Introduction to Functional Programming

Carlos Encarnación
encarnacion.carlos@gmail.com

November 21, 2014

Lecture 5: Mathematical induction, part 1

Proof by induction

$$\mathbf{data}\ Nat = Zero \mid Succ\ Nat\ \mathbf{deriving}\ (Eq, Ord)$$

Natural numbers are either of the form $Zero$ or $Succ\ n$ for some natural number n. Hence to prove that a property P is true for all natural numbers n, we prove that

1. $P(Zero)$ holds;
2. For all natural numbers n, $P(Succ\ n)$ is true assuming that $P(n)$ does.

Definition of *expN*

$$
\begin{aligned}
\mathit{expN}\ x\ \mathit{Zero} &= 1 \\
\mathit{expN}\ x\ (\mathit{Succ}\ n) &= x * (\mathit{expN}\ x\ n)
\end{aligned}
$$

Law of exponents

$$expN\ x\ (m + n) = expN\ x\ m * expN\ x\ n$$

Proof of the law of exponents, case *Zero*

$$
\begin{aligned}
& expN\ x\ (Zero + n) \\
= \quad & \{\ Zero + n = n\ \} \\
& expN\ x\ n \\
= \quad & \{\ 1 * y = y\ \} \\
& 1 * (expN\ x\ n) \\
= \quad & \{\ expN.1\ \} \\
& (expN\ x\ Zero) * (expN\ x\ n)
\end{aligned}
$$

Proof of the law of exponents, case *Succ n*

$$
\begin{aligned}
& expN\ x\ ((Succ\ m) + n) \\
= & \quad \{\ (Succ\ m) + n = Succ\ (m + n)\ \} \\
& expN\ x\ Succ\ (m + n) \\
= & \quad \{\ expN.2\ \} \\
& x * (expN\ x\ (m + n)) \\
= & \quad \{\ \text{induction hypothesis}\ \} \\
& x * ((expN\ x\ m) * (expN\ x\ n)) \\
= & \quad \{\ \text{associativity of } *\ \} \\
& (x * (expN\ x\ m)) * (expN\ x\ n) \\
= & \quad \{\ succ.2\ \} \\
& (expN\ x\ (Succ\ m)) * (expN\ x\ n)
\end{aligned}
$$

Definition of $[\,]\ a$

$$\textbf{data}\ [\,]\ a = [\,] \mid a : ([\,]\ a)$$

Induction over finite lists

Finite lists are either of the form $[]$ or $(x : xs)$ for some finite list xs. Hence to prove by induction that a property P is true for all finite lists xs, we prove that

1. $P([])$ holds;
2. For all finite list xs, $P(x : xs)$ is true assuming that $P(xs)$ does.

Concatenation

Recall the definition of $+\!\!+$.

$$
\begin{aligned}
&(+\!\!+) && :: [a] \to [a] \to [a] \\
&(+\!\!+)\ [\,]\ && ys = ys \\
&(+\!\!+)\ (x : xs)\ && ys = x : (xs +\!\!+ ys)
\end{aligned}
$$

Concatenation is associative

For all finite lists xs, ys, zs

$$xs ++ (ys ++ zs) = (xs ++ ys) ++ zs$$

Proof that $+\!\!+$ is associative, case [ ]

$$
\begin{array}{ll}
 & [\,] +\!\!+ (ys +\!\!+ zs) \\
= & \quad \{ +\!\!+.1 \} \\
 & ys +\!\!+ zs \\
= & \quad \{ +\!\!+.1 \} \\
 & ([\,] +\!\!+ ys) +\!\!+ zs
\end{array}
$$

Proof that $+\!\!+$ is associative, case $(x : xs)$

$$
\begin{array}{ll}
 & (x : xs) +\!\!+ (ys +\!\!+ zs) \\
= & \quad \{\ +\!\!+.2\ \} \\
 & x : (xs +\!\!+ (ys +\!\!+ zs)) \\
= & \quad \{\ \text{induction hypothesis}\ \} \\
 & x : ((xs +\!\!+ ys) +\!\!+ zs) \\
= & \quad \{\ +\!\!+.2\ \} \\
 & (x : (xs +\!\!+ ys)) +\!\!+ zs \\
= & \quad \{\ +\!\!+.2\ \} \\
 & ((x : xs) +\!\!+ ys) +\!\!+ zs
\end{array}
$$

