
Well-definedness & Correctness Issues

- *Streams and functions on streams*
...well-defined?
- *Correctness of programs, proof of program properties*
...recursion vs. induction, proofs by induction

First...

- *Mathematical background*
...CPOs, fixed points, fixed point theorems

References

The following presentation is based on...

- Hanne Riis Nielson, Flemming Nielson. *Semantics with Applications – A Formal Introduction*. Wiley, 1992.
http://www.daimi.au.dk/~bra8130/Wiley_book/wiley.html
- Chapter 11 and 14
Paul Hudak. *The Haskell School of Expression – Learning Functional Programming through Multimedia*. Cambridge University Press, 2000.
- Chapter 8 and 17
Simon Thompson. *Haskell – The Craft of Functional Programming*. Addison-Wesley, 2nd edition, 1999.
- Chapter 10
Peter Pepper, Petra Hofstedt. *Funktionale Programmierung*. Springer-Verlag, Heidelberg, Germany, 2006. (In German)

Streams, Fixed Points, and Equation Systems

- Streams
 - `onetwo = 1 : 2 : onetwo`
 $\rightsquigarrow [1,2,1,2,1,2,\dots]$
 - `onestwos = 1 : onestwos : 2`
 $\rightsquigarrow [1,1,1,1,1,1,\dots]$
- Equation systems
 - `x = E[x]`

More on this in the following...

Sets and Relations 1(2)

Let M be a set and R a relation on M , i.e. $R \subseteq M \times M$.

Then R is called...

- *reflexive* iff $\forall m \in M. m R m$
- *transitive* iff $\forall m, n, p \in M. m R n \wedge n R p \Rightarrow m R p$
- *anti-symmetric* iff $\forall m, n \in M. m R n \wedge n R m \Rightarrow m = n$

Related further notions... (though less important for us in the following)

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

Sets and Relations 2(2)

A relation R on M is called a...

- *quasi-order* iff R is reflexive and transitive
- *partial order* iff R is reflexive, transitive, and anti-symmetric

For the sake of completeness we recall...

- *equivalence relation* iff R is reflexive, transitive, and symmetric

...i.e., a partial order is an anti-symmetric quasi-order, an equivalence relation a symmetric quasi-order.

Note: We here use terms like “partial order” as a short hand for the more accurate term “partially ordered set”.

Bounds, least and greatest Elements

Let (Q, \sqsubseteq) be a quasi-order, let $q \in Q$ and $Q' \subseteq Q$.

Then q is called...

- *upper (lower) bound* of Q' , in signs: $Q' \sqsubseteq q$ ($q \sqsubseteq Q'$), if for all $q' \in Q'$ holds: $q' \sqsubseteq q$ ($q \sqsubseteq q'$)
- *least upper (greatest lower) bound* of Q' , if q is an upper (lower) bound of Q' and for every other upper (lower) bound \hat{q} of Q' holds: $q \sqsubseteq \hat{q}$ ($\hat{q} \sqsubseteq q$)
- *greatest (least) element* of Q , if holds: $Q \sqsubseteq q$ ($q \sqsubseteq Q$)

Uniqueness of Bounds

- Given a partial order, least upper and greatest lower bounds are uniquely determined, if they exist.
- Given existence (and thus uniqueness), the least upper (greatest lower) bound of a set $P' \subseteq P$ of the basic set of a partial order (P, \sqsubseteq) is denoted by $\sqcup P'$ ($\sqcap P'$). These elements are also called *supremum* and *infimum* of P' .
- Analogously this holds for least and greatest elements. Given existence, these elements are usually denoted by \perp and \top .

Lattices and Complete Lattices

Let (P, \sqsubseteq) be a partial order.

Then (P, \sqsubseteq) is called...

- *Lattice*, if each *finite* subset P' of P contains a least upper and a greatest lower bound in P
- *complete lattice*, if *each* subset P' of P contains a least upper and a greatest lower bound in P

...(complete) lattices are special partial orders.

Complete Partial Orders

...a slightly weaker, in computer science, however, often sufficient and thus more adequate notion:

Let (P, \sqsubseteq) be a partial order.

Then (P, \sqsubseteq) is called...

- *complete*, or shorter a *CPO* (complete partial order), if each ascending chain $C \subseteq P$ has a least upper bound in P .

We have:

- A CPO (C, \sqsubseteq) (more accurate would be: "chain-complete partially ordered set (CCPO)") has always a least element. This element is uniquely determined as supremum of the empty chain and usually denoted by \perp : $\perp =_{df} \bigsqcup \emptyset$.

