

Laws and Equations and Coq

Dave Laing

March 26, 2013

Outline

1 Laws

Outline

1 Laws

2 Induction

Outline

1 Laws

2 Induction

3 Equational Reasoning

Outline

1 Laws

2 Induction

3 Equational Reasoning

4 Coq

Laws

Laws

- Help when reasoning about code
- Particularly code that makes use of typeclasses
- There's only so much that a typeclass can enforce
- For everything else there is
 - QuickCheck and pray to the RNG gods
 - Equational reasoning
 - Theorem provers
 - Angry letters to the authors

Monoids are great!

- Taking the tree out of tree shaped computations
 - Summarize with monoids and `Data.Foldable`
 - Fun times with finger trees

Monoids

A monoid is a combination of

- a type A

and a function

- $\oplus :: A \rightarrow A \rightarrow A$

that satisfies certain laws.

Monoid Laws

There must exist an element $e :: A$ such that for all $a :: A$

$$\blacksquare e \oplus a = a \quad (\text{Left identity})$$

$$\blacksquare a \oplus e = a \quad (\text{Right identity})$$

For all $a, b, c :: A$

$$\blacksquare a \oplus (b \oplus c) = (a \oplus b) \oplus c \quad (\text{Associativity})$$

Monoid typeclass and List instance

```
class Monoid a where  
  mempty :: a  
  mappend :: a -> a -> a
```

```
instance Monoid [a] where  
  mempty = []  
  mappend = (++)
```

(Remembering that `(++)` has type `[a] -> [a] -> [a]`)

Functors are great!

- Apply a function to everything in a structure
 - You can double the values in a tree
- Preserve the shape of the structure
- Compose functions and apply once
 - You can double and then add one to the values in a tree
- Saves multiple traversals

Aside: Type constructors

Suppose F is a type constructor (like `Maybe` or `List`)

- F isn't a type
- $F\ \text{Int}$, $F\ \text{String}$ are types
- we'll use $F\ a$, $F\ b$ as types for abstraction

Functors

A functor is a combination of

- a type constructor F

and

- a function $\text{fmap} :: (a \rightarrow b) \rightarrow F\ a \rightarrow F\ b$

that satisfies certain laws.

Functor Laws

- $\text{fmap id} = \text{id}$ (Identity)
- $\text{fmap } g (\text{fmap } f \text{ xs}) = \text{fmap } (g \circ f) \text{ xs}$ (Composition)

Functor typeclass and List instance

```
class Functor f where  
  fmap :: (a -> b) -> f a -> f b
```

```
instance Functor [] where  
  fmap = map
```

(Remember that map has type $(a \rightarrow b) \rightarrow [a] \rightarrow [b]$)

Induction

Natural numbers

- $0 :: \text{Nat}$
 - this is zero
- $S :: \text{Nat} \rightarrow \text{Nat}$
 - this is the successor function
 - it returns one more than its input
- $\text{Nat} = 0 \mid S(x)$
 - $x \in \text{Nat}$
 - so the definition of Nat is recursive
- Ascii time!

$$\begin{array}{ccccccc}
 & S & & S & & S & & \dots \\
 & / \backslash & & / \backslash & & / \backslash & & \\
 0 & & S(0) & & S(S(0)) & & S(S(S(0))) & \dots
 \end{array}$$

