# Übungsblatt 10

Abgabe der Lösungen: Do, 18.07., 12:00 im Briefkasten am blauen Hochhaus

**PRÄSENZAUFGABEN**

## Übung 1 Beweise mittels struktureller Induktion

Die von Ihnen den Übungen auf dem vorletzten Blatt definierten Funktionen sollten die folgenden plausiblen Eigenschaften erfüllen. Beweisen Sie dies jeweils durch Induktion über der Struktur der Argumentliste. Rechtfertigen Sie hierbei Ihre Schritte und geben Sie jeweils Ihre Induktionshypothese an. Falls es Ihnen nicht gelingt, eine der Eigenschaften zu beweisen, so kann das darauf hinweisen, dass Ihre Implementierung fehlerhaft ist!

*Hinweis:* Wir erinnern daran, dass $s = t$ als $s =_{\beta\delta} t$ zu lesen ist. Außerdem können Sie jederzeit zuvor bereits bewiesene Eigenschaften verwenden.

1. $\forall x\ xs.\ \mathit{length}\ (\mathit{snoc}\ xs\ x) = 1 + \mathit{length}\ xs$
2. $\forall xs.\ \mathit{length}\ (\mathit{reverse}\ xs) = \mathit{length}\ xs$

## Übung 2 Eine binäre Funktion: Listenkonkatenation

Wir betrachten die folgende Definition einer Funktion zur Listenkonkatenation:

$$\begin{aligned} &\mathit{Nil} \oplus ys = ys \\ &(\mathit{Cons}\ x\ xs) \oplus ys = \mathit{Cons}\ x\ (xs \oplus ys) \end{aligned}$$

Wir möchten die folgende Eigenschaft mittels struktureller Induktion beweisen:

$$\forall xs\ ys.\ \mathit{length}\ (xs \oplus ys) = \mathit{length}\ xs + \mathit{length}\ ys$$

1. Über welche Liste(n) sollten wir induzieren, über das erste Argument von $(\_ \oplus \_)$, über das zweite, oder über beide? Warum?
2. Beweisen Sie die oben angegebene Eigenschaft; begründen Sie Ihre Schritte und geben Sie explizit die Induktionshypothese an.
3. Beweisen Sie die folgenden Eigenschaften mittels struktureller Induktion. Begründen Sie Ihre Schritte und geben Sie jeweils Ihre Induktionshypothese an:
   (a) $\forall xs.\ xs \oplus \mathit{Nil} = xs$
   (b) $\forall x\ xs\ ys.\ \mathit{snoc}\ (xs \oplus ys)\ x = xs \oplus (\mathit{snoc}\ ys\ x)$

## Übung 3 Induktionsbeweise zum Selbststudium

Hier noch ein paar weitere Eigenschaften, die sich mittels struktureller Induktion zeigen lassen. Sie können die Beweise beispielsweise zur Vorbereitung auf die Klausur durchführen.

1. $\forall x\ xs.\ \mathit{reverse}\ (\mathit{snoc}\ xs\ x) = \mathit{Cons}\ x\ (\mathit{reverse}\ xs)$
2. $\forall xs.\ \mathit{reverse}\ (\mathit{reverse}\ xs) = xs$
3. $\forall x\ xs.\ xs \oplus [x] = \mathit{snoc}\ xs\ x$

## Hinweis

Sie dürfen in den Hausaufgaben alle Eigenschaften aus den Präsenzaufgaben (inklusive Übung 3!) direkt ohne erneuten Beweis verwenden.

### HAUSAUFGABEN

## Übung 4 Listenkonkatenation II (8 Punkte)

1. Beweisen Sie die folgende Eigenschaft von Listenfunktionen wie in Übung 2 mittels struktureller Induktion. Begründen Sie Ihre Schritte und geben Sie Ihre Induktionshypothese an:

   $$\forall xs\ ys.\ \mathit{reverse}\ (xs \oplus ys) = (\mathit{reverse}\ ys) \oplus (\mathit{reverse}\ xs)$$

2. Sei $reverse'$ gegeben durch die folgende induktive Definition:

   $$\begin{aligned}&\mathit{reverse'}\ ::\ \textbf{List}\ a \to \textbf{List}\ a \to \textbf{List}\ a\\ &\mathit{reverse'}\ \mathit{Nil}\ ys = ys\\ &\mathit{reverse'}\ (\mathit{Cons}\ x\ xs)\ ys = \mathit{reverse'}\ xs\ (\mathit{Cons}\ x\ ys)\end{aligned}$$