Definition of *reverse*

$$
\begin{aligned}
&\mathit{reverse} &&:: [a] \to [a] \\
&\mathit{reverse}\ [\,] &&= [\,] \\
&\mathit{reverse}\ (x : xs) &&= \mathit{reverse}\ xs +\!\!+ [x]
\end{aligned}
$$

reverse is an involution

For all finite lists xs

$$reverse\ (reverse\ xs) = xs$$

Before we prove the above equation, we will prove that

$$reverse\ (ys \mathbin{+\!\!+} [x]) = x : reverse\ ys$$

Subsidiary function, case [ ]

$$
\begin{array}{ll}
 & \mathit{reverse}\ ([\,] + [x]) \\
= & \quad \{\ +.1\ \} \\
 & \mathit{reverse}\ [x] \\
= & \quad \{\ \mathit{reverse}.2\ \} \\
 & \mathit{reverse}\ [\,] + [x] \\
= & \quad \{\ \mathit{reverse}.1 \text{ and } +.1\ \} \\
 & [x] \\
= & \quad \{\ \mathit{reverse}.1\ \} \\
 & x : \mathit{reverse}\ [\,]
\end{array}
$$

Subsidiary function, case $(y : ys)$

$$
\begin{array}{ll}
 & \mathit{reverse}\ ((y : ys) +\!\!+ [x]) \\
= & \quad \{ +\!\!+.2 \} \\
 & \mathit{reverse}\ (y : (ys +\!\!+ [x])) \\
= & \quad \{ \mathit{reverse}.2 \} \\
 & \mathit{reverse}\ (ys +\!\!+ [x]) +\!\!+ [y] \\
= & \quad \{ \text{induction hypothesis} \} \\
 & x : \mathit{reverse}\ ys +\!\!+ [y] \\
= & \quad \{ +\!\!+.2 \} \\
 & x : (\mathit{reverse}\ ys +\!\!+ [y]) \\
= & \quad \{ \mathit{reverse}.2 \} \\
 & x : (\mathit{reverse}\ (y : ys))
\end{array}
$$

Proof that *reverse* is an involution, case [ ]

$$
\begin{array}{ll}
 & \mathit{reverse}\ (\mathit{reverse}\ [\,]) \\
= & \quad \{\ \mathit{reverse}.1\ \} \\
 & \mathit{reverse}\ [\,] \\
= & \quad \{\ \mathit{reverse}.1\ \} \\
 & [\,]
\end{array}
$$

Proof that *reverse* is an involution, case []

$$
\begin{aligned}
& \mathit{reverse}\ (\mathit{reverse}\ (x : xs)) \\
= & \quad \{\ \mathit{reverse}.2\ \} \\
& \mathit{reverse}\ (\mathit{reverse}\ xs +\!\!+ [x]) \\
= & \quad \{\ \mathit{reverse}\ (ys +\!\!+ [x]) = x : \mathit{reverse}\ ys\ \} \\
& x : \mathit{reverse}\ (\mathit{reverse}\ xs) \\
= & \quad \{\ \text{induction hypothesis}\ \} \\
& x : xs
\end{aligned}
$$

Induction over partial lists

Partial lists are either of the form $\perp$ or $(x : xs)$ for some partial list xs. Thus to prove by induction that a property P is true for all partial lists we must prove that

1. $P(\perp)$ holds;
2. For all partial lists xs, $P(x : xs)$ holds assuming that $P(xs)$ does.

Example

Let xs be a partial list. Then

$$xs +\!\!+ ys = xs$$

Proof: case $\perp$

$$
\begin{aligned}
& \perp \mathbin{+\!\!+} ys \\
= & \quad \{ \mathbin{+\!\!+}.0 \} \\
& \perp
\end{aligned}
$$

Proof: case $x : xs$

$$
\begin{aligned}
& (x : xs) +\!\!+\, ys \\
= & \quad \{\ +\!\!+.2\ \} \\
& x : (xs +\!\!+\, ys) \\
= & \quad \{\ \text{induction hypothesis}\ \} \\
& x : xs
\end{aligned}
$$

Induction over infinite lists

Infinite lists are build from (:) alone.

A property P is *chain complete* if whenever xs_0, xs_1, ... is a sequence of partial lists with limit xs, and $P(xs_n)$ holds for all n, then $P\ (xs)$ also holds.