Addition

$(+) :: \text{Nat} \rightarrow \text{Nat} \rightarrow \text{Nat}$

$0 + n = n$

$S(m) + n = S(m + n)$

$S(S(0)) + S(S(S(0)))$

$2 + 3$

Addition

$(+) :: \text{Nat} \rightarrow \text{Nat} \rightarrow \text{Nat}$

$0 + n = n$

$S(m) + n = S(m + n)$

$S(S(0)) + S(S(S(0)))$

$2 + 3$

Addition

$(+) :: \text{Nat} \rightarrow \text{Nat} \rightarrow \text{Nat}$

$0 + n = n$

$S(m) + n = S(m + n)$

$S(S(0)) + S(S(S(0)))$

$2 + 3$

Addition

$(+) :: \text{Nat} \rightarrow \text{Nat} \rightarrow \text{Nat}$

$0 + n = n$

$S(m) + n = S(m + n)$

$S(S(0)) + S(S(S(0)))$

$2 + 3$

Addition

$(+) :: \text{Nat} \rightarrow \text{Nat} \rightarrow \text{Nat}$

$0 + n = n$

$S(m) + n = S(m + n)$

$S(S(0) + S(S(S(0))))$

$1 + (1 + 3)$

Addition

$(+) :: \text{Nat} \rightarrow \text{Nat} \rightarrow \text{Nat}$

$0 + n = n$

$S(m) + n = S(m + n)$

$S(S(0) + S(S(S(0))))$

$1 + (1 + 3)$

Addition

$(+) :: \text{Nat} \rightarrow \text{Nat} \rightarrow \text{Nat}$

$0 + n = n$

$S(m) + n = S(m + n)$

$S(S(0) + S(S(S(0))))$

$1 + (1 + 3)$

Addition

$(+) :: \text{Nat} \rightarrow \text{Nat} \rightarrow \text{Nat}$

$0 + n = n$

$S(m) + n = S(m + n)$

$S(S(0) + S(S(S(0))))$

$1 + (1 + 3)$

Addition

$(+) :: \text{Nat} \rightarrow \text{Nat} \rightarrow \text{Nat}$

$0 + n = n$

$S(m) + n = S(m + n)$

$S(S(0 + S(S(S(0))))))$

$1 + 1 + (0 + 3)$

Addition

$(+) :: \text{Nat} \rightarrow \text{Nat} \rightarrow \text{Nat}$

$0 + n = n$

$S(m) + n = S(m + n)$

$S(S(0 + S(S(S(0))))))$

$1 + 1 + (0 + 3)$

Addition

$(+) :: \text{Nat} \rightarrow \text{Nat} \rightarrow \text{Nat}$

$0 + n = n$

$S(m) + n = S(m + n)$

$S(S(0 + S(S(S(0))))))$

$1 + 1 + (0 + 3)$

Addition

$(+) :: \text{Nat} \rightarrow \text{Nat} \rightarrow \text{Nat}$

$0 + n = n$

$S(m) + n = S(m + n)$

$S(S(S(S(0))))$

$1 + 1 + 3$

Addition

$(+) :: \text{Nat} \rightarrow \text{Nat} \rightarrow \text{Nat}$

$0 + n = n$

$S(m) + n = S(m + n)$

$S(S(S(S(0))))$

5

Right identity - the long way

Goal: $\forall x :: \text{Nat}$

$$x + 0 = x$$

$(+) :: \text{Nat} \rightarrow \text{Nat} \rightarrow \text{Nat}$

$$0 + n = n$$

$$S(m) + n = S(m + n)$$

$$0 + 0 = ?$$

Right identity - the long way

Goal: $\forall x :: \text{Nat}$

$$x + 0 = x$$

$(+) :: \text{Nat} \rightarrow \text{Nat} \rightarrow \text{Nat}$

$$0 + n = n$$

$$S(m) + n = S(m + n)$$

$$0 + 0 = ?$$

Right identity - the long way

Goal: $\forall x :: \text{Nat}$

$$x + 0 = x$$

$(+) :: \text{Nat} \rightarrow \text{Nat} \rightarrow \text{Nat}$

$$0 + n = n$$

$$S(m) + n = S(m + n)$$

$$0 + 0 = ?$$

Right identity - the long way

Goal: $\forall x :: \text{Nat}$

$$x + 0 = x$$

$(+) :: \text{Nat} \rightarrow \text{Nat} \rightarrow \text{Nat}$

$$0 + n = n$$

$$S(m) + n = S(m + n)$$

$$0 + 0 = ?$$

Right identity - the long way

Goal: $\forall x :: \text{Nat}$

$$x + 0 = x$$

$(+) :: \text{Nat} \rightarrow \text{Nat} \rightarrow \text{Nat}$

$$0 + n = n$$

$$S(m) + n = S(m + n)$$

$$0 + 0 = 0$$

Right identity - the long way

Goal: $\forall x :: \text{Nat}$

$$x + 0 = x$$

$(+) :: \text{Nat} \rightarrow \text{Nat} \rightarrow \text{Nat}$

$$0 + n = n$$

$$S(m) + n = S(m + n)$$

$$0 + 0 = 0$$

$$S(0) + 0 = ?$$

Right identity - the long way

Goal: $\forall x :: \text{Nat}$

$$x + 0 = x$$

$(+) :: \text{Nat} \rightarrow \text{Nat} \rightarrow \text{Nat}$

$$0 + n = n$$

$$S(m) + n = S(m + n)$$

$$0 + 0 = 0$$

$$S(0) + 0 = ?$$

Right identity - the long way

Goal: $\forall x :: \text{Nat}$

$$x + 0 = x$$

$(+) :: \text{Nat} \rightarrow \text{Nat} \rightarrow \text{Nat}$

$$0 + n = n$$

$$S(m) + n = S(m + n)$$

$$0 + 0 = 0$$

$$S(0) + 0 = ?$$

Right identity - the long way

Goal: $\forall x :: \text{Nat}$

$$x + 0 = x$$

$(+) :: \text{Nat} \rightarrow \text{Nat} \rightarrow \text{Nat}$

$$0 + n = n$$

$$S(m) + n = S(m + n)$$

$$0 + 0 = 0$$

$$S(0) + 0 = ?$$

Right identity - the long way

