Wohldefiniertheits- & Korrektheitsfragen

- *Ströme und Funktionen auf Strömen*
  ...fundiert und wohldefiniert?
- *Korrektheit von Programmen, Beweis von Programmeigenschaften*
  ...Rekursion vs. Induktion, Induktionsbeweise

Dazu zunächst...

- *Mathematische Grundlagen*
  ...CPOs, Fixpunkte, Fixpunktsätze

Literaturhinweise

Die folgende Darstellung orientiert sich u.a. an...

- Hanne Riis Nielson, Flemming Nielson. *Semantics with Applications – A Formal Introduction*. Wiley, 1992.
- Kapitel 11 und 14
  Paul Hudak. *The Haskell School of Expression – Learning Functional Programming through Multimedia*, Cambridge University Press, 2000.
- Kapitel 8 und 17
  Simon Thompson. *Haskell – The Craft of Functional Programming*, Addison-Wesley, 2nd edition, 1999.

Unendliche Listen, Fixpunkte und Gleichungssysteme

- Ströme
  - `onetwo = 1 : 2 : onetwo`
    $\rightsquigarrow$ `[1,2,1,2,1,2,...`
  - `onestwos = 1 : onestwos : 2`
    $\rightsquigarrow$ `[1,1,1,1,1,1,...`
- Gleichungssysteme
  - `x = E[x]`

Mehr dazu im folgenden...

Mengen und Relationen 1(2)

Sei M eine Menge und R eine Relation auf M, d.h. $R \subseteq M \times M$.

Dann heißt R...

- *reflexiv* gdw. $\forall m \in M.\ m\,R\,m$
- *transitiv* gdw. $\forall m,n,p \in M.\ m\,R\,n \ \wedge\ n\,R\,p \ \Rightarrow\ m\,R\,p$
- *antisymmetrisch* gdw. $\forall m,n \in M.\ m\,R\,n \wedge n\,R\,m \ \Rightarrow\ m = n$

Wo wir dabei sind... (wenn auch in der Folge weniger wichtig)

- *symmetrisch* gdw. $\forall m,n \in M.\ m\,R\,n \iff n\,R\,m$
- *total* gdw. $\forall m,n \in M.\ m\,R\,n \ \vee\ n\,R\,m$

Mengen und Relationen 2(2)

Eine Relation R auf M heißt

- *Quasiordnung* gdw. R ist reflexiv und transitiv
- *partielle Ordnung* gdw. R ist reflexiv, transitiv und antisymmetrisch

Der Vollständigkeit halber sei ergänzt...

- *Äquivalenzrelation* gdw. R ist reflexiv, transitiv und symmetrisch

...eine partielle Ordnung ist also eine antisymmetrische Quasiordnung, eine Äquivalenzrelation eine symmetrische Quasiordnung.

Schranken, kleinste, grösste Elemente

Sei $(Q, \sqsubseteq)$ eine Quasiordnung, sei $q \in Q$ und $Q' \subseteq Q$.

Dann heißt q...

- *obere* (*untere*) *Schranke* von Q', in Zeichen: $Q' \sqsubseteq q$ ($q \sqsubseteq Q'$), wenn für alle $q' \in Q$ gilt: $q' \sqsubseteq q$ ($q \sqsubseteq q'$)
- *kleinste obere* (*größte untere*) *Schranke* von Q', wenn q obere (untere) Schranke von Q' ist und für jede andere obere (untere) Schranke $\hat{q}$ von Q' gilt: $q \sqsubseteq \hat{q}$ ($\hat{q} \sqsubseteq q$)
- *größtes* (*kleinstes*) Element von Q, wenn gilt: $Q \sqsubseteq q$ ($q \sqsubseteq Q$)

Eindeutigkeit von Schranken

- In partiellen Ordnungen sind kleinste obere und größte untere Schranken eindeutig bestimmt, wenn sie existieren.
- Existenz (und damit Eindeutigkeit) vorausgesetzt, wird die kleinste obere (größte untere) Schranke einer Menge $P' \subseteq P$ der Grundmenge einer partiellen Ordnung $(P, \sqsubseteq)$ mit $\bigsqcup P'$ ($\bigsqcap P'$) bezeichnet. Man spricht dann auch vom *Supremum* und *Infimum* von P'.
- Analog für kleinste und größte Elemente. Existenz vorausgesetzt, werden sie üblicherweise mit $\bot$ und $\top$ bezeichnet.

