Prof. Dr. A. Poetzsch-Heffter
Dipl.-Inform. J. O. Blech
Dipl.-Inform. M. J. Gawkowski

TU Kaiserslautern
Fachbereich Informatik
AG Softwaretechnik

Übungsblatt 1: Spezifikation und Verifikation mit Logik Höherer Ordnung (Sommersemester 2008)

Ausgabe: 8. April 2007

Aufgabe 1 Sortieralgorithmen in ML

Implementieren Sie ein Sortierverfahren Ihrer Wahl auf Listen von ganzen Zahlen in ML. Ihr Verfahren soll so sortieren, dass kleine Elemente in der Liste weiter vorne stehen.

Entwerfen Sie jetzt ein Prädikat, das testet, ob eine gegebene Liste eine korrekte Sortierung einer gegebenen anderen Liste ist. Implementieren Sie Ihr Prädikat in ML.

Aufgabe 2 Mengen in ML

In dieser Aufgabe sollen Sie Operationen auf Mengen für einen Typ `''a` in ML implementieren. Eine Menge soll als Funktion dargestellt werden. Sie soll einen Wert vom Typ `''a` nehmen und als Ergebnis einen bool'schen Wert liefern, der anzeigt, ob der `''a` Wert in der Menge liegt.

a) Schreiben Sie eine Funktion, die die leere Menge repräsentiert.

b) Schreiben Sie eine Higher-Order Funktion, die ein Element in eine Menge einfügt.

c) Schreiben Sie eine Higher-Order Funktion, die ein Element aus einer Menge löscht.

d) Schreiben Sie Higher-Order Funktionen für Durchschnitt und Vereinigung von Mengen.

e) Spezifizieren Sie die Menge aller geraden ganzen Zahlen.

Aufgabe 3 Listen in Isabelle/HOL

a) Definieren Sie eine Funktion `appendright`, die ein Element hinten an eine Liste dranhängt. Benutzen Sie nicht den `@`-Operator.

```
consts
  appendright :: "'a list => 'a => 'a list"
```

Beweisen Sie jetzt das folgende Theorem in Isabelle/HOL.

```
theorem rev_cons: "rev (x # xs) = appendright (rev xs) x"
```

b) Definieren Sie eine Funktion `replace` mit der Eigenschaft, dass `replace x y zs`, die Liste `zs` so verändert zurückliefert, dass alle Vorkommen von `x` durch `y` ersetzt worden sind.

```
consts replace :: "'a => 'a => 'a list => 'a list"
```

Beweisen oder wiederlegen Sie jetzt die folgenden Theoreme:

```
theorem "rev(replace x y zs) = replace x y (rev zs)"
(*<*)oops(*>*)

theorem "replace x y (replace u v zs) = replace u v (replace x y zs)"
(*<*)oops(*>*)

theorem "replace y z (replace x y zs) = replace x z zs"
(*<*)oops(*>*)
```

c) In dieser Aufgabe sollen Sie Quantoren auf Listen entwerfen und mit ihnen arbeiten. Definieren Sie einen All- und einen Existenzquantor auf Listen. Die Quantoren sollen jeweils als primitiv rekursive Funktionen realisiert werden und ein Prädikat, sowie eine Liste als Argumente bekommen. Der All-Quantor soll genau dann `true` als Ergebnis liefern, wenn alle Elemente in der übergebenen Liste das Prädikat erfüllen. Der Existenz-Quantor wird analog definiert.

```
consts
  alls :: "('a => bool) => 'a list => bool"
  exs  :: "('a => bool) => 'a list => bool"
```

Beweisen oder widerlegen Sie folgende Aussagen:

```
lemma "alls (% x. P x & Q x) xs = (alls P xs & alls Q xs)"
(*<*)oops(*>*)

lemma "alls P (rev xs) = alls P xs"
(*<*)oops(*>*)

lemma "exs (% x. P x & Q x) xs = (exs P xs & exs Q xs)"
(*<*)oops(*>*)

lemma "exs P (map f xs) = exs (P o f) xs"
(*<*)oops(*>*)

lemma "exs P (rev xs) = exs P xs"
(*<*)oops(*>*)
```

Wie können Sie den Existenz-Quantor (auf Listen) mit Hilfe eines All-Quantors ausdrücken?