Goal: $\forall x :: \text{Nat}$

$$x + 0 = x$$

$(+) :: \text{Nat} \rightarrow \text{Nat} \rightarrow \text{Nat}$

$$0 + n = n$$

$$S(m) + n = S(m + n)$$

$$0 + 0 = 0$$

$$S(0) + 0 = S(0 + 0)$$

Right identity - the long way

Goal: $\forall x :: \text{Nat}$

$$x + 0 = x$$

$(+) :: \text{Nat} \rightarrow \text{Nat} \rightarrow \text{Nat}$

$$0 + n = n$$

$$S(m) + n = S(m + n)$$

$$0 + 0 = 0$$

$$S(0) + 0 = S(0 + 0)$$

Right identity - the long way

Goal: $\forall x :: \text{Nat}$

$$x + 0 = x$$

$(+) :: \text{Nat} \rightarrow \text{Nat} \rightarrow \text{Nat}$

$$0 + n = n$$

$$S(m) + n = S(m + n)$$

$$0 + 0 = 0$$

$$S(0) + 0 = S(0 + 0)$$

Right identity - the long way

Goal: $\forall x :: \text{Nat}$

$$x + 0 = x$$

$(+) :: \text{Nat} \rightarrow \text{Nat} \rightarrow \text{Nat}$

$$0 + n = n$$

$$S(m) + n = S(m + n)$$

$$0 + 0 = 0$$

$$S(0) + 0 = S(0 + 0)$$

Right identity - the long way

Goal: $\forall x :: \text{Nat}$

$$x + 0 = x$$

$(+)$ $:: \text{Nat} \rightarrow \text{Nat} \rightarrow \text{Nat}$

$$0 + n = n$$

$$S(m) + n = S(m + n)$$

$$0 + 0 = 0$$

$$S(0) + 0 = S(0)$$

Right identity - the long way

Goal: $\forall x :: \text{Nat}$

$$x + 0 = x$$

$(+) :: \text{Nat} \rightarrow \text{Nat} \rightarrow \text{Nat}$

$$0 + n = n$$

$$S(m) + n = S(m + n)$$

$$0 + 0 = 0$$

$$S(0) + 0 = S(0)$$

$$S(S(0)) + 0 = ?$$

Right identity - the long way

Goal: $\forall x :: \text{Nat}$

$$x + 0 = x$$

$(+) :: \text{Nat} \rightarrow \text{Nat} \rightarrow \text{Nat}$

$$0 + n = n$$

$$S(m) + n = S(m + n)$$

$$0 + 0 = 0$$

$$S(0) + 0 = S(0)$$

$$S(S(0)) + 0 = ?$$

Right identity - the long way

Goal: $\forall x :: \text{Nat}$

$$x + 0 = x$$

$(+) :: \text{Nat} \rightarrow \text{Nat} \rightarrow \text{Nat}$

$$0 + n = n$$

$$S(m) + n = S(m + n)$$

$$0 + 0 = 0$$

$$S(0) + 0 = S(0)$$

$$S(S(0)) + 0 = ?$$

Right identity - the long way

Goal: $\forall x :: \text{Nat}$

$$x + 0 = x$$

$(+) :: \text{Nat} \rightarrow \text{Nat} \rightarrow \text{Nat}$

$$0 + n = n$$

$$S(m) + n = S(m + n)$$

$$0 + 0 = 0$$

$$S(0) + 0 = S(0)$$

$$S(S(0)) + 0 = ?$$

Right identity - the long way

Goal: $\forall x :: \text{Nat}$

$$x + 0 = x$$

$(+) :: \text{Nat} \rightarrow \text{Nat} \rightarrow \text{Nat}$

$$0 + n = n$$

$$S(m) + n = S(m + n)$$

$$0 + 0 = 0$$

$$S(0) + 0 = S(0)$$

$$S(S(0)) + 0 = S(S(0) + 0)$$

Right identity - the long way

Goal: $\forall x :: \text{Nat}$

$$x + 0 = x$$

$(+) :: \text{Nat} \rightarrow \text{Nat} \rightarrow \text{Nat}$

$$0 + n = n$$

$$S(m) + n = S(m + n)$$

$$0 + 0 = 0$$

$$S(0) + 0 = S(0)$$

$$S(S(0)) + 0 = S(S(0) + 0)$$

Right identity - the long way

Goal: $\forall x :: \text{Nat}$

$$x + 0 = x$$

$(+) :: \text{Nat} \rightarrow \text{Nat} \rightarrow \text{Nat}$

$$0 + n = n$$

$$S(m) + n = S(m + n)$$

$$0 + 0 = 0$$

$$S(0) + 0 = S(0)$$

$$S(S(0)) + 0 = S(S(0) + 0)$$

Right identity - the long way

Goal: $\forall x :: \text{Nat}$

$$x + 0 = x$$

$(+) :: \text{Nat} \rightarrow \text{Nat} \rightarrow \text{Nat}$

$$0 + n = n$$

$$S(m) + n = S(m + n)$$

$$0 + 0 = 0$$

$$S(0) + 0 = S(0)$$

$$S(S(0)) + 0 = S(S(0) + 0)$$

Right identity - the long way

Goal: $\forall x :: \text{Nat}$

$$x + 0 = x$$

$(+) :: \text{Nat} \rightarrow \text{Nat} \rightarrow \text{Nat}$

$$0 + n = n$$

$$S(m) + n = S(m + n)$$

$$0 + 0 = 0$$

$$S(0) + 0 = S(0)$$

$$S(S(0)) + 0 = S(S(0))$$

Right identity - the long way

Goal: $\forall x :: \text{Nat}$

$$x + 0 = x$$

$(+) :: \text{Nat} \rightarrow \text{Nat} \rightarrow \text{Nat}$

$$0 + n = n$$

$$S(m) + n = S(m + n)$$

$$0 + 0 = 0$$

$$S(0) + 0 = S(0)$$

$$S(S(0)) + 0 = S(S(0))$$

Right identity - the long way

Goal: $\forall x :: \text{Nat}$

$$x + 0 = x$$

$(+) :: \text{Nat} \rightarrow \text{Nat} \rightarrow \text{Nat}$

$$0 + n = n$$

$$S(m) + n = S(m + n)$$

$$0 + 0 = 0$$

$$S(0) + 0 = S(0)$$

$$S(S(0)) + 0 = S(S(0))$$

The rest of Nat left as an exercise...