Verbände und vollständige Verbände

Sei $(P, \sqsubseteq)$ eine partielle Ordnung.

Dann heißt $(P, \sqsubseteq)$...

- *Verband*, wenn jede *endliche* Teilmenge P' von P eine kleinste obere und eine kleinste untere Schranke in P besitzt
- *vollständiger Verband*, wenn *jede* Teilmenge P' von P eine kleinste obere und eine kleinste untere Schranke in P besitzt

...(vollständige) Verbände sind also spezielle partielle Ordnungen.

Vollständige partielle Ordnungen

...ein etwas schwächerer, aber in der Informatik oft ausreichender und daher angemessenerer Begriff.

Sei $(P, \sqsubseteq)$ eine partielle Ordnung.

Dann heißt $(P, \sqsubseteq)$...

- *vollständig*, kurz *CPO* (von engl. complete partial order), wenn jede aufsteigende Kette $K \subseteq P$ eine kleinste obere Schranke in P besitzt.

Es gilt:

- Eine CPO $(C, \sqsubseteq)$ (genauer wäre: "kettenvollständige partielle Ordnung (engl. chain complete partial order (CCPO)") besitzt stets ein kleinstes Element, eindeutig bestimmt als Supremum der leeren Kette und üblicherweise mit $\perp$ bezeichnet: $\perp =_{df} \bigsqcup \emptyset$.

Ketten

Sei $(P, \sqsubseteq)$ eine partielle Ordnung.

Eine Teilmenge $K \subseteq P$ heißt...

- *Kette* in P, wenn die Elemente in K total geordnet sind. Für $K = \{k_0 \sqsubseteq k_1 \sqsubseteq k_2 \sqsubseteq \ldots\}$ ($\{k_0 \sqsupseteq k_1 \sqsupseteq k_2 \sqsupseteq \ldots\}$) spricht man auch genauer von einer *aufsteigenden* (*absteigenden*) Kette in P.

Eine Kette K heißt...

- *endlich*, wenn K endlich ist, sonst *unendlich*.

Kettenendlichkeit, endliche Elemente

Eine partielle Ordnung $(P, \sqsubseteq)$ heißt

- *kettenendlich* gdw. P enthält keine unendlichen Ketten

Ein Element $p \in P$ heißt

- *endlich* gdw. die Menge $Q =_{df} \{q \in P \mid q \sqsubseteq p\}$ keine unendliche Kette enthält

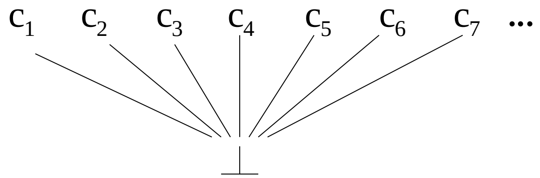
- *endlich relativ zu* $r \in P$ gdw. die Menge $Q =_{df} \{q \in P \mid r \sqsubseteq q \sqsubseteq p\}$ keine unendliche Kette enthält

(Standard-) CPO-Konstruktionen 1(4)

Flache CPOs...

Sei $(C, \sqsubseteq)$ eine CPO. Dann heißt $(C, \sqsubseteq)$...

- *flach*, wenn für alle $c, d \in C$ gilt: $c \sqsubseteq d \Leftrightarrow c = \perp \ \vee \ c = d$



(Standard-) CPO-Konstruktionen 2(4)

Produktkonstruktionen...

Seien $(P_1, \sqsubseteq_1), (P_2, \sqsubseteq_2), \ldots, (P_n, \sqsubseteq_n)$ CPOs. Dann sind auch...

