

Praktische Informatik 3

Einführung in die Funktionale Programmierung

Vorlesung vom 23.01.2007: Verifikation und Beweis

Christoph Lüth

WS 06/07



Inhalt

- **Verifikation**: Wann ist ein Programm **korrekt**?
- **Beweis**: Wie **beweisen** wir Korrektheit und andere Eigenschaften?
- Techniken:
 - Vollständige Induktion
 - Strukturelle Induktion
 - Fixpunktinduktion
- **Beispiele**, und längeres **Fallbeispiel**: mergesort

Rekursive Definition, induktiver Beweis

- Definition ist **rekursiv**

- Basisfall (leere Liste)
- Rekursion ($x:xs$)

```
rev :: [a] -> [a]
```

```
rev [] = []
```

```
rev (x:xs) = rev xs ++ [x]
```

- Reduktion der Eingabe (vom größeren aufs kleinere)

- **Beweis** durch Induktion

- Schluß vom kleineren aufs größere

Beweis durch vollständige Induktion

Zu zeigen:

Für alle natürlichen Zahlen x gilt $P(x)$.

Beweis:

- Induktionsbasis: $P(0)$
- Induktionsschritt:

Induktionsvoraussetzung $P(x)$, zu zeigen $P(x + 1)$.

Beweis durch strukturelle Induktion

Zu zeigen:

Für alle (endlichen) Listen xs gilt $P(xs)$

Beweis:

- Induktionsbasis: $P([])$
- Induktionsschritt:

Induktionsvoraussetzung $P(xs)$, zu zeigen $P(x : xs)$

Ein einfaches Beispiel

Zu zeigen:

$$\forall xs\ ys. \text{len } (xs ++ ys) = \text{len } xs + \text{len } ys$$

Beweis: Induktion über xs .

- **Induktionsbasis:** $xs = []$

$$\begin{aligned}\text{len } [] + \text{len } ys &= 0 + \text{len } ys \\ &= \text{len } ys \\ &= \text{len } ([] ++ ys)\end{aligned}$$

- **Induktionsschritt:**

Voraussetzung: $\text{len } xs + \text{len } ys = \text{len } (xs ++ ys)$, dann

$$\begin{aligned}\text{len } (x : xs) + \text{len } ys &= 1 + \text{len } xs + \text{len } ys && \text{(Def. von len)} \\ &= 1 + \text{len } (xs ++ ys) && \text{(Ind.vor.)} \\ &= \text{len } (x : xs ++ ys) && \text{(Def. von len)}\end{aligned}$$

Noch ein Beispiel

Zu zeigen:

$$\forall xs\ ys. \text{rev}\ (xs ++ ys) = \text{rev}\ ys ++ \text{rev}\ xs$$

Beweis: Induktion über xs .

- **Induktionsbasis:**

$$\begin{aligned} \text{rev}\ ([\] ++ ys) &= \text{rev}\ ys \\ &= \text{rev}\ ys ++ \text{rev}\ [\] \end{aligned}$$

- **Induktionsschritt:**

Voraussetzung ist $\text{rev}\ (xs ++ ys) = \text{rev}\ ys ++ \text{rev}\ xs$, dann

$$\begin{aligned} \text{rev}\ (x : xs ++ ys) &= \text{rev}\ (xs ++ ys) ++ [x] && \text{(Def. von rev)} \\ &= (\text{rev}\ ys ++ \text{rev}\ xs) ++ [x] && \text{(Ind.vor.)} \\ &= \text{rev}\ ys ++ (\text{rev}\ xs ++ [x]) && \text{(++ assoziativ)} \\ &= \text{rev}\ ys ++ \text{rev}\ (x : xs) && \text{(Def. von rev)} \end{aligned}$$

Strukturelle Induktion über anderen Datentypen

Gegeben binäre Bäume:

```
data Tree a = Mt | T (Tree a) a (Tree a)
```

Zu zeigen:

Für alle (endlichen) Bäume t gilt $P(t)$

Beweis:

- Induktionsbasis: $P(\text{Mt})$
- Induktionsschritt:

Voraussetzung $P(s)$, $P(t)$, zu zeigen $P(\text{T } s \ a \ t)$.

Ein einfaches Beispiel

Gegeben: `map` für Bäume:

```
fmap :: (a -> b) -> Tree a -> Tree b
```

```
fmap f Mt = Mt
```

```
fmap f (T s a b) = T (fmap f s) (f a) (fmap f t)
```

Sowie Aufzählung der Knoten:

```
inorder :: Tree a -> [a]
```

```
inorder Mt = []
```

```
inorder (T s a b) = inorder s ++ [a] ++ inorder t
```

Ein einfaches Beispiel

Behauptung: `inorder (fmap f t) = map f (inorder t)`

Beweis:

- Induktionsbasis:

`inorder (fmap f Mt) = inorder Mt = [] = map f (inorder Mt)`

- Induktionsschritt:

```
inorder (fmap f (T s a t))
= inorder (T (fmap f s) (f a) (fmap f t))
= inorder (fmap f s) ++ [f a] ++ inorder (fmap f t)
= -- Nach Induktionssvoraussetzung
  map f (inorder s) ++ [f a] ++ map f (inorder t)
= -- Nach Lemma, und [f a] = map f [a]
  map f (inorder s ++ [a] ++ inorder t)
= map f (inorder (T s a t))
```

Fixpunktinduktion

Gegeben: allgemein rekursive Definition

$$fx = E \quad E \text{ enthält rekursiven Aufruf } f t$$

Zu zeigen: $\forall x. P(f x)$

Beweis: Annahme $P(f t)$, zu zeigen: $P(E)$.

- Zu zeigen: ein Rekursionsschritt erhält P
- Ein Fall für jede rekursive Gleichung.
- Induktionsverankerung: nichtrekursive Gleichungen.

Fallbeispiel: Verifikation von Mergesort