Chains

Let (P, \sqsubseteq) be a partial order.

A subset $C \subseteq P$ is called...

- *chain* of P , if the elements of C are totally ordered. For $C = \{c_0 \sqsubseteq c_1 \sqsubseteq c_2 \sqsubseteq \dots\}$ ($\{c_0 \supseteq c_1 \supseteq c_2 \supseteq \dots\}$) we also speak more precisely of an *ascending* (*descending*) chain of P .

A chain C is called...

- *finite*, if C is finite; *infinite* otherwise.

Finite Chains, finite Elements

A partial order (P, \sqsubseteq) is called

- *chain-finite* (German: kettenendlich) iff P is free of infinite chains

An element $p \in P$ is called

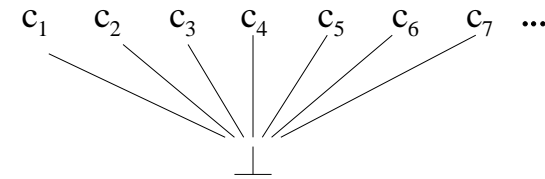
- *finite* iff the set $Q =_{df} \{q \in P \mid q \sqsubseteq p\}$ is free of infinite chains
- *finite relative to* $r \in P$ iff the set $Q =_{df} \{q \in P \mid r \sqsubseteq q \sqsubseteq p\}$ is free of infinite chains

(Standard) CPO Constructions 1(4)

Flat CPOs...

Let (C, \sqsubseteq) be a CPO. Then (C, \sqsubseteq) is called...

- *flat*, if for all $c, d \in C$ holds: $c \sqsubseteq d \Leftrightarrow c = \perp \vee c = d$



(Standard) CPO Constructions 2(4)

Product construction...

Let $(P_1, \sqsubseteq_1), (P_2, \sqsubseteq_2), \dots, (P_n, \sqsubseteq_n)$ be CPOs. Then...

- the *non-strict (direct) product* $(\times P_i, \sqsubseteq)$ with
 - $(\times P_i, \sqsubseteq) = (P_1 \times P_2 \times \dots \times P_n, \sqsubseteq)$ with $\forall (p_1, p_2, \dots, p_n), (q_1, q_2, \dots, q_n) \in \times P_i. (p_1, p_2, \dots, p_n) \sqsubseteq (q_1, q_2, \dots, q_n) \Rightarrow \forall i \in \{1, \dots, n\}. p_i \sqsubseteq_i q_i$
- and the *strict (direct) product (smash product)* with
 - $(\otimes P_i, \sqsubseteq) = (P_1 \otimes P_2 \otimes \dots \otimes P_n, \sqsubseteq)$, where \sqsubseteq is defined as above under the additional constraint:
$$(p_1, p_2, \dots, p_n) = \perp \Rightarrow \exists i \in \{1, \dots, n\}. p_i = \perp_i$$
are CPOs, too.

(Standard) CPO Constructions 3(4)

Sum construction...

Let $(P_1, \sqsubseteq_1), (P_2, \sqsubseteq_2), \dots, (P_n, \sqsubseteq_n)$ CPOs. Then...

- the *direct sum* $(\oplus P_i, \sqsubseteq)$ with...
 - $(\oplus P_i, \sqsubseteq) = (P_1 \dot{\cup} P_2 \dot{\cup} \dots \dot{\cup} P_n, \sqsubseteq)$ disjoint union of $P_i, i \in \{1, \dots, n\}$ and $\forall p, q \in \oplus P_i. p \sqsubseteq q \Rightarrow \exists i \in \{1, \dots, n\}. p, q \in P_i \wedge p \sqsubseteq_i q$ and the identification of the least elements of $(P_i, \sqsubseteq_i), i \in \{1, \dots, n\}$, i.e. $\perp =_{df} \perp_i, i \in \{1, \dots, n\}$is a CPO.

(Standard) CPO Constructions 4(4)

Function space...

Let (C, \sqsubseteq_C) and (D, \sqsubseteq_D) be two CPOs and $[C \rightarrow D] =_{df} \{f : C \rightarrow D \mid f \text{ continuous}\}$ the set of continuous functions from C to D .

Then...