- das *nichtstrikte* (*direkte*) *Produkt* $(\times P_i, \sqsubseteq)$ mit
  - $(\times P_i, \sqsubseteq) = (P_1 \times P_2 \times \ldots \times P_n, \sqsubseteq)$ mit $\forall (p_1, p_2, \ldots, p_n), (q_1, q_2, \ldots, q_n) \in \times P_i.\ (p_1, p_2, \ldots, p_n) \sqsubseteq (q_1, q_2, \ldots, q_n) \Rightarrow \forall i \in \{1, \ldots, n\}.\ p_i \sqsubseteq_i q_i$
- und das *strikte* (*direkte*) *Produkt* (*smash Produkt*) mit
  - $(\otimes P_i, \sqsubseteq) = (P_1 \otimes P_2 \otimes \ldots \otimes P_n, \sqsubseteq)$, wobei $\sqsubseteq$ wie oben definiert ist, jedoch zusätzlich gesetzt wird:

$$(p_1, p_2, \ldots, p_n) = \perp \Rightarrow \exists\, i \in \{1, \ldots, n\}.\ p_i = \perp_i$$

CPOs.

(Standard-) CPO-Konstruktionen 3(4)

Summenkonstruktion...

Seien $(P_1, \sqsubseteq_1), (P_2, \sqsubseteq_2), \ldots, (P_n, \sqsubseteq_n)$ CPOs. Dann ist auch...

- die *direkte Summe* $(\bigoplus P_i, \sqsubseteq)$ mit...
  - $(\bigoplus P_i, \sqsubseteq) = (P_1 \dot\cup P_2 \dot\cup \ldots \dot\cup P_n, \sqsubseteq)$ disjunkte Vereinigung der P_i, $i \in \{1, \ldots, n\}$ und $\forall\, p, q \in \bigoplus P_i.\ p \sqsubseteq q \Rightarrow \exists\, i \in \{1, \ldots, n\}.\ p, q \in P_i \ \wedge\ p \sqsubseteq_i q$ und der Identifikation der kleinsten Elemente der $(P_i, \sqsubseteq_i)$, $i \in \{1, \ldots, n\}$, d.h. $\perp =_{df} \perp_i$, $i \in \{1, \ldots, n\}$

eine CPO.

(Standard-) CPO-Konstruktionen 4(4)

Funktionenraum...

Seien $(C, \sqsubseteq_C)$ und $(D, \sqsubseteq_D)$ zwei CPOs und $[C \to D] =_{df} \{f : C \to D \mid f \text{ stetig}\}$ die Menge der stetigen Funktionen von C nach D.

Dann ist auch...

- der *stetige Funktionenraum* $([C \to D], \sqsubseteq)$ eine CPO mit
  - $\forall\, f, g \in [C \to D].\ f \sqsubseteq g \Longleftrightarrow \forall\, c \in C.\ f(c) \sqsubseteq_D g(c)$

Funktionen auf CPOs / Eigenschaften

Seien $(C, \sqsubseteq_C)$ und $(D, \sqsubseteq_D)$ zwei CPOs und sei $f : C \to D$ eine Funktion von C nach D.

Dann heißt f...

- *monoton* gdw. $\forall\, c, c' \in C.\ c \sqsubseteq_C c' \Rightarrow f(c) \sqsubseteq_D f(c')$
  (*Erhalt der Ordnung der Elemente*)
- *stetig* gdw. $\forall\, C' \subseteq C.\ f(\bigsqcup_C C') =_D \bigsqcup_D f(C')$
  (*Erhalt der kleinsten oberen Schranken*)

Sei $(C, \sqsubseteq)$ eine CPO und sei $f : C \to C$ eine Funktion auf C.

Dann heißt f...

- *inflationär* (*vergrößernd*) gdw. $\forall\, c \in C.\ c \sqsubseteq f(c)$

Funktionen auf CPOs / Resultate

Mit den vorigen Bezeichnungen gilt...

Lemma
f ist monoton gdw. $\forall C' \subseteq C.\ f(\bigsqcup_C C') \sqsupseteq_D \bigsqcup_D f(C')$

Korollar
Eine stetige Funktion ist stets monoton, d.h. f stetig $\Rightarrow$ f monoton.

