgabe: 01.12.05

Die Universität Bremen erbittet Ihre Mithilfe!

Der Veranstalter würde in den nächsten Vorlesungen gerne einen Beweis der Korrektheit von Mergesort in Isabelle formalisieren. Leider fehlen ihm noch einige wichtige Lemmata, und er ist mit dem Ausfüllen von studienbegleitenden Leistungsnachweisen und Dienstreiseanträgen mehr als überlastet. Wir bitten Sie deshalb, in diesem Übungsblatt die fehlenden Beweise zu erarbeiten.

3 *Elemente zählen* *10 Punkte*

Definieren Sie eine Funktion

```
count :: "'a=> 'a list => nat"
```

welche zählt, wie oft ein Element in einer Liste auftritt, und beweisen Sie ein Lemma `count_concat`, welches besagt, dass ein Element in der Verkettung zweier Listen so oft auftritt wie in den beiden Listen zusammen.

4 *Permutationen* *10 Punkte*

Mit Hilfe der Funktion `count` definieren Sie jetzt eine Relation

```
perm :: "'a list=> 'a list => bool" (infixr "~~" 65)
```

die besagt, ob eine Liste einer Permutation einer anderen ist. Informell ist eine Liste eine Permutation einer anderen, wenn alle Elemente in beiden Listen gleich oft auftreten.

Zeigen Sie dann, dass `perm` eine Äquivalenzrelation ist (d.h. reflexiv, symmetrisch und transitiv), sowie eine Kongruenz bezüglich der Listenkonkatenation: wenn `a` eine Permutation von `x` ist, und `b` eine Permutation von `y`, dann ist `a @ b` eine Permutation von `x @ y`.

Hinweise:

- Um in einer Taktik, speziell in einer Resolution, eine bestimmte Instantiierung einer Variablen zu erzwingen, benutzen wir die Varianten `rule_tac`, `erule_tac` und `drule_tac` mit folgender Notation (um beispielsweise die Variable `x` mit `foo` zu instantiieren): `apply (erule_tac x= "foo" in allE)`
- Fallunterscheidungen werden mit der Taktik `cases` eingeführt; zum Beispiel erzeugt `apply (cases "x= 0")` zwei neue Beweisziele, mit den jeweiligen Voraussetzungen `x= 0` und `x ~= 0`.

Dies ist Version 1.0 vom 2005/11/16 17:14:49.