PROGRAMMIERUNG

ÜBUNG 5: INDUKTION

Eric Kunze
eric.kunze@tu-dresden.de

INHALT

Induktionsbeweise

Aufgaben 1 und 2

VOLLSTÄNDIGE INDUKTION AUF $\mathbb{N}$

Definition: natürliche Zahlen $\mathbb{N} := \{0, 1, \ldots\}$

$$\begin{array}{rll} \text{Basisfall:} & 0 \in \mathbb{N} & \\ \text{Rekursionsfall:} & x + 1 \in \mathbb{N} & \text{für} \quad x \in \mathbb{N} \end{array}$$

Beweis von Eigenschaften: Eigenschaft $=$ Prädikat P

$$\boxed{\text{zu zeigen:} \qquad \text{für alle } x \in \mathbb{N} \text{ gilt } P(x)}$$

vollständige Induktion:

- Induktionsanfang:
  zeige $P(x)$ für $x = 0$
- Induktionsvoraussetzung:
  Sei $x \in \mathbb{N}$, sodass $P(x)$ gilt. $P(x)$ gilt noch nicht für *alle* $x \in \mathbb{N}$
- Induktionsschritt:
  zeige $P(x + 1)$ unter Nutzung der Induktionsvoraussetzung

INDUKTION AUF LISTEN

Erinnerung: Rekursion über Listen `xs`

Basisfall: `xs = []`

Rekursionsfall: `xs = (y:ys)` für `ys :: [a]`

Beweis von Programmeigenschaften: Eigenschaft = Prädikat P

zu zeigen:	für alle `xs :: [a]` gilt P(`xs`)

Induktion auf Listen:

- Induktionsanfang:
  zeige P(`xs`) für `xs == []`
- Induktionsvoraussetzung:
  Sei `xs :: [a]` eine Liste für die P(`xs`) gilt.
- Induktionsschritt:
  zeige P(`x:xs`) für alle `x :: a` unter Nutzung der Induktionsvoraussetzung

Allgemeiner Hinweis: Es müssen immer alle Variablen quantifiziert werden!

STRUKTURELLE INDUKTION

Erinnerung: Rekursion über Bäume

Basisfall: `Nil` oder `Leaf x` für `x :: a`

Rekursionsfall: `Branch x l r` für `x :: a` und `l,r :: BinTree a`

zu zeigen:	für alle `t :: BinTree a` gilt P(`t`)

strukturelle Induktion:

- Induktionsanfang:
  zeige P(`t`) für `t == Nil` oder `t == Leaf x` für alle `x :: a`
- Induktionsvoraussetzung:
  Seien `l, r :: BinTree a` zwei Bäume, sodass P(`l`) und P(`r`) gilt.
- Induktionsschritt:
  zeige P(`Branch x l r`) für alle `x :: a` unter Nutzung der Induktionsvoraussetzung

Allgemeiner Hinweis: Es müssen immer alle Variablen quantifiziert werden!

FEHLERQUELLEN

- kein Induktionsprinzip
- IV wird im Induktionsschritt nicht verwendet
- fehlende Quantifizierung (nur Gleichungen bringen kaum Punkte)
- *Missachtung freier Variablen*
- zu beweisende Eigenschaft P wird für `xs` angenommen, um sie dann im Induktionsschritt nochmal für `xs` zu beweisen — eine Tautologie
- Annahme, dass P bereits für alle Listen gilt, um es dann für `x:xs` nochmal zu zeigen

AUFGABE 1

Zu zeigen ist die Gleichung

$$\texttt{sum (foo xs) = 2 * sum xs - length xs} \qquad \text{für alle } \texttt{xs :: Int}$$

mittels Induktion über Listen.

Induktionsanfang: Sei `xs == []`.

linke Seite:

$$\texttt{sum (foo [])} \overset{(2)}{=} \texttt{sum []} \overset{(6)}{=} \texttt{0}$$

rechte Seite:

$$\texttt{2 * sum [] - length []} \overset{(10)}{=} \texttt{2 * sum [] - 0} \overset{(6)}{=} \texttt{2 * 0 - 0 = 0}$$

Induktionsvoraussetzung: Sei `xs :: [Int]`, sodass

$$\texttt{sum (foo xs) = 2 * sum xs - length xs}$$

gilt.