(Kleinste und größte) Fixpunkte 1(2)

Sei $(C, \sqsubseteq)$ eine CPO, $f : C \to C$ eine Funktion auf C und sei c ein Element von C, also $c \in C$.

Dann heißt c ...

- *Fixpunkt* von f gdw. $f(c) = c$

Ein Fixpunkt c von f heißt...

- *kleinster Fixpunkt* von f gdw. $\forall d \in C.\ f(d) = d \Rightarrow c \sqsubseteq d$
- *größter Fixpunkt* von f gdw. $\forall d \in C.\ f(d) = d \Rightarrow d \sqsubseteq c$

(Kleinste und größte) Fixpunkte 2(2)

Seien $d, c_d \in C$. Dann heißt c_d ...

- *bedingter kleinster Fixpunkt* von f bezüglich d gdw. c_d ist der kleinste Fixpunkt von C mit $d \sqsubseteq c_d$, d.h. für alle anderen Fixpunkte x von f mit $d \sqsubseteq x$ gilt: $c_d \sqsubseteq x$.

Bezeichnungen:
Der kleinste bzw. größte Fixpunkt einer Funktion f wird oft mit μf bzw. νf bezeichnet.

Fixpunktsatz

Theorem (Knaster/Tarski, Kleene)
Sei $(C, \sqsubseteq)$ eine CPO und sei $f : C \to C$ eine stetige Funktion auf C.

Dann hat f einen kleinsten Fixpunkt μf und dieser Fixpunkt ergibt sich als kleinste obere Schranke der Kette (sog. *Kleene*-Kette) $\{\bot, f(\bot), f^2(\bot), \ldots\}$, d.h.

$$\mu f = \bigsqcup_{i \in \mathbb{N}_0} f^i(\bot) = \bigsqcup \{\bot, f(\bot), f^2(\bot), \ldots\}$$

Beweis des Fixpunktsatzes 1(4)

Zu zeigen: $\mu f \ldots$

1. existiert
2. ist Fixpunkt
3. ist kleinster Fixpunkt

Beweis des Fixpunktsatzes 2(4)

1. *Existenz*
   - Es gilt $f^0\ \bot = \bot$ und $\bot \sqsubseteq c$ für alle $c \in C$.
   - Durch vollständige Induktion lässt sich damit zeigen: $f^n\bot \sqsubseteq f^n c$ für alle $c \in C$.
   - Somit gilt $f^n\bot \sqsubseteq f^m\bot$ für alle n, m mit $n \le m$. Somit ist $\{f^n\bot \mid n \ge 0\}$ eine (nichtleere) Kette in C.
   - Damit folgt die Existenz von $\bigsqcup_{i \in \mathbb{N}_0} f^i(\bot)$ aus der CPO-Eigenschaft von $(C, \sqsubseteq)$.

Beweis des Fixpunktsatzes 3(4)

2. *Fixpunkteigenschaft*

$$
\begin{array}{rcl}
 & & f(\bigsqcup_{i \in \mathbb{N}_0} f^i(\bot)) \\
(f \text{ stetig}) & = & \bigsqcup_{i \in \mathbb{N}_0} f(f^n\bot) \\
 & = & \bigsqcup_{i \in \mathbb{N}_1} f^n\bot \\
(K \text{ Kette} \Rightarrow \bigsqcup K = \bot \sqcup \bigsqcup K) & = & \bigsqcup_{i \in \mathbb{N}_1} f^n\bot \ \sqcup \ \bot \\
(f^0 = \bot) & = & \bigsqcup_{i \in \mathbb{N}_0} f^n\bot \\
 & = & \bigsqcup_{i \in \mathbb{N}_0} f^i(\bot)
\end{array}
$$

Beweis des Fixpunktsatzes 4(4)

3. *Kleinster Fixpunkt*
   - Sei c beliebig gewählter Fixpunkt von f. Dann gilt $\bot \sqsubseteq c$ und somit auch $f^n\bot \sqsubseteq f^n c$ für alle $n \ge 0$.
   - Folglich gilt $f^n\bot \sqsubseteq c$ wg. der Wahl von c als Fixpunkt von f.
   - Somit gilt auch, dass c eine obere Schranke von $\{f^i(\bot) \mid i \in \mathbb{N}_0\}$ ist.
   - Da $\bigsqcup_{i \in \mathbb{N}_0} f^i(\bot)$ nach Definition die kleinste obere Schranke dieser Kette ist, gilt wie gewünscht $\bigsqcup_{i \in \mathbb{N}_0} f^i(\bot) \sqsubseteq c$.