Right identity - the pattern

- Proof for $S(0)$ used proof for 0
- Proof for $S(S(0))$ used proof for $S(0)$
- At each step we look at the last one and think "just one more"
- Exploit the pattern, use a little abstraction.

$$\begin{array}{ccccccc}
 P(0) & \rightarrow & P(S(0)) & \rightarrow & P(S(S(0))) & \dots \\
 / & & \backslash & / & & \backslash & / \\
 0 & & S(0) & & S(S(0)) & & \dots
 \end{array}$$

$$\begin{array}{ccccccc}
 \dots & P(x) & \rightarrow & P(S(x)) & \rightarrow & P(S(S(x))) \\
 & \backslash & / & & \backslash & / \\
 \dots & S(x) & & S(S(x)) & &
 \end{array}$$

Right identity - the details

- Show that if $x + 0 = x$ then $S(x) + 0 = S(x)$
 - Captures the "just one more" part of the proof
- Need to prove the base case as well.
 - In this case the base case is $0 + 0 = 0$
 - So the "just one more" part has somewhere sensible to start from.
 - Otherwise we can also prove that $x + S(0) = x$ for $x > 0$

Induction

To prove $P(n)$ for all $n \in \text{Nat}$

- prove that $P(0)$ holds
- prove that $P(n) \rightarrow P(S\ n)$ holds

To prove $P(xs)$ for all $xs \in [a]$

- prove that $P([])$ holds
- prove that $P(xs) \rightarrow P(x:xs)$

Similar for trees, any other recursive datatypes

Equational Reasoning

Monoid - Left Identity

Goal : \forall bs :: [a]
[] ++ bs = bs

(++) :: [a] -> [a] -> [a]
[] ++ ys = ys
(x:xs) ++ ys = x:(xs ++ ys)

Case: []

Monoid - Left Identity

Goal : $\forall \text{ bs} :: [a]$

$[\] \mathrel{++} \text{ bs} = \text{bs}$

$(++) :: [a] \rightarrow [a] \rightarrow [a]$

$[\] \mathrel{++} \text{ ys} = \text{ys}$

$(x:\text{xs}) \mathrel{++} \text{ ys} = x:(\text{xs} \mathrel{++} \text{ ys})$

Case: $[\]$

Monoid - Left Identity

Goal : $\forall \text{ bs} :: [\text{a}]$
 $[] ++ \text{bs} = \text{bs}$

$(++) :: [\text{a}] \rightarrow [\text{a}] \rightarrow [\text{a}]$
 $[] ++ \text{ys} = \text{ys}$
 $(\text{x}:\text{xs}) ++ \text{ys} = \text{x}:(\text{xs} ++ \text{ys})$

Case: $[]$

$[] ++ \text{bs} = \text{bs}$

Monoid - Left Identity

Goal : $\forall bs :: [a]$
 $[] ++ bs = bs$

$(++) :: [a] \rightarrow [a] \rightarrow [a]$
 $[] ++ ys = ys$
 $(x:xs) ++ ys = x:(xs ++ ys)$

Case: $[]$

$[] ++ bs = bs$

Monoid - Left Identity

Goal : \forall bs :: [a]
[] ++ bs = bs

(++) :: [a] -> [a] -> [a]
[] ++ ys = ys
(x:xs) ++ ys = x:(xs ++ ys)

Case: []

[] ++ bs = bs

Monoid - Left Identity

Goal : \forall bs :: [a]
[] ++ bs = bs

(++) :: [a] -> [a] -> [a]
[] ++ ys = ys
(x:xs) ++ ys = x:(xs ++ ys)

Case: []

[] ++ bs = bs

Monoid - Left Identity

Goal : \forall bs :: [a]
[] ++ bs = bs

(++) :: [a] -> [a] -> [a]
[] ++ ys = ys
(x:xs) ++ ys = x:(xs ++ ys)

Case: []

bs = bs

Monoid - Left Identity

Goal : $\forall \text{ bs} :: [\text{a}]$
 $[] ++ \text{bs} = \text{bs}$

$(++) :: [\text{a}] \rightarrow [\text{a}] \rightarrow [\text{a}]$
 $[] ++ \text{ys} = \text{ys}$
 $(\text{x}:\text{xs}) ++ \text{ys} = \text{x}:(\text{xs} ++ \text{ys})$

Case: $[]$

$\text{bs} = \text{bs}$

Monoid - Left Identity

Goal : $\forall \text{ bs} :: [\text{a}]$
 $[] ++ \text{bs} = \text{bs}$

$(++) :: [\text{a}] \rightarrow [\text{a}] \rightarrow [\text{a}]$
 $[] ++ \text{ys} = \text{ys}$
 $(\text{x}:\text{xs}) ++ \text{ys} = \text{x}:(\text{xs} ++ \text{ys})$