- the *continuous function space* $([C \rightarrow D], \sqsubseteq)$ is a CPO where
 - $\forall f, g \in [C \rightarrow D]. f \sqsubseteq g \iff \forall c \in C. f(c) \sqsubseteq_D g(c)$

Functions on CPOs / Properties

Let (C, \sqsubseteq_C) and (D, \sqsubseteq_D) be two CPOs and let $f : C \rightarrow D$ be a function from C to D .

Then f is called...

- *monotone* iff $\forall c, c' \in C. c \sqsubseteq_C c' \Rightarrow f(c) \sqsubseteq_D f(c')$
(Preservation of the ordering of elements)
- *continuous* iff $\forall C' \sqsubseteq C. f(\sqcup_C C') =_D \sqcup_D f(C')$
(Preservation of least upper bounds)

Let (C, \sqsubseteq) be a CPO and let $f : C \rightarrow C$ be a function on C .

Then f is called...

- *inflationary (increasing)* iff $\forall c \in C. c \sqsubseteq f(c)$

Functions on CPOs / Results

Using the notations introduced before...

Lemma

f is monotone iff $\forall C' \subseteq C. f(\sqcup_C C') \sqsupseteq_D \sqcup_D f(C')$

Corollary

A continuous function is always monotone, i.e. f continuous $\Rightarrow f$ monotone.

Least and greatest Fixed Points 1(2)

Let (C, \sqsubseteq) be a CPO, $f : C \rightarrow C$ be a function on C and let c be an element of C , i.e., $c \in C$.

Then c is called...

- *fixed point* of f iff $f(c) = c$

A fixed point c of f is called...

- *least fixed point* of f iff $\forall d \in C. f(d) = d \Rightarrow c \sqsubseteq d$
- *greatest fixed point* of f iff $\forall d \in C. f(d) = d \Rightarrow d \sqsubseteq c$

Least and greatest Fixed Points 2(2)

Let $d, c_d \in C$. Then c_d is called...

- *conditional (German: bedingter) least fixed point* of f wrt d iff c_d is the least fixed point of C with $d \sqsubseteq c_d$, i.e. for all other fixed points x of f with $d \sqsubseteq x$ holds: $c_d \sqsubseteq x$.

Notations:

The least resp. greatest fixed point of a function f is usually denoted by μf resp. νf .

Fixed Point Theorem

Theorem (Knaster/Tarski, Kleene)

Let (C, \sqsubseteq) be a CPO and let $f : C \rightarrow C$ be a continuous function on C .

Then f has a least fixed point μf , which equals the least upper bound of the chain (so-called *Kleene-Chain*) $\{\perp, f(\perp), f^2(\perp), \dots\}$, i.e.

$$\mu f = \sqcup_{i \in \mathbb{N}_0} f^i(\perp) = \sqcup \{\perp, f(\perp), f^2(\perp), \dots\}$$

Proof of the Fixed Point Theorem 1(4)

We have to prove: $\mu f \dots$

1. exists
2. is a fixed point
3. is the least fixed point

Proof of the Fixed Point Theorem 2(4)

1. Existence

- It holds $f^0 \perp = \perp$ and $\perp \sqsubseteq c$ for all $c \in C$.
- By means of (complete) induction we can show: $f^n \perp \sqsubseteq f^n c$ for all $c \in C$.
- Thus we have $f^n \perp \sqsubseteq f^m \perp$ for all n, m with $n \leq m$. Hence, $\{f^n \perp \mid n \geq 0\}$ is a (non-finite) chain of C .
- The existence of $\bigsqcup_{i \in \mathbb{N}_0} f^i(\perp)$ is thus an immediate consequence of the CPO properties of (C, \sqsubseteq) .

Proof of the Fixed Point Theorem 3(4)

2. Fixed point property

$$\begin{aligned} & f(\bigsqcup_{i \in \mathbb{N}_0} f^i(\perp)) \\ (f \text{ continuous}) &= \bigsqcup_{i \in \mathbb{N}_0} f(f^i \perp) \\ &= \bigsqcup_{i \in \mathbb{N}_1} f^i \perp \\ (K \text{ chain} \Rightarrow \bigsqcup K = \perp \sqcup \bigsqcup K) &= \bigsqcup_{i \in \mathbb{N}_1} f^i \perp \sqcup \perp \\ (f^0 \perp = \perp) &= \bigsqcup_{i \in \mathbb{N}_0} f^i \perp \\ &= \bigsqcup_{i \in \mathbb{N}_0} f^i(\perp) \end{aligned}$$

Proof of the Fixed Point Theorem 4(4)