Bedingte Fixpunkte

Theorem (Endliche Fixpunkte)
Sei $(C, \sqsubseteq)$ eine CPO, sei $f : C \to C$ eine stetige, inflationäre Funktion auf C und sei $d \in C$.

Dann hat f einen kleinsten bedingten Fixpunkt μf_d und dieser Fixpunkt ergibt sich als kleinste obere Schranke der Kette $\{d, f(d), f^2(d), \ldots\}$, d.h.

$$\mu f_d = \bigsqcup_{i \in \mathbb{N}_0} f^i(d) = \bigsqcup \{d, f(d), f^2(d), \ldots\}$$

Endliche Fixpunkte

Theorem (Endliche Fixpunkte)
Sei $(C, \sqsubseteq)$ eine CPO und sei $f : C \to C$ eine stetige Funktion auf C.

Dann gilt: Sind in der Kleene-Kette von f zwei aufeinanderfolgende Glieder gleich, etwa $f^i(\bot) = f^{i+1}(\bot)$, so gilt $\mu f = f^i(\bot)$.

Existenz endlicher Fixpunkte

Hinreichende Bedingungen für die Existenz endlicher Fixpunkte sind...

- Endlichkeit von Definitions- und Wertebereich von f
- f ist von der Form $f(c) = c \sqcup g(c)$ für monotones g über kettenendlichem Wertebereich

Kegel und Ideale

Sei $(P, \sqsubseteq)$ eine partielle Ordnung und Q eine Teilmenge von P, also $Q \subseteq P$.

Dann heißt Q ...

- *gerichtet* (*gerichtete Menge*) (engl. directed set), wenn jede *endliche* Teilmenge $R \subseteq Q$ ein Supremum in Q besitzt, d.h. $\exists\, q \in Q.\ q = \bigsqcup R$
- *Kegel*, wenn Q nach unten abgeschlossen ist, d.h. $\forall\, q \in Q\ \forall\, p \in P.\ p \sqsubseteq q \Rightarrow p \in Q$
- *Ideal*, wenn Q ein gerichteter Kegel ist, d.h. wenn Q nach unten abgeschlossen ist und jede endliche Teilmenge ein Supremum in Q besitzt.

Beachte: Für gerichtete Mengen Q gilt wg. $\emptyset \subseteq Q$ und somit $\bigsqcup \emptyset = \bot \in Q$ stets $Q \neq \emptyset$.

Idealvervollständigung

Theorem (Idealvervollständigung)
Sei $(P, \sqsubseteq)$ eine partielle Ordnung und sei I_P die Menge aller Ideale von P. Dann gilt:

- $(I_P, \subseteq)$ ist eine CPO.

Induzierte "Vervollständigung"...

- Identifikation jedes Element $p \in P$ mit dem zugehörigen Ideal $I_p =_{df} \{q \mid q \sqsubseteq p\}$ liefert eine Einbettung von P in I_P mit $p \sqsubseteq q \Leftrightarrow I_P \subseteq I_Q$

Korollar (Erweiterungskorollar für Funktionen)
Sei $(P, \sqsubseteq_P)$ eine partielle Ordnung und sei $(C, \sqsubseteq_C)$ eine CPO. Dann gilt: Alle monotonen Funktionen $f : P \to C$ lassen sich eindeutig zu stetigen Funktionen $\hat{f} : I_P \to C$ erweitern.

Schlussfolgerung

...voriges Ergebnis impliziert...