Case: $[]$ ✓

$\text{bs} = \text{bs}$

Monoid - Right Identity

Goal : $\forall \text{ bs} :: [\text{a}]$
 $\text{bs} ++ [] = \text{bs}$

$(++) :: [\text{a}] \rightarrow [\text{a}] \rightarrow [\text{a}]$
 $[] ++ \text{ys} = \text{ys}$
 $(\text{x}:\text{xs}) ++ \text{ys} = \text{x}:(\text{xs} ++ \text{ys})$

Cases: $\text{bs} = []$ and $\text{bs} = (\text{x}:\text{xs})$

Monoid - Right Identity

Goal : $\forall \text{ bs} :: [\text{a}]$
 $\text{bs} ++ [] = \text{bs}$

$(++) :: [\text{a}] \rightarrow [\text{a}] \rightarrow [\text{a}]$
 $[] ++ \text{ys} = \text{ys}$
 $(\text{x}:\text{xs}) ++ \text{ys} = \text{x}:(\text{xs} ++ \text{ys})$

Case: $\text{bs} = []$

Monoid - Right Identity

Goal : $\forall \text{ bs} :: [\text{a}]$
 $\text{bs} ++ [] = \text{bs}$

$(++) :: [\text{a}] \rightarrow [\text{a}] \rightarrow [\text{a}]$
 $[] ++ \text{ys} = \text{ys}$
 $(\text{x}:\text{xs}) ++ \text{ys} = \text{x}:(\text{xs} ++ \text{ys})$

Case: $\text{bs} = []$

$\text{bs} ++ [] = \text{bs}$

Monoid - Right Identity

Goal : $\forall \text{ bs} :: [\text{a}]$
 $\text{bs} ++ [] = \text{bs}$

$(++) :: [\text{a}] \rightarrow [\text{a}] \rightarrow [\text{a}]$
 $[] ++ \text{ys} = \text{ys}$
 $(\text{x}:\text{xs}) ++ \text{ys} = \text{x}:(\text{xs} ++ \text{ys})$

Case: $\text{bs} = []$

$\text{bs} ++ [] = \text{bs}$

Monoid - Right Identity

Goal : $\forall \text{ bs} :: [\text{a}]$
 $\text{bs} ++ [] = \text{bs}$

$(++) :: [\text{a}] \rightarrow [\text{a}] \rightarrow [\text{a}]$
 $[] ++ \text{ys} = \text{ys}$
 $(\text{x}:\text{xs}) ++ \text{ys} = \text{x}:(\text{xs} ++ \text{ys})$

Case: $\text{bs} = []$

$[] ++ [] = []$

Monoid - Right Identity

Goal : $\forall \text{ bs} :: [\text{a}]$
 $\text{bs} ++ [] = \text{bs}$

$(++) :: [\text{a}] \rightarrow [\text{a}] \rightarrow [\text{a}]$
 $[] ++ \text{ys} = \text{ys}$
 $(\text{x}:\text{xs}) ++ \text{ys} = \text{x}:(\text{xs} ++ \text{ys})$

Case: $\text{bs} = []$

$[] ++ [] = []$

Monoid - Right Identity

Goal : \forall bs :: [a]
bs ++ [] = bs

(++) :: [a] -> [a] -> [a]
[] ++ ys = ys
(x:xs) ++ ys = x:(xs ++ ys)

Case: bs = []

[] ++ [] = []

Monoid - Right Identity

Goal : $\forall \text{ bs} :: [\text{a}]$
 $\text{bs} ++ [] = \text{bs}$

$(++) :: [\text{a}] \rightarrow [\text{a}] \rightarrow [\text{a}]$
 $[] ++ \text{ys} = \text{ys}$
 $(\text{x}:\text{xs}) ++ \text{ys} = \text{x}:(\text{xs} ++ \text{ys})$

Case: $\text{bs} = []$

$[] ++ [] = []$

Monoid - Right Identity

Goal : $\forall \text{ bs} :: [\text{a}]$
 $\text{bs} ++ [] = \text{bs}$

$(++) :: [\text{a}] \rightarrow [\text{a}] \rightarrow [\text{a}]$
 $[] ++ \text{ys} = \text{ys}$
 $(\text{x}:\text{xs}) ++ \text{ys} = \text{x}:(\text{xs} ++ \text{ys})$

Case: $\text{bs} = []$

$[] = []$

Monoid - Right Identity

Goal : $\forall \text{ bs} :: [\text{a}]$
 $\text{bs} ++ [] = \text{bs}$

$(++) :: [\text{a}] \rightarrow [\text{a}] \rightarrow [\text{a}]$
 $[] ++ \text{ys} = \text{ys}$
 $(\text{x}:\text{xs}) ++ \text{ys} = \text{x}:(\text{xs} ++ \text{ys})$

Case: $\text{bs} = []$

$[] = []$

Monoid - Right Identity

Goal : $\forall \text{ bs} :: [\text{a}]$
 $\text{bs} ++ [] = \text{bs}$

$(++) :: [\text{a}] \rightarrow [\text{a}] \rightarrow [\text{a}]$
 $[] ++ \text{ys} = \text{ys}$
 $(\text{x}:\text{xs}) ++ \text{ys} = \text{x}:(\text{xs} ++ \text{ys})$

