Informatik I, Blatt Nr. 8, Abgabe: gar nicht um 11 Uhr
http://proglang.informatik.uni-freiburg.de/teaching/info1/2009/

Auf vielfachen Wunsch gibt es noch eine Listeninduktionsaufgabe.

1 Aufgabe

Seien folgende Prozeduren definiert:

```
; bildet Liste (a1 a2 a3 ... an ) auf (a1 a1 a2 a2 a3 a3 ... an an) ab
(: stutter ((list %a) -> (list %a)))
(define stutter
  (lambda (xs)
    (if (empty? xs)
        empty
        (make-pair (first xs) (make-pair (first xs) (stutter (rest xs)))))))

; Laenge einer Liste
(: len ((list %a) -> natural))
(define len
  (lambda (xs)
    (if (empty? xs)
        0
        (+ 1 (len (rest xs))))))
```

Beweisen Sie nun folgendes per struktureller Induktion:

Für alle Listen L gilt: `(len (stutter` L`))` $= 2 \cdot$ `(len` L`)`

Lösung

Beweis per struktureller Induktion über L.
Wir verwenden `m-p` als Abkürzung für `make-pair`.

Induktionsanfang $L =$ `empty`. Zu zeigen: `(len (stutter empty))` $= 2 \cdot$ `(len empty)`.

L.S. = `(len (if (empty? empty) empty ...))` (Einsetzen `stutter`)
= `(len (if #t empty ...))` (Def. `empty?`)
= `(len empty)` (Def. `if`)
= `(if (empty? empty) 0 ...)` (Def. `len`)
= `(if #t 0 ...)` (Def. `empty?`)
= 0 (Def. `if`)
R.S. = $2 \cdot$ `(len empty)`
= $2 \cdot 0$ (schon oben gezeigt: `(len empty)` $= 0$)
= L.S.

Induktionsschritt $L \to$ `(make-pair` x L`)`.
Gelte bereits die **Induktionsannahme**: `(len (stutter` L`))` $= 2 \cdot$ `(len` L`)`.
Zu zeigen: $\forall x.$`(len (stutter (m-p` x L`)))` $= 2 \cdot$ `(len (m-p` x L`))`.
Sei also x ein beliebiger Wert. Definiere zunächst $L' :=$ `(m-p` x L`)`.

$$
\begin{aligned}
\text{L.S.} &= \texttt{(len (stutter } L'\texttt{))} \\
&= \texttt{(len (if (empty? } L'\texttt{) ...(m-p ...)))} \qquad \text{(Def. } \texttt{stutter}\text{)} \\
&= \texttt{(len (m-p (first } L'\texttt{) (m-p (first } L'\texttt{) (stutter (rest } L'\texttt{)))))} \qquad \text{nach Def. von } \texttt{empty?}, \texttt{if} \\
&= \texttt{(len (m-p } x \texttt{ (m-p } x \texttt{ (stutter } L\texttt{))))} \qquad \text{nach Def. } \texttt{first}, \texttt{rest} \\
&= \texttt{(if (empty? } S\texttt{) 0 (+ 1 (len (rest } S\texttt{))))} \qquad \text{nach Def. } \texttt{len} \\
&\quad \text{mit } S := \texttt{(m-p } x \texttt{ (m-p } x \texttt{ (stutter } L\texttt{)))} \\
&= \texttt{(+ 1 (len (rest } S\texttt{)))} \qquad \text{mit } \texttt{if}, \texttt{empty?} \\
&= 1 + \texttt{(len (m-p } x \texttt{ (stutter } L\texttt{)))} \qquad \text{mit } \texttt{+}, \texttt{rest} \\
&= 1 + \texttt{(if (empty? } S'\texttt{) 0 (+ 1 (len (rest } S'\texttt{))))} \qquad \text{Def. } \texttt{len} \\
&\quad \text{mit } S' := \texttt{(m-p } x \texttt{ (stutter } L\texttt{))} \\
&= 1 + \texttt{(+ 1 (len (rest } S'\texttt{)))} \qquad \text{mit } \texttt{empty?}, \texttt{if} \\
&= 1 + \texttt{(+ 1 }\underline{\texttt{(len (stutter } L\texttt{))}}\texttt{)} \qquad \text{nach Einsetzen von } S' \text{ und mit } \texttt{rest} \\
&= 1 + \texttt{(+ 1 }\underline{2 \cdot \texttt{(len } L\texttt{)}}\texttt{)} \qquad \text{per Induktionsannahme} \\
&= 1 + 1 + 2 \cdot \texttt{(len } L\texttt{)} \qquad \text{nach Def. } \texttt{+} \\
&= 2 \cdot (1 + \texttt{(len } L\texttt{)}) \qquad \text{vereinfacht}
\end{aligned}
$$

$$
\begin{aligned}
\text{R.S.} &= 2 \cdot \texttt{(len } L'\texttt{)} \\
&= 2 \cdot \texttt{(if (empty? } L'\texttt{) 0 (+ 1 (len (rest } L'\texttt{))))} \qquad \text{nach Def } \texttt{len} \\
&= 2 \cdot \texttt{(+ 1 (len (rest } L'\texttt{)))} \qquad \text{nach Def } \texttt{empty?}, \texttt{if} \\
&= 2 \cdot \texttt{(+ 1 (len } L\texttt{))} \qquad \text{nach Def } \texttt{rest}, L' \\
&= 2 \cdot (1 + \texttt{(len } L\texttt{)}) \qquad \text{nach Def } \texttt{+} \\
&= \text{L.S.}
\end{aligned}
$$

$\square$