- Ströme bilden eine CPO
- Rekursive Gleichungen und Funktionen über Strömen sind wohldefiniert
- Anwendung einer Funktion auf die endlichen Präfixe eines Stroms liefert Kette von Approximationen der Anwendung der Funktion auf den Strom selbst und ist damit korrekt

Korrektheit von Programmen/Beweis von Programmeigenschaften

Induktion vs. Rekursion

- *...eine Liste ist leer oder ein Paar bestehend aus einem Element und einer weiteren Liste*
- *...ein Baum ist leer oder besteht aus einem Knoten und einer Menge von weiteren Bäumen*

Beobachtung

- Definition von Datenstrukturen
  ...häufig induktives Definitionsmuster
- Funktionen auf Datenstrukturen
  ...häufig rekursives Definitionsmuster

Induktive Beweise über (endliche) Listen

Beweismuster... Sei P eine Eigenschaft über Listen...

1. *Induktionsanfang*: ...beweise, dass P für die leere Liste gilt, d.h. beweise $P([])$.
2. *Induktionsschluss*: ...beweise unter Annahme der Gültigkeit von $P(xs)$ (*Induktionshypothese*) die Gültigkeit von $P(x : xs)$.

Allgemeiner

- ...nicht nur für Listen
  induktiver Beweis über die Struktur (*strukturelle Induktion*)

Induktion über endl. Listen / Bsp. 1(2)

Behauptung

$$\forall\, xs, ys.\ length\ (xs +\!\!+ ys) = length\ xs\ +\ length\ ys$$

Beweis über den induktiven Aufbau von xs

Induktionsanfang

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

Induktion über endl. Listen / Bsp. 2(2)

Induktionsschluss

$$\begin{aligned} & length((x:xs) +\!\!+ ys) \\ =\ & length\ (x:(xs +\!\!+ ys)) \\ =\ & 1\ +\ length\ (xs +\!\!+ ys) \\ =\ & 1\ +\ (length\ xs\ +\ length\ ys) \quad \text{(Induktionshypothese)} \\ =\ & (1\ +\ length\ xs)\ +\ length\ ys \\ =\ & length\ (x:xs)\ +\ length\ ys \end{aligned}$$

□

Gleichheit von Funktionen 1(2)

```
listSum :: Num a => [a] -> a
listSum []    = 0
listSum (x:xs) = x + listSum xs
```

Behauptung

$$\forall\, xs.\ listSum\ xs\ =\ foldr\ (+)\ 0\ xs$$

Beweis über den induktiven Aufbau von xs

Induktionsanfang

$$\begin{aligned} & listSum\ [] \\ =\ & 0 \\ =\ & foldr\ (+)\ 0\ [] \end{aligned}$$

Gleichheit von Funktionen 2(2)

Induktionsschluss

$$\begin{aligned} & listSum\ (x:xs) \\ =\ & x\ +\ listSum\ xs \\ =\ & x\ +\ foldr\ (+)\ 0\ xs \quad \text{(Induktionshypothese)} \\ =\ & foldr\ (+)\ 0\ (x:xs) \end{aligned}$$

□

Eigenschaften von map und fold 1(2)

...weitere Beispiele induktiv beweisbarer Eigenschaften:

```
map (\x -> x) = \x -> x
map (f.g) = map f . map g
map f.tail = tail . map f
map f . reverse = reverse . map f
map f . concat = concat . map (map f)
map f (xs++ys) = map f xs ++ map f ys
```

Ist `f` strikt, gilt zusätzlich

```
f . head = head . map f
```


Eigenschaften von map und fold 2(2)

...induktiv lässt sich ebenfalls zeigen:

```
foldr op e xs = foldl op e xs
```

...für assoziativen Operator `op` mit `e 'op' x = x 'op' e` für alle `x` und endliches `xs`

```
foldr op e xs = foldl (flip op) e (reverse xs)
```

...für alle endlichen `xs`

```
foldr op1 e xs = foldl op2 e xs
```

...falls

```
x 'op1' (y 'op2' z) = (x 'op1' y) 'op2' z   und
x 'op1' e = e 'op2' x
```


Eigenschaften der Listenkonkatenation

...für alle `xs`, `ys` und `zs` gilt:

```
(xs++ys) ++ zs = xs ++ (ys++zs)   (Assoziativitaet von ++)
xs++[] = []++xs    ([] neutrales Element von ++)
```