Case: $\text{bs} = []$ ✓

$[] = []$

Monoid - Right Identity

Goal : $\forall \text{ bs} :: [\text{a}]$
 $\text{bs} ++ [] = \text{bs}$

$(++) :: [\text{a}] \rightarrow [\text{a}] \rightarrow [\text{a}]$
 $[] ++ \text{ys} = \text{ys}$
 $(\text{x}:\text{xs}) ++ \text{ys} = \text{x}:(\text{xs} ++ \text{ys})$

Case: $\text{bs} = (\text{x}:\text{xs})$

Assume: $\text{xs} ++ [] = \text{xs}$

Monoid - Right Identity

Goal : $\forall \text{ bs} :: [\text{a}]$

$\text{bs} ++ [] = \text{bs}$

$(++) :: [\text{a}] \rightarrow [\text{a}] \rightarrow [\text{a}]$

$[] ++ \text{ys} = \text{ys}$

$(\text{x}:\text{xs}) ++ \text{ys} = \text{x}:(\text{xs} ++ \text{ys})$

Case: $\text{bs} = (\text{x}:\text{xs})$

Assume: $\text{xs} ++ [] = \text{xs}$

$\text{bs} ++ [] = \text{bs}$

Monoid - Right Identity

Goal : $\forall \text{ bs} :: [\text{a}]$
 $\text{bs} ++ [] = \text{bs}$

$(++) :: [\text{a}] \rightarrow [\text{a}] \rightarrow [\text{a}]$
 $[] ++ \text{ys} = \text{ys}$
 $(\text{x}:\text{xs}) ++ \text{ys} = \text{x}:(\text{xs} ++ \text{ys})$

Case: $\text{bs} = (\text{x}:\text{xs})$

Assume: $\text{xs} ++ [] = \text{xs}$

$\text{bs} ++ [] = \text{bs}$

Monoid - Right Identity

Goal : $\forall \text{ bs} :: [\text{a}]$
 $\text{bs} ++ [] = \text{bs}$

$(++) :: [\text{a}] \rightarrow [\text{a}] \rightarrow [\text{a}]$
 $[] ++ \text{ys} = \text{ys}$
 $(\text{x}:\text{xs}) ++ \text{ys} = \text{x}:(\text{xs} ++ \text{ys})$

Case: $\text{bs} = (\text{x}:\text{xs})$

Assume: $\text{xs} ++ [] = \text{xs}$

$(\text{x}:\text{xs}) ++ [] = (\text{x}:\text{xs})$

Monoid - Right Identity

Goal : $\forall bs :: [a]$
 $bs ++ [] = bs$

$(++) :: [a] \rightarrow [a] \rightarrow [a]$
 $[] ++ ys = ys$
 $(x:xs) ++ ys = x:(xs ++ ys)$

Case: $bs = (x:xs)$

Assume: $xs ++ [] = xs$

$(x:xs) ++ [] = (x:xs)$

Monoid - Right Identity

Goal : $\forall \text{ bs} :: [\text{a}]$
 $\text{bs} ++ [] = \text{bs}$

$(++) :: [\text{a}] \rightarrow [\text{a}] \rightarrow [\text{a}]$
 $[] ++ \text{ys} = \text{ys}$
 $(\text{x}:\text{xs}) ++ \text{ys} = \text{x}:(\text{xs} ++ \text{ys})$

Case: $\text{bs} = (\text{x}:\text{xs})$

Assume: $\text{xs} ++ [] = \text{xs}$

$(\text{x}:\text{xs}) ++ [] = (\text{x}:\text{xs})$

Monoid - Right Identity

Goal : $\forall \text{ bs} :: [\text{a}]$
 $\text{bs} ++ [] = \text{bs}$

$(++) :: [\text{a}] \rightarrow [\text{a}] \rightarrow [\text{a}]$
 $[] ++ \text{ys} = \text{ys}$
 $(\text{x}:\text{xs}) ++ \text{ys} = \text{x}:(\text{xs} ++ \text{ys})$

Case: $\text{bs} = (\text{x}:\text{xs})$

Assume: $\text{xs} ++ [] = \text{xs}$

$(\text{x}:\text{xs}) ++ [] = (\text{x}:\text{xs})$

Monoid - Right Identity

Goal : $\forall \text{ bs} :: [\text{a}]$
 $\text{bs} ++ [] = \text{bs}$

$(++) :: [\text{a}] \rightarrow [\text{a}] \rightarrow [\text{a}]$
 $[] ++ \text{ys} = \text{ys}$
 $(\text{x}:\text{xs}) ++ \text{ys} = \text{x}:(\text{xs} ++ \text{ys})$