3. Least fixed point

- Let c be an arbitrarily chosen fixed point of f . Then we have $\perp \sqsubseteq c$, and hence also $f^n \perp \sqsubseteq f^n c$ for all $n \geq 0$.
- Thus, we have $f^n \perp \sqsubseteq c$ because of our choice of c as fixed point of f .
- Thus, we have, too, that c is an upper bound of $\{f^i(\perp) \mid i \in \mathbb{N}_0\}$.
- Since $\bigsqcup_{i \in \mathbb{N}_0} f^i(\perp)$ is the least upper bound of this chain by definition, we obtain as desired $\bigsqcup_{i \in \mathbb{N}_0} f^i(\perp) \sqsubseteq c$.

Conditional Fixed Points

Theorem (Conditional Fixed Points)

Let (C, \sqsubseteq) be a CPO, let $f : C \rightarrow C$ be a continuous, inflationary function on C , and let $d \in C$.

Then f has a unique conditional fixed point μf_d . This fixed point equals the least upper bound of the chain $\{d, f(d), f^2(d), \dots\}$, d.h.

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

Finite Fixed Points

Theorem (Finite Fixed Points)

Let (C, \sqsubseteq) be a CPO and let $f : C \rightarrow C$ be a continuous function on C .

Then we have: If two elements in a row occurring in the Kleene-chain of f are equal, e.g. $f^i(\perp) = f^{i+1}(\perp)$, then we have: $\mu f = f^i(\perp)$.

Existence of Finite Fixed Points

Sufficient conditions for the existence of finite fixed points e.g. are...

- Finiteness of domain and range of f
- f is of the form $f(c) = c \sqcup g(c)$ for monotone g on some chain-complete domain

Cones und Ideals

Let (P, \sqsubseteq) be a partial order and Q be a subset of P , i.e., $Q \subseteq P$.

Then Q is called...

- *directed* set (German: gerichtet (gerichtete Menge)), if each *finite* subset $R \subseteq Q$ has a supremum q , i.e. $\exists q \in Q. q = \bigsqcup R$
- *cone* (German: Kegel), if Q is downward closed, i.e. $\forall q \in Q \forall p \in P. p \sqsubseteq q \Rightarrow p \in Q$
- *ideal* (German: Ideal), if Q is a directed cone, i.e. if Q is downward closed and each finite subset has a supremum in Q .

Note: If Q is a directed set, then, we have because of $\emptyset \subseteq Q$ also $\bigsqcup \emptyset = \perp \in Q$ and thus $Q \neq \emptyset$.

Completion of Ideals

Theorem (Completion of Ideals)

Let (P, \sqsubseteq) be a partial order and let I_P be the set of all ideals of P . Then we have:

- (I_P, \subseteq) is a CPO.

Induced “completion” ...

- Identifying each element $p \in P$ with its corresponding ideal $I_p \stackrel{\text{df}}{=} \{q \mid q \sqsubseteq p\}$ yields an embedding of P into I_P with $p \sqsubseteq q \Leftrightarrow I_p \subseteq I_q$

Corollary (Extensability of Functions)

Let (P, \sqsubseteq_P) be a partial order and let (C, \sqsubseteq_C) be a CPO. Then we have: All monotone functions $f : P \rightarrow C$ can be extended to a uniquely determined continuous function $\hat{f} : I_P \rightarrow C$.

Conclusion

The previous result implies...

- Streams constitute a CPO
- Recursive equations and functions on streams are well-defined
- The application of a function to the finite prefixes of a stream yields the chain of approximations of the application of the function to the stream itself; it is thus correct

Correctness of Programs/Proof of Program Properties

Induction vs. recursion

- ...a list is either empty or a pair consisting of an element and another list
- ...a tree is either empty or consists of a node and a set of other trees

Note:

- Definition of data structures
...follow often an inductive definition pattern
- Functions on data structures
...follow often a recursive definition pattern

Inductive Proving / Proof Principles

Complete, generalized, structural induction

As a reminder: The principles of...

- complete induction

$$(A(1) \wedge (\forall n \in \mathbb{N}. A(n) \Rightarrow A(n+1))) \Rightarrow \forall n \in \mathbb{N}. A(n)$$

- generalized induction

$$(\forall n \in \mathbb{N}. (\forall m < n. A(m)) \Rightarrow A(n)) \Rightarrow \forall n \in \mathbb{N}. A(n)$$

- structural induction

$$(\forall s \in S. \forall s' \in \text{Comp}(s). A(s')) \Rightarrow A(s) \Rightarrow \forall s \in S. A(s)$$

Example: Generalized Induction

Direct computation of the Fibonacci numbers...

