Prof. Dr. Martin Erwig

# Kurs 01852

# Fortgeschrittene Konzepte funktionaler Programmierung

**LESEPROBE**

Fakultät für
**Mathematik und Informatik**

FernUniversität in Hagen

Setzt man dieses Ergebnis in die Gleichung (3.2) ein, so erhält man

$$\texttt{power'}\ (x',n) = x' \ \texttt{*}\ x'^{n-1} = x'^{n}$$

womit der Satz bewiesen ist. □

### 3.1.2 Strukturelle Induktion auf Datentypen

Die Gültigkeit der beiden Induktionsverfahren aus dem vorigen Abschnitt ergibt sich aus der induktiven Definition der natürlichen Zahlen:

1. $0 \in \mathbb{N}$
2. $n \in \mathbb{N} \Rightarrow (suc(n) \in \mathbb{N} \wedge suc(n) \neq 0)$
3. $\mathbb{N}$ enthält keine weiteren Elemente.

Das heißt, die natürlichen Zahlen kann man auch als freie[1] Termalgebra über der Konstanten $0 : \mathbb{N}$ und der Operation $suc : \mathbb{N} \to \mathbb{N}$ auffassen. Ebenso sind ML-Datentypen nichts anderes als freie Termalgebren – zumindest gilt dies für solche Datentypen, die keine Funktionstypen enthalten. Ein ML-Datentyp gemäß der folgenden Form (alle $\tau_{ij}$ seien konstante Typen)

$$\begin{array}{l}\texttt{datatype}\ (\alpha_1,\ldots,\alpha_k)\ \textit{type}\ \texttt{=}\\ \quad \textit{con}_1\ \texttt{of}\ \tau_{11}\ \texttt{*}\ \ldots\ \texttt{*}\ \tau_{1m_1}\\ \texttt{|}\ \ldots\\ \texttt{|}\ \textit{con}_n\ \texttt{of}\ \tau_{n1}\ \texttt{*}\ \ldots\ \texttt{*}\ \tau_{nm_n}\end{array}$$

definiert eine freie Algebra von Termen des Typs $(\alpha_1,\ldots,\alpha_k)\ \textit{type}$, die über die Operationen

$$\begin{array}{l}\textit{con}_1\texttt{:}\ \tau_{11}\ \texttt{*}\ \ldots\ \texttt{*}\ \tau_{1m_1}\ \texttt{->}\ (\alpha_1,\ldots,\alpha_k)\ \textit{type}\\ \ldots\\ \textit{con}_n\texttt{:}\ \tau_{n1}\ \texttt{*}\ \ldots\ \texttt{*}\ \tau_{nm_n}\ \texttt{->}\ (\alpha_1,\ldots,\alpha_k)\ \textit{type}\end{array}$$

konstruiert werden. Für jede solche Termalgebra stellt die *strukturelle Induktion* ein eigenes Induktionsprinzip zur Verfügung. Im folgenden sei $x_{ij}$ vom Typ $\tau_{ij}$ für $1 \leq i \leq n, 1 \leq j \leq m_i$. Dann lautet das Prinzip der strukturellen Induktion in allgemeiner Form:

*Prinzip der strukturellen Induktion*

*Falls aus der Gültigkeit von $P$ für alle Argumente vom Typ $\tau$ eines Konstruktors die Gültigkeit von $P$ für den entsprechenden Konstruktorterm folgt, und dies für alle Konstruktoren des Typs $\tau$, so folgt daraus die Gültigkeit der Aussage $P$ für alle Elemente des Typs $\tau$.*

[1] Das bedeutet, daß alle Terme verschieden sind.

Als Deduktionsregel notieren wir dies:

$$\frac{\begin{array}{r}\forall i \le n : (\forall j \le m_i : (\tau_{ij} = (\alpha_1, \ldots, \alpha_k)\ \ type \Rightarrow P(x_{ij}))) \\ \Rightarrow P(con_i\ \ (x_{i1}, \ldots, x_{im_i}))\end{array}}{\forall x \in (\alpha_1, \ldots, \alpha_k)\ \ type : P(x)}$$

Für Listen ergibt sich daraus das folgende Induktionsprinzip:

*Strukturelle Induktion für Listen*

$$\frac{P(\texttt{[]}) \qquad \forall l \in \texttt{'a list} : \ \forall x \in \texttt{'a} : \ (P(l) \Rightarrow P(x::l))}{\forall l \in \texttt{'a list} : P(l)}$$

Im folgenden nehmen wir an, daß freie Variablen in Induktionsregeln immer typkorrekt allquantifiziert sind, das heißt, wir können die obige Regel auch kürzer notieren:

$$\frac{P(\texttt{[]}) \qquad (P(l) \Rightarrow P(x::l))}{P(l)}$$

(Die Quantifizierung der Schlußfolgerung ergibt sich aus der Art der Induktion, hier: Listen). Mit der Listeninduktion kann man sehr viele Eigenschaften von Listenfunktionen verifizieren.