The function *length*

$$
\begin{aligned}
&\mathit{length} && :: [a] \to a \\
&\mathit{length}\ [\,] && = 0 \\
&\mathit{length}\ (\_ : \mathit{xs}) && = 1 + \mathit{length}\ \mathit{xs}
\end{aligned}
$$

The function *sum*

$$
\begin{aligned}
&sum &&:: (Num\ a) \Rightarrow [a] \rightarrow a \\
&sum\ [\,] &&= 0 \\
&sum\ (x : xs) &&= x + sum\ xs
\end{aligned}
$$

The function *concat*

$$
\begin{aligned}
&\mathit{concat} &&:: [[a]] \to [a] \\
&\mathit{concat}\ [\,] &&= [\,] \\
&\mathit{concat}\ (\mathit{xs} : \mathit{xss}) &&= \mathit{xs} +\!\!+\ \mathit{concat}\ \mathit{xs}
\end{aligned}
$$

The function *filter*

$$
\begin{array}{lll}
\mathit{filter} & & :: (a \to \mathit{Bool}) \to [a] \to [a] \\
\mathit{filter}\ p\ [\,] & & = [\,] \\
\mathit{filter}\ p\ (x : xs) & \mid p\ x & = x : \mathit{filter}\ p\ xs \\
& \mid \mathit{otherwise} & = \mathit{filter}\ p\ xs
\end{array}
$$

The function *map*

$$
\begin{aligned}
&map && :: (a \to b) \to [a] \to [b] \\
&map\ f\ [\,] && = [\,] \\
&map\ f\ (x : xs) && = f\ x : map\ xs
\end{aligned}
$$

Distributivity over concatenation

$$sum\ (xs +\!\!+\ ys) = sum\ xs + sum\ ys$$

$$concat\ (xss +\!\!+\ yss) = concat\ xss +\!\!+\ concat\ yss$$

$$filter\ p\ (xs +\!\!+\ ys) = filter\ p\ xs +\!\!+\ filter\ p\ ys$$

$$map\ f\ (xs +\!\!+\ ys) = map\ f\ xs +\!\!+\ map\ f\ ys$$

Definition

$$
\begin{aligned}
&\mathit{foldr} && :: (a \to b \to b) \to b \to [a] \to b \\
&\mathit{foldr}\ (\odot)\ e\ [\,] && = e \\
&\mathit{foldr}\ (\odot)\ e\ (x : xs) && = x \odot \mathit{foldr}\ (\odot)\ e\ xs
\end{aligned}
$$

Examples

$$\mathit{length} = \mathit{foldr}\ ((+) \circ \mathit{const}\ 1)\ 0$$

$$\mathit{sum} = \mathit{foldr}\ (+)\ 0$$

$$\mathit{concat} = \mathit{foldr}\ (+\!\!+)\ [\,]$$

$$\mathit{filter}\ p = \mathit{foldr}\ (\lambda x\ \mathit{xs} \to \mathbf{if}\ p\ x\ \mathbf{then}\ x : \mathit{xs}\ \mathbf{else}\ \mathit{xs})\ [\,]$$

$$\mathit{map}\ f = \mathit{foldr}\ ((:) \circ f)\ [\,]$$

Distributivity over concatenation

$$foldr\ (\odot)\ e\ (xs +\!\!+\ ys) = foldr\ (\odot)\ e\ xs \odot foldr\ (\odot)\ e\ ys$$

Proof: distributivity over concatenation, case [ ]

$$
\begin{array}{ll}
 & \mathit{foldr}\ (\odot)\ e\ ([\,] \mathbin{+\!\!+} \mathit{ys}) \\
= & \quad \{\ \mathbin{+\!\!+}.1\ \} \\
 & \mathit{foldr}\ (\odot)\ e\ \mathit{ys} \\
\Longleftarrow & \quad \{\ \mathit{foldr}.1 \text{ and } e \odot x = x\ \} \\
 & \mathit{foldr}\ (\odot)\ e\ [\,] \odot \mathit{foldr}\ (\odot)\ e\ \mathit{ys}
\end{array}
$$

Proof: distributivity over concatenation, case $(x : xs)$

$$
\begin{array}{ll}
 & foldr\ (\odot)\ e\ ((x : xs) +\!\!+\ ys) \\
= & \quad \{\ +\!\!+.2\ \} \\
 & foldr\ (\odot)\ e\ (x : (xs +\!\!+\ ys)) \\
= & \quad \{\ foldr.2\ \} \\
 & x \odot foldr\ (\odot)\ e\ (xs +\!\!+\ ys) \\
= & \quad \{\ \text{induction}\ \} \\
 & x \odot (foldr\ (\odot)\ e\ xs \odot foldr\ (\odot)\ e\ ys) \\
\Longleftarrow & \quad \{\ \text{associativity of } \odot \text{ and } foldr.2\ \} \\
 & foldr\ (\odot)\ e\ (x : xs) \odot foldr\ (\odot)\ e\ ys
\end{array}
$$