Let F_n , $n \in \mathbb{N}$, denote the n -th F-number, which is defined as follows:

$$F_0 = 0; F_1 = 1; \text{ for each } n \geq 2, F_n = F_{n-2} + F_{n-1}$$

Using these notations we can prove:

Theorem

$$\forall n \in \mathbb{N}. F_n = \frac{\left(\frac{1+\sqrt{5}}{2}\right)^n - \left(\frac{1-\sqrt{5}}{2}\right)^n}{\sqrt{5}}$$

Observation

Since

$$(F_i)_{i \in \mathbb{N}} = 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, \dots$$

$$(fib_i)_{i \in \mathbb{N}} = 1, 1, 2, 3, 5, 8, 13, 21, 34, \dots$$

we conclude:

Corollary $\forall n \in \mathbb{N}. fib(n) = F_{n+1}$

Proof of the Theorem 1(5)

Proof of the theorem ...by means of generalized induction.

Using the induction hypothesis that for all $k < n$ with $n \in \mathbb{N}$ some natural number the equality

$$F_k = \frac{\left(\frac{1+\sqrt{5}}{2}\right)^k - \left(\frac{1-\sqrt{5}}{2}\right)^k}{\sqrt{5}}$$

holds, we can prove the premise underlying the implication of the principle of generalized induction for all natural numbers n by investigating the following cases.

Proof of the Theorem 2(5)

Case 1: $n = 0$. In this case we obtain by a simple calculation as desired:

$$F_0 = 0 = \frac{1-1}{\sqrt{5}} = \frac{\left(\frac{1+\sqrt{5}}{2}\right)^0 - \left(\frac{1-\sqrt{5}}{2}\right)^0}{\sqrt{5}}$$

Proof of the Theorem 3(5)

Case 2: $n = 1$. Also in this case, we obtain by a straightforward calculation as desired:

$$F_1 = 1 = \frac{\sqrt{5}}{\sqrt{5}} = \frac{\left(\frac{1+\sqrt{5}}{2}\right)^1 - \left(\frac{1-\sqrt{5}}{2}\right)^1}{\sqrt{5}}$$

Proof of the Theorem 4(5)

Case 3: $n \geq 2$. Applying the induction hypothesis (IH) for $n - 2$ and $n - 1$ we obtain the desired equality:

$$\begin{aligned} \text{(Def. of } F_n) &= \frac{F_n}{F_{n-2} + F_{n-1}} \\ \text{(IH (two times))} &= \frac{\left(\frac{1+\sqrt{5}}{2}\right)^{n-2} - \left(\frac{1-\sqrt{5}}{2}\right)^{n-2}}{\sqrt{5}} + \frac{\left(\frac{1+\sqrt{5}}{2}\right)^{n-1} - \left(\frac{1-\sqrt{5}}{2}\right)^{n-1}}{\sqrt{5}} \\ &= \frac{\left[\left(\frac{1+\sqrt{5}}{2}\right)^{n-2} + \left(\frac{1+\sqrt{5}}{2}\right)^{n-1}\right] - \left[\left(\frac{1-\sqrt{5}}{2}\right)^{n-2} + \left(\frac{1-\sqrt{5}}{2}\right)^{n-1}\right]}{\sqrt{5}} \\ &= \frac{\left(\frac{1+\sqrt{5}}{2}\right)^{n-2} \left[1 + \frac{1+\sqrt{5}}{2}\right] - \left(\frac{1-\sqrt{5}}{2}\right)^{n-2} \left[1 + \frac{1-\sqrt{5}}{2}\right]}{\sqrt{5}} \\ (*) &= \frac{\left(\frac{1+\sqrt{5}}{2}\right)^{n-2} \left(\frac{1+\sqrt{5}}{2}\right)^2 - \left(\frac{1-\sqrt{5}}{2}\right)^{n-2} \left(\frac{1-\sqrt{5}}{2}\right)^2}{\sqrt{5}} \\ &= \frac{\left(\frac{1+\sqrt{5}}{2}\right)^n - \left(\frac{1-\sqrt{5}}{2}\right)^n}{\sqrt{5}} \end{aligned}$$

Proof of the Theorem 5(5)

