

Übungen zur „Deskriptiven Programmierung“ Blatt 4

Aufgabe 8. Die Funktion *map* setzt Funktionen auf Listen fort.

$$\begin{aligned} \text{map} &:: (\alpha \rightarrow \beta) \rightarrow ([\alpha] \rightarrow [\beta]) \\ \text{map } f \ [] &= [] \\ \text{map } f \ (a : x) &= f \ a : \text{map } f \ x \end{aligned}$$

Definiere *map* ohne explizite Rekursion mit Hilfe von *fold*.

Aufgabe 9. Die Funktion *map* hat die beiden folgenden, charakteristischen Eigenschaften:

$$\begin{aligned} \text{map } id &= id \\ \text{map } (f \cdot g) &= \text{map } f \cdot \text{map } g \end{aligned}$$

Zeige die Gesetze mit Hilfe der universellen Eigenschaft von *fold*.

Aufgabe 10. Das folgende Gesetz (Englisch: map-fusion) läßt sich ebenfalls oft zur Programmoptimierung einsetzen. Es zeigt, wie ein vorgeschalteter Aufruf von *map* in *fold* hineingezogen werden kann:

$$\text{fold } e \ f \cdot \text{map } g = \text{fold } e \ (f \cdot g)$$

Leite das Gesetz aus der universellen Eigenschaft von *fold* ab.

Aufgabe 11. Der Aufruf $\text{fold } e \ f \ (a_0 : a_1 : a_2 : [])$ ergibt $f \ a_0 \ (f \ a_1 \ (f \ a_2 \ e)) = (f \ a_0 \cdot f \ a_1 \cdot f \ a_2) \ e$. Dieser Sachverhalt läßt sich wie folgt formalisieren:

$$\text{fold } e \ f = \text{post } e \cdot \text{fold } id \ (\cdot) \cdot \text{map } f$$

wobei *post* wie folgt definiert ist:

$$\begin{aligned} \text{post} &:: \alpha \rightarrow (\alpha \rightarrow \beta) \rightarrow \beta \\ \text{post } a \ f &= f \ a \end{aligned}$$

Beweise die obige Aussage.