Eigenschaften von take und drop

...für alle `m`, `n` mit `m,n` $\geq$ 0 und endlichen `xs` gilt:

```
take n xs ++ drop n xs = xs
take m . take n = take (min m n)
drop m . drop n = drop (m+n)
take m . drop n = drop n . take (m+n)
```

...für `n` $\geq$ `m` gilt zusätzlich

```
drop m . take n = take (n-m) . drop m
```

Eigenschaften von `reverse`

...für alle endlichen `xs` gilt:

```
reverse (reverse xs) = xs
head (reverse xs) = last xs
last (reverse xs) = head xs
```


Endliche Listen vs. Ströme

Eigenschaften auf endlichen Listen

- können...
  z.B. `take n xs ++ drop n xs = xs`
- ...müssen sich aber nicht auf Ströme übertragen
  z.B. `reverse (reverse xs)) = xs`

...neue Beweisstrategie erforderlich.

Zur Intuition

Sukzessive Approximation von Listen

- endliche Situation ...[1,2,3,4]

```
bottom
1 : bottom
1 : 2 : bottom
1 : 2 : 3 : bottom
1 : 2 : 3 : 4 : bottom
1 : 2 : 3 : 4 : []
```

- unendliche Situation ...[1,2,3,4,..

```
bottom
1 : bottom
1 : 2 : bottom
1 : 2 : 3 : bottom
1 : 2 : 3 : 4 : bottom
1 : 2 : 3 : 4 : 5 : bottom
...
```


Sprechweisen

- `bottom` *...total undefinierte Liste*
- `1 : 2 : 3 : 4 : 5 : .. : bottom` *...partielle Liste*

Anmerkung

...jeder Haskell-Datentyp enthält den speziellen Wert $\perp$.

Polymorph	Konkret
`bot :: a`	`bot :: Integer`
`bot = bot`	

$\perp$ repräsentiert...

- fehlerhafte oder nicht terminierende Berechnungen
- kann als "kleinste" Approximation von (gewöhnlichen) Elementen des zugehörigen Datentyps aufgefasst werden

Induktive Beweise über Ströme

Beweismuster... Sei P eine Eigenschaft über Strömen

1. *Induktionsanfang*: ...beweise, dass P für die am wenigsten definierte Liste gilt, d.h. beweise $P(\perp)$ (anstelle von $P([])$).
2. *Induktionsschluss*: ...beweise unter Annahme der Gültigkeit von $P(xs)$ (*Induktionshypothese*) die Gültigkeit von $P(x:xs)$.

Induktion über Ströme / Bsp. 1(2)

Behauptung

$$\forall \text{Ströme } xs.\ take\ n\ xs\ +\!+\ drop\ n\ xs\ =\ xs$$

Beweis über den induktiven Aufbau von xs

Induktionsanfang

$$\begin{aligned} & take\ n\ \perp +\!+\ drop\ n\ \perp \\ =\ & \perp\ +\!+\ drop\ n\ \perp \\ =\ & \perp \end{aligned}$$

Induktion über Ströme / Bsp. 2(2)

Induktionsschluss

$$\begin{aligned} & take\ n\ (x:xs)\ +\!+\ drop\ n\ (x:xs) \\ =\ & x\ :\ (take\ (n-1)\ xs\ +\!+\ drop\ (n-1)\ xs \\ =\ & x\ :\ xs \qquad \text{(Induktionshypothese)} \end{aligned}$$

$\square$

Ergänzende und weiterführende Literatur

- L. C. Paulson. *Logic and Computation – Interactive Proof with Cambridge LCF*. Cambridge University Press, 1987.
- Simon Thompson. *Proof for Functional Programming*. In K. Hammond, G. Michaelson (Hrsg.), *Research Directions in Parallel Functional Programming*, Springer, 1999.

Vorschau auf die nächsten Vorlesungstermine...

- Do, 13.05.2004, 18 - 20 Uhr, FH Hörsaal 4 (Achtung: anderer Ort) (statt Di, 11.05.2004)
- Di, 18.05.01.2004, 16 - 18 Uhr, FH Hörsaal 3 (wie üblich)