...where the equality marked by (*) holds because of the following two sequences of equalities, whose validity can be established by means of the binomial formulae:

$$\left(\frac{1+\sqrt{5}}{2}\right)^2 = \frac{1+2\sqrt{5}+5}{4} = \frac{6+2\sqrt{5}}{4} = \frac{3+\sqrt{5}}{2} = 1 + \frac{1+\sqrt{5}}{2}$$

Similarly we can show:

$$\left(\frac{1-\sqrt{5}}{2}\right)^2 = \frac{1-2\sqrt{5}+5}{4} = \frac{6-2\sqrt{5}}{4} = \frac{3-\sqrt{5}}{2} = 1 + \frac{1-\sqrt{5}}{2}$$

□

Inductive Proofs on (finite) Lists

Proof pattern... Let P be a property on lists...

1. *Induction start:* ...prove that P holds for the empty list, i.e. prove $P([])$.
2. *Induction step:* ...prove under the assumption of the validity of $P(xs)$ (*induction hypothesis*) the validity of $P(x : xs)$.

More generally

- ...not only for lists
inductive proof along the structure (*structural induction*)

Induction on finite Lists / Example 1(2)

Proposition

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

Proof ...over the inductive structure of xs

Induction start

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

Induction on finite Lists / Example 2(2)

Induction step

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

□

Equality of Functions 1(2)

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

Proposition

$$\forall xs. \text{listSum } xs = \text{foldr } (+) 0 xs$$

Proof ...over the inductive structure of xs

Induction start

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

Equality of Functions 2(2)

Induction step

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

□

Properties of map and fold 1(2)

Some more examples of inductively provable properties...

```
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
```

Supposed f is strict, we can additionally prove:

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

Properties of map and fold 2(2)

We can also show inductively...

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

...where op is an associative operator with e 'op' $x = x$ 'op' e for all x and finite xs

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

...for all finite xs

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

...if

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

Properties of List Concatenation

...for all xs , ys and zs hold:

```
(xs++ys) ++ zs = xs ++ (ys++zs)    (Associativity of ++)
xs++[] = []++xs    ([] neutral element of ++)
```

Properties of take and drop

...for all m , n with $m, n \geq 0$ and finite xs holds:

```
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)
```

...for $n \geq m$ holds additionally

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

Properties of reverse

...for all finite xs hold:

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

Finite Lists vs. Streams

Properties of finite lists

- can...
e.g. `take n xs ++ drop n xs = xs`
- ...but need not be transferable to streams
e.g. `reverse (reverse xs) = xs`

...new proof strategies are required.

Intuition

Successively approximating lists

- finite situation ...[1,2,3,4]

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

- infinite 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
...
```

We say...

- `bottom` ...*totally undefined list*
- `1 : 2 : 3 : 4 : 5 : .. : bottom` ...*partial list*

Remark

...each Haskell data type has a special value \perp .

| | |
|------------------------|-----------------------------|
| Polymorphic | Concrete |
| <code>bot :: a</code> | <code>bot :: Integer</code> |
| <code>bot = bot</code> | |

\perp represents...

- faulty or non-terminating computations
- can be considered the “least” approximation of (ordinary) elements of the corresponding data type

Inductive Proofs over Streams

Proof pattern... Let P be a property of streams

1. *Induction start*: ...prove that P holds for the least defined list, i.e. prove $P(\perp)$ (instead of $P([])$).
2. *Induction step*: ...prove under the assumption of the validity of $P(xs)$ (*induction hypothesis*) the validity of $P(x : xs)$.

Induction over Streams / Example 1(2)

Proposition

\forall streams xs . $take\ n\ xs\ ++\ drop\ n\ xs = xs$

Proof ...over the inductive structure of xs

Induction start

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

Induction over Streams / Example 2(2)

Induction step

$$\begin{aligned} & take\ n\ (x : xs)\ ++\ drop\ n\ (x : xs) \\ &= x : (take\ (n - 1)\ xs\ ++\ drop\ (n - 1)\ xs) \\ &= x : xs \quad (\text{induction hypothesis}) \end{aligned}$$

□

Further Readings

- 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.
- Hanne and Flemming Nielson, Springer-Verlag, Heidelberg, Germany, 2007 (forthcoming).

Next lecture...

- Thu, May 31, 2007, lecture time: 4.15 p.m. to 5.45 p.m., lecture room on the ground floor of the building Argentinierstr. 8

Fifth assignment (as well as previous assignments)...

- Please check out the homepage of the course for details.