Case: $\text{bs} = (\text{x}:\text{xs})$

Assume: $\text{xs} ++ [] = \text{xs}$

$\text{x}:(\text{xs} ++ []) = (\text{x}:\text{xs})$

Monoid - Right Identity

Goal : $\forall \text{ bs} :: [\text{a}]$
 $\text{bs} ++ [] = \text{bs}$

$(++) :: [\text{a}] \rightarrow [\text{a}] \rightarrow [\text{a}]$
 $[] ++ \text{ys} = \text{ys}$
 $(\text{x}:\text{xs}) ++ \text{ys} = \text{x}:(\text{xs} ++ \text{ys})$

Case: $\text{bs} = (\text{x}:\text{xs})$

Assume: $\text{xs} ++ [] = \text{xs}$

$\text{x}:(\text{xs} ++ []) = (\text{x}:\text{xs})$

Monoid - Right Identity

Goal : $\forall \text{ bs} :: [\text{a}]$
 $\text{bs} ++ [] = \text{bs}$

$(++) :: [\text{a}] \rightarrow [\text{a}] \rightarrow [\text{a}]$
 $[] ++ \text{ys} = \text{ys}$
 $(\text{x}:\text{xs}) ++ \text{ys} = \text{x}:(\text{xs} ++ \text{ys})$

Case: $\text{bs} = (\text{x}:\text{xs})$

Assume: $\text{xs} ++ [] = \text{xs}$

$\text{x}:(\text{xs} ++ []) = (\text{x}:\text{xs})$

Monoid - Right Identity

Goal : $\forall \text{ bs} :: [\text{a}]$
 $\text{bs} ++ [] = \text{bs}$

$(++) :: [\text{a}] \rightarrow [\text{a}] \rightarrow [\text{a}]$
 $[] ++ \text{ys} = \text{ys}$
 $(\text{x}:\text{xs}) ++ \text{ys} = \text{x}:(\text{xs} ++ \text{ys})$

Case: $\text{bs} = (\text{x}:\text{xs})$

Assume: $\text{xs} ++ [] = \text{xs}$

$\text{x}:(\text{xs} ++ []) = (\text{x}:\text{xs})$

Monoid - Right Identity

Goal : $\forall \text{ bs} :: [\text{a}]$
 $\text{bs} ++ [] = \text{bs}$

$(++) :: [\text{a}] \rightarrow [\text{a}] \rightarrow [\text{a}]$
 $[] ++ \text{ys} = \text{ys}$
 $(\text{x}:\text{xs}) ++ \text{ys} = \text{x}:(\text{xs} ++ \text{ys})$

Case: $\text{bs} = (\text{x}:\text{xs})$

Assume: $\text{xs} ++ [] = \text{xs}$

$\text{x}:(\text{xs}) = (\text{x}:\text{xs})$

Monoid - Right Identity

Goal : $\forall \text{ bs} :: [\text{a}]$
 $\text{bs} ++ [] = \text{bs}$

$(++) :: [\text{a}] \rightarrow [\text{a}] \rightarrow [\text{a}]$
 $[] ++ \text{ys} = \text{ys}$
 $(\text{x}:\text{xs}) ++ \text{ys} = \text{x}:(\text{xs} ++ \text{ys})$

Case: $\text{bs} = (\text{x}:\text{xs})$

Assume: $\text{xs} ++ [] = \text{xs}$

$(\text{x}:\text{xs}) = (\text{x}:\text{xs})$

Monoid - Right Identity

Goal : $\forall \text{ bs} :: [\text{a}]$
 $\text{bs} ++ [] = \text{bs}$

$(++) :: [\text{a}] \rightarrow [\text{a}] \rightarrow [\text{a}]$
 $[] ++ \text{ys} = \text{ys}$
 $(\text{x}:\text{xs}) ++ \text{ys} = \text{x}:(\text{xs} ++ \text{ys})$

Case: $\text{bs} = (\text{x}:\text{xs}) \checkmark$

Assume: $\text{xs} ++ [] = \text{xs}$

$(\text{x}:\text{xs}) \quad = (\text{x}:\text{xs})$

Monoid - Associativity

Goal : $\forall bs, cs, ds :: [a]$
 $bs ++ (cs ++ ds) = (bs ++ cs) ++ ds$

$(++) :: [a] \rightarrow [a] \rightarrow [a]$
 $[] \quad ++ \text{ys} = \text{ys}$
 $(x:xs) ++ \text{ys} = x:(xs ++ \text{ys})$

Cases: $bs = []$ and $bs = (x:xs)$

Monoid - Associativity

Goal : $\forall bs, cs, ds :: [a]$
 $bs ++ (cs ++ ds) = (bs ++ cs) ++ ds$

$(++) :: [a] \rightarrow [a] \rightarrow [a]$
 $[] ++ ys = ys$
 $(x:xs) ++ ys = x:(xs ++ ys)$

Case: $bs = []$

Monoid - Associativity

Goal : \forall bs, cs, ds :: [a]
bs ++ (cs ++ ds) = (bs ++ cs) ++ ds

(++) :: [a] -> [a] -> [a]
[] ++ ys = ys
(x:xs) ++ ys = x:(xs ++ ys)

Case: bs = []

bs ++ (cs ++ ds) = (bs ++ cs) ++ ds

Monoid - Associativity

Goal : $\forall bs, cs, ds :: [a]$
 $bs ++ (cs ++ ds) = (bs ++ cs) ++ ds$

$(++) :: [a] \rightarrow [a] \rightarrow [a]$
 $[] ++ ys = ys$
 $(x:xs) ++ ys = x:(xs ++ ys)$