Als Beispiel wollen wir zeigen, wie sich das Spiegeln einer Liste (mit `rev`) bezüglich der Konkatenation von Listen (`@`) verhält. Zunächst geben wir die Definitionen der beiden Listenfunktionen an.[2]

```
fun []      @ l = l
 |  (x::l') @ l = x::(l' @ l)

fun rev []     = []
 |  rev (x::l) = (rev l) @ [x]
```

Als Zwischenschritt zeigen wir zunächst, daß die Listenkonkatenation assoziativ ist:

**Satz 3.3** *Für beliebige Listen $l_1$, $l_2$ und $l_3$ gilt:*

$$(l_1\ \texttt{@}\ l_2)\ \texttt{@}\ l_3 = l_1\ \texttt{@}\ (l_2\ \texttt{@}\ l_3).$$

**Beweis.** Wir führen eine strukturelle Induktion über $l_1$ durch. (Der Linkspfeil über der Nummer einer Funktionsgleichung bezeichnet deren Anwendung von rechts nach links.)

[2] Infix-Symbole wie `@` werden auch auf der linken Seite von Funktionsdefinitionen infix notiert.

$\underline{l_1 = \texttt{[]}}$ Hier gilt:

$$\begin{aligned} \texttt{[] @ (}l_2\texttt{ @ }l_3\texttt{)} &= l_2\texttt{ @ }l_3 && \{\texttt{@}_1\} \\ &= \texttt{([] @ }l_2\texttt{) @ }l_3 && \{\texttt{@}_1^{\leftarrow}\} \end{aligned}$$

$\underline{l_1 = x\texttt{::}l}$ Hier ergibt sich nun:

$$\begin{aligned} \texttt{(}x\texttt{::}l\texttt{) @ (}l_2\texttt{ @ }l_3\texttt{)} &= x\texttt{::(}l\texttt{ @ (}l_2\texttt{ @ }l_3\texttt{))} && \{\texttt{@}_2\} \\ &= x\texttt{::((}l\texttt{ @ }l_2\texttt{) @ }l_3\texttt{)} && \{\text{Ind.Ann.}\} \\ &= \texttt{(}x\texttt{::(}l\texttt{ @ }l_2\texttt{)) @ }l_3 && \{\texttt{@}_2^{\leftarrow}\} \\ &= \texttt{((}x\texttt{::}l\texttt{) @ }l_2\texttt{) @ }l_3 && \{\texttt{@}_2^{\leftarrow}\} \end{aligned}$$

□

Nun können wir den folgenden Satz beweisen:

**Satz 3.4** *Für beliebige Listen $l_1$ und $l_2$ gilt:*

$$\texttt{rev (}l_1\texttt{ @ }l_2\texttt{) = rev }l_2\texttt{ @ rev }l_1.$$

**Beweis.** Wir führen eine strukturelle Induktion über $l_1$ durch.

$\underline{l_1 = \texttt{[]}}$ $\texttt{rev ([] @ }l_2\texttt{) = rev }l_2\texttt{ = rev }l_2\texttt{ @ [] = rev }l_2\texttt{ @ rev }l_1$. Wir haben hier die Gleichung $l\texttt{ @ [] } = l$ ausgenutzt. Deren Gültigkeit kann man leicht durch strukturelle Induktion über $l$ nachweisen.

$\underline{l_1 = x\texttt{::}l}$ Wir formen wie folgt um:

$$\begin{aligned} \texttt{rev ((}x\texttt{::}l\texttt{) @ }l_2\texttt{)} &= \texttt{rev (}x\texttt{::(}l\texttt{ @ }l_2\texttt{))} && \{\texttt{@}_2\} \\ &= \texttt{rev (}l\texttt{ @ }l_2\texttt{) @ [}x\texttt{]} && \{\texttt{rev}_2\} \\ &= \texttt{(rev }l_2\texttt{ @ rev }l\texttt{) @ [}x\texttt{]} && \{\text{Ind.Ann.}\} \\ &= \texttt{rev }l_2\texttt{ @ (rev }l\texttt{ @ [}x\texttt{])} && \{\text{Satz 3.3}\} \\ &= \texttt{rev }l_2\texttt{ @ rev (}x\texttt{::}l\texttt{)} && \{\texttt{rev}_2^{\leftarrow}\} \end{aligned}$$

□

---

**Selbsttestaufgabe 4.** Zeigen Sie, daß für alle Listen $l$ und $l'$ gilt:

$$\texttt{length (}l_1\texttt{ @ }l_2\texttt{) = length }l_1\texttt{ + length }l_2$$

---

Eine weitere Instanz des allgemeinen Schemas für die strukturelle Induktion ist die Induktion über Binärbäumen. Wir verwenden die Definition von Binärbäumen aus Abschnitt 1.6:

```
datatype 'a tree = NODE of 'a * 'a tree * 'a tree
                 | EMPTY;
```