Property

Given functions f and g and a value a, find a function h and a value b such that

$$f \circ foldr\ g\ a = foldr\ h\ b$$

Examples

$$
\begin{aligned}
\mathit{double} \circ \mathit{sum} &= \mathit{foldr}\ ((+) \circ \mathit{double})\ 0 \\
\mathit{length} \circ \mathit{concat} &= \mathit{foldr}\ ((+\!\!+) \circ \mathit{length})\ [\,]
\end{aligned}
$$

Proof: fusion law, case $\perp$

$$
\begin{array}{ll}
 & f\ (foldr\ g\ a\ \perp) \\
\Longleftarrow & \quad \{\ foldr.0 \text{ and } f\ \perp = \perp\ \} \\
 & foldr\ h\ b\ \perp
\end{array}
$$

Proof: fusion law, case [ ]

$$
\begin{array}{ll}
 & f\ (foldr\ g\ a\ [\,])\\
\Longleftarrow & \quad \{\ foldr.1 \text{ and } b = f\ a\ \}\\
 & foldr\ h\ b\ [\,]
\end{array}
$$

Proof: fusion law, case $(x : xs)$

$$
\begin{array}{ll}
 & f \circ foldr\ g\ a\ (x : xs) \\
= & \quad \{\ foldr.2 \text{ and } \circ\ \} \\
 & f\ (g\ x\ (foldr\ g\ a\ xs)) \\
\Longleftarrow & \quad \{\ h\ x\ (f\ y) = f\ (g\ x\ y)\ \} \\
 & h\ x\ (f\ (foldr\ g\ a\ xs)) \\
= & \quad \{\ \text{induction}\ \} \\
 & h\ x\ (foldr\ h\ b\ xs) \\
= & \quad \{\ foldr.2\ \} \\
 & foldr\ h\ b\ (x : xs)
\end{array}
$$

Putting it all together

Given a strict function f we have that

$$f \circ foldr\ g\ a = foldr\ h\ b$$

whenever

$$f\ a = b \wedge f\ (g\ x\ y) = h\ x\ (f\ y)$$

Example

Consider the following equation

$$foldr\ f\ a \circ map\ g = foldr\ h\ b$$

Now, derive h

First, recall that

$$map\ g = foldr\ ((:) \circ g)\ []$$

thus

$$foldr\ f\ a\ \perp = \perp$$

and

$$foldr\ f\ a\ [] = []$$

Example

$$
\begin{array}{ll}
 & foldr\ f\ \ a\ (map\ g\ (x : xs)) \\
= & \quad \{\ map \text{ in terms of } foldr\ \} \\
 & foldr\ f\ \ a\ (foldr\ ((:) \circ g)\ [\,]\ (x : xs)) \\
= & \quad \{\ foldr.2\ \} \\
 & foldr\ f\ \ a\ (g\ x : foldr\ ((:) \circ g)\ [\,]\ xs) \\
= & \quad \{\ map \text{ in terms of } foldr\ \} \\
 & foldr\ f\ \ a\ (g\ x : map\ g\ xs) \\
= & \quad \{\ foldr.2\ \} \\
 & f\ (g\ x)\ (foldr\ f\ \ a\ (map\ g\ xs))
\end{array}
$$

hence

$$foldr\ f\ \ a\ (map\ g\ (x : xs)) = f\ (g\ x)\ (foldr\ f\ \ a\ (map\ g\ xs))$$

which implies that

$$foldr\ f\ \ a \circ map\ g = foldr\ (f \circ g)\ [\,]$$

Definition

$$
\begin{aligned}
&\mathit{foldr1} && :: (a \to a \to a) \to [a] \to a \\
&\mathit{foldr1}\ (\odot)\ [x] && = x \\
&\mathit{foldr1}\ (\odot)\ (x : xs) && = x \odot \mathit{foldr1}\ (\odot)\ xs
\end{aligned}
$$

The function *minimum*

$$minimum :: [a] \to a$$
$$minimum = foldr1\ min$$

The function *maximum*

$$\mathit{maximum} :: [a] \to a$$
$$\mathit{maximum} = \mathit{foldr1}\ \mathit{max}$$

Definition

$$
\begin{aligned}
\mathit{foldl} &:: (b \to a \to a) \to b \to [a] \to b \\
\mathit{foldl}\ (\odot)\ e\ [\,] &= e \\
\mathit{foldl}\ (\odot)\ e\ (x : xs) &= \mathit{foldl}\ (\odot)\ (e \odot x)\ xs
\end{aligned}
$$

Yet another way to define *reverse*

$$reverse = foldl\ (flip\ (:))\ [\,]$$

Duality

$$\mathit{foldl}\ (\odot)\ e = \mathit{foldr}\ (\mathit{flip}\ (\odot))\ e \circ \mathit{reverse}$$