   Wir wollen nun zeigen, dass sich $reverse$ mittels $reverse'$ ausdrücken lässt. Genauer:

   $$\forall xs.\ \mathit{reverse}\ xs = \mathit{reverse'}\ xs\ \mathit{Nil} \tag{1}$$

   (a) Die Gleichung (1) verwendet $reverse'$ mit einem trivialen zweiten Parameter. Ein naiv ausgeführter Induktionsbeweis wird hier aus diesem Grund wahrscheinlich fehlschlagen. Finden Sie daher eine geeignete Gleichung der Form

   $$\forall xs\ ys.\ \mathit{reverse'}\ xs\ ys = ?$$

   als Hilfslemma und zeigen Sie diese durch Induktion. *Hinweis:* Die gesuchte Gleichung drückt *reverse' xs ys* mittels *reverse' xs Nil* und einer geeigneten Verknüpfung mit der Liste *ys* aus.

   (b) Zeigen Sie (1) per Induktion und verwenden Sie dabei Ihr Hilfslemma. *Hinweis:* Übung 3 (c) könnte hilfreich sein.

## Übung 5 Higher-order-Programmierung (8 Punkte)

Wir betrachten die folgenden Definitionen von *higher-order* Funktionen, also Funktionen, die als Argumente wiederum Funktionen erhalten:

$$
\begin{aligned}
&(.) \;:\; (b \to c) \to (a \to b) \to a \to c \\
&f \;.\; g = \lambda x.\, f\ (g\ x)
\end{aligned}
$$

$$
\begin{aligned}
&map : (a \to b) \to \mathbf{List}\ a \to \mathbf{List}\ b \\
&map\ f\ Nil = Nil \\
&map\ f\ (Cons\ x\ xs) = Cons\ (f\ x)\ (map\ f\ xs)
\end{aligned}
$$

1. Die Rekursionsvorschrift von *map* entspricht dem *fold*-Schema. Drücken Sie *map* mittels *foldL* aus.
2. Beweisen Sie die folgenden weiteren Eigenschaften mittels struktureller Induktion über der jeweiligen Argumentliste. Begründen Sie Ihre Schritte und geben Sie für jede Eigenschaft explizit Ihre Induktionshypothese an:
   (a) $\forall f\ g\ xs.\ (map\ f\ .\ map\ g)\ xs = map\ (f\ .\ g)\ xs$
   (b) $\forall xs\ ys\ f.\ map\ f\ (xs \oplus ys) = (map\ f\ xs) \oplus (map\ f\ ys)$

*Hinweis:* Wir nehmen an, dass die verwendeten Quantifizierungen jeweils die entsprechenden Typbeschränkungen respektieren.

## Übung 6 Binäre Bäume (6 Punkte)

Die folgenden Definitionen definieren eine Datentyp für *binäre Bäume* sowie einige Grundfunktionen für diesen Typ:

$$
\begin{aligned}
&\mathbf{data}\ \mathbf{BinTree}\ a\ \mathbf{where} \\
&\quad Leaf \;:\; () \to \mathbf{BinTree}\ a \\
&\quad Bin \;:\; \mathbf{BinTree}\ a \to a \to \mathbf{BinTree}\ a \to \mathbf{BinTree}\ a
\end{aligned}
$$

$$
\begin{aligned}
&mirror\ Leaf = Leaf \\
&mirror\ (Bin\ l\ x\ r) = Bin\ (mirror\ r)\ x\ (mirror\ l)
\end{aligned}
$$

$$
\begin{aligned}
&inorder\ Leaf = Nil \\
&inorder\ (Bin\ l\ x\ r) = inorder\ l \oplus (Cons\ x\ (inorder\ r))
\end{aligned}
$$

Beweisen Sie mittels struktureller Induktion die folgenden Eigenschaften. Begründen Sie dabei Ihre Schritte und geben Sie Ihre Induktionshypothese(n) jeweils explizit an.

*Hinweis:* Sie sollten jeweils zwei Induktionshypothesen aufstellen! Außerdem werden Sie vermutlich in vorherigen Übungen bereits bewiesene Eigenschaften benötigen.

1. $\forall t.\ mirror\ (mirror\ t) = t.$
2. $\forall t.\ inorder\ (mirror\ t) = reverse\ (inorder\ t).$