AUFGABE 1 (FORTSETZUNG)

Induktionsschritt: Sei `x :: Int`. Es gilt

$$
\begin{aligned}
\texttt{sum (foo (x:xs))} &\overset{(3)}{=} \texttt{sum (x : x : (-1) : foo xs)} \\
&\overset{3*(7)}{=} \texttt{x + x + (-1) + sum (foo xs)} \\
&\overset{(IV)}{=} \texttt{x + x + (-1) + 2 * sum xs - length xs} \\
&\overset{(Komm.)}{=} \texttt{2 * x + 2 * sum xs - 1 - length xs)} \\
&\overset{(Dist.)}{=} \texttt{2 * (x + sum xs) - (1 + length xs)} \\
&\overset{(7)}{=} \texttt{2 * sum (x:xs) - (1 + length xs)} \\
&\overset{(11)}{=} \texttt{2 * sum (x:xs) - length (x:xs)}
\end{aligned}
$$

AUFGABE 2 — TEIL (A)

Sei $\texttt{a}$ ein beliebiger Typ. Zu zeigen ist die Gleichung

$$\texttt{[x] ++ rev ys ++ rev xs = rev (xs ++ ys ++ [x])} \quad \text{für alle } \texttt{xs, ys :: [a]} \tag{H3}$$

Der Beweis funktioniert ohne Induktion:

$$\begin{aligned}
\texttt{[x] ++ rev ys ++ rev xs} &\overset{\text{(Ass.)}}{=} \texttt{[x] ++ (rev ys ++ rev xs)} \\
&\overset{\text{(H2)}}{=} \texttt{[x] ++ rev (xs ++ ys)} \\
&\overset{\text{(H1)}}{=} \texttt{rev [x] ++ rev (xs ++ ys)} \\
&\overset{\text{(H2)}}{=} \texttt{rev ((xs ++ ys) ++ [x])} \\
&\overset{\text{(Ass.)}}{=} \texttt{rev (xs ++ ys ++ [x])}
\end{aligned}$$

AUFGABE 2 — TEIL (B)

Sei `a` ein beliebiger Typ. Zu zeigen ist die Gleichung

$$\texttt{preOrder t = rev (mPostOrder t)} \quad \text{für alle} \quad \texttt{t :: BinTree a}$$

mittels struktureller Induktion.

Induktionsanfang: Sei `x :: a` beliebig und `t = Leaf x`.

$$\text{linke Seite:} \quad \texttt{preOrder (Leaf x)} \overset{(4)}{=} \texttt{[x]}$$

$$\text{rechte Seite:} \quad \texttt{rev (mPostOrder (Leaf x))} \overset{(8)}{=} \texttt{rev [x]} \overset{(\texttt{H1})}{=} \texttt{[x]}$$

Induktionsvoraussetzung: Seien `l, r:: BinTree a`, sodass

$$\texttt{preOrder l} = \texttt{rev(mPostOrder l)} \tag{IV1}$$

$$\texttt{preOrder r} = \texttt{rev(mPostOrder r)} \tag{IV2}$$

gilt.

Induktionsschritt: Sei `x :: a` beliebig. Es gilt

$$
\begin{aligned}
\texttt{preOrder (Node x l r)} &\overset{(5)}{=} \texttt{[x] ++ preOrder l ++ preOrder r} \\
&\overset{(IV1)}{=} \texttt{[x] ++ rev(mPostOrder l) ++ preOrder r} \\
&\overset{(IV2)}{=} \texttt{[x] ++ rev(mPostOrder l) ++ rev(mPostOrder r)} \\
&\overset{(H3)}{=} \texttt{rev(mPostOrder r ++ mPostOrder l ++ [x])} \\
&\overset{(9)}{=} \texttt{rev(mPostOrder (Node x l r))}
\end{aligned}
$$

ENDE

Fragen?