Abgabe: 20.01.2009 (vor der Vorlesung)

Aufgabe 12.1 (H) Verifikation funktionaler Programme

Es seien folgende, bekannte Definitionen gegeben:

```
let rec len = fun l ->
  match l with
    []     -> 0
  | x::xs -> 1 + len xs

let rec app = fun l1 l2 ->
  match l1 with
    []     -> l2
  | x::xs -> x :: app xs l2

let rec sorted = fun l ->
  match l with
    []     -> true
  | x::xs ->
      match xs with
        []    -> true
      | y::_ -> x <= y & sorted xs

let rec insert = fun y l ->
  match l with
    []     -> [y]
  | x::xs ->
      match x <= y with
        true  -> x :: insert y xs
      | false -> y::x::xs

let rec sort = fun l ->
  match l with
    []     -> []
  | [x]    -> [x]
  | x::xs -> insert x (sort xs)
```

a) Zeigen Sie, dass unter der Voraussetzung, dass alle Aufrufe terminieren,

$$\texttt{len}\,(\texttt{app}\;\texttt{l1}\;\texttt{l2}) = \texttt{len}\;\texttt{l1} + \texttt{len}\;\texttt{l2}$$

für alle Listen `l1`, `l2` gilt.

b) Zeigen Sie, dass unter der Voraussetzung, dass alle Aufrufe terminieren,

$$\text{Aus } \texttt{sorted}\;\texttt{l} \text{ folgt } \texttt{sorted}\,(\texttt{insert}\;\texttt{y}\;\texttt{l}) \tag{1}$$

für alle `y`, `l` gilt. **Hinweis:** Im Induktionsschritt müssen sie verschiedene Fälle unterscheiden. Bei diesen Fällen kommt es auf die Beziehung zwischen `y` und den ersten beiden Elementen der Liste `l` an.

c) Zeigen Sie, dass unter der Voraussetzung, dass alle Aufrufe terminieren,

$$\texttt{sorted}\ (\texttt{sort}\ \texttt{xs}) = \mathbf{true}$$

für alle Listen `xs` gilt.

Aufgabe 12.2 (P) app und rev

Gegeben sei folgende MiniOcaml-Funktion zur Umkehrung einer Liste

```
let rec rev = fun list ->
  match list with
    [] -> []
  | el :: rest -> app (rev rest) [el]
```

sowie die in der Vorlesung wie folgt definierte Funktion `app`

```
let rec app = fun x -> fun y ->
  match x with
    [] -> y
  | x::xs -> x :: app xs y
```

Zeigen Sie unter der Annahme, dass alle vorkommenden Funktionsaufrufe terminieren, dass folgende Aussagen gelten:

a) `rev (app xs ys) = app (rev ys) (rev xs)`

b) `rev (rev list) = list`

Hinweis: Folgende Aussagen sind bereits in der Vorlesung bewiesen worden:

$$\text{app x (app y z)} = \text{app (app x y) z} \tag{2}$$

$$\text{app x []} = \text{x} \tag{3}$$