Case: $bs = []$

$bs ++ (cs ++ ds) = (bs ++ cs) ++ ds$

Monoid - Associativity

Goal : \forall bs, cs, ds :: [a]
bs ++ (cs ++ ds) = (bs ++ cs) ++ ds

(++) :: [a] -> [a] -> [a]
[] ++ ys = ys
(x:xs) ++ ys = x:(xs ++ ys)

Case: bs = []

[] ++ (cs ++ ds) = ([] ++ cs) ++ ds

Monoid - Associativity

Goal : $\forall bs, cs, ds :: [a]$
 $bs ++ (cs ++ ds) = (bs ++ cs) ++ ds$

$(++) :: [a] \rightarrow [a] \rightarrow [a]$
 $[] ++ ys = ys$
 $(x:xs) ++ ys = x:(xs ++ ys)$

Case: $bs = []$

$[] ++ (cs ++ ds) = ([] ++ cs) ++ ds$

Monoid - Associativity

Goal : $\forall bs, cs, ds :: [a]$
 $bs ++ (cs ++ ds) = (bs ++ cs) ++ ds$

$(++) :: [a] \rightarrow [a] \rightarrow [a]$
 $[] ++ ys = ys$
 $(x:xs) ++ ys = x:(xs ++ ys)$

Case: $bs = []$

$[] ++ (cs ++ ds) = ([] ++ cs) ++ ds$

Monoid - Associativity

Goal : $\forall bs, cs, ds :: [a]$
 $bs ++ (cs ++ ds) = (bs ++ cs) ++ ds$

$(++) :: [a] \rightarrow [a] \rightarrow [a]$
 $[] ++ ys = ys$
 $(x:xs) ++ ys = x:(xs ++ ys)$

Case: $bs = []$

$[] ++ (cs ++ ds) = ([] ++ cs) ++ ds$

Monoid - Associativity

Goal : \forall bs, cs, ds :: [a]
bs ++ (cs ++ ds) = (bs ++ cs) ++ ds

(++) :: [a] -> [a] -> [a]
[] ++ ys = ys
(x:xs) ++ ys = x:(xs ++ ys)

Case: bs = []

cs ++ ds = ([] ++ cs) ++ ds

Monoid - Associativity

Goal : $\forall bs, cs, ds :: [a]$
 $bs ++ (cs ++ ds) = (bs ++ cs) ++ ds$

$(++) :: [a] \rightarrow [a] \rightarrow [a]$
 $[] ++ ys = ys$
 $(x:xs) ++ ys = x:(xs ++ ys)$

Case: $bs = []$

$cs ++ ds = ([] ++ cs) ++ ds$

Monoid - Associativity

Goal : $\forall bs, cs, ds :: [a]$
 $bs ++ (cs ++ ds) = (bs ++ cs) ++ ds$

$(++) :: [a] \rightarrow [a] \rightarrow [a]$
 $[] ++ ys = ys$
 $(x:xs) ++ ys = x:(xs ++ ys)$

Case: $bs = []$

$cs ++ ds = ([] ++ cs) ++ ds$

Monoid - Associativity

Goal : $\forall bs, cs, ds :: [a]$
 $bs ++ (cs ++ ds) = (bs ++ cs) ++ ds$

$(++) :: [a] \rightarrow [a] \rightarrow [a]$
 $[] ++ ys = ys$
 $(x:xs) ++ ys = x:(xs ++ ys)$

Case: $bs = []$

$cs ++ ds = ([] ++ cs) ++ ds$

Monoid - Associativity

Goal : $\forall bs, cs, ds :: [a]$
 $bs ++ (cs ++ ds) = (bs ++ cs) ++ ds$

$(++) :: [a] \rightarrow [a] \rightarrow [a]$
 $[] ++ ys = ys$
 $(x:xs) ++ ys = x:(xs ++ ys)$

Case: $bs = []$

$cs ++ ds = (cs) ++ ds$

Monoid - Associativity

Goal : $\forall bs, cs, ds :: [a]$
 $bs ++ (cs ++ ds) = (bs ++ cs) ++ ds$

$(++) :: [a] \rightarrow [a] \rightarrow [a]$
 $[] ++ ys = ys$
 $(x:xs) ++ ys = x:(xs ++ ys)$

Case: $bs = []$

$cs ++ ds = cs ++ ds$

Monoid - Associativity

Goal : $\forall bs, cs, ds :: [a]$
 $bs ++ (cs ++ ds) = (bs ++ cs) ++ ds$

$(++) :: [a] \rightarrow [a] \rightarrow [a]$
 $[] ++ ys = ys$
 $(x:xs) ++ ys = x:(xs ++ ys)$

Case: $bs = []$ ✓

$cs ++ ds = cs ++ ds$

Monoid - Associativity

Goal : $\forall bs, cs, ds :: [a]$
 $bs ++ (cs ++ ds) = (bs ++ cs) ++ ds$

$(++) :: [a] \rightarrow [a] \rightarrow [a]$
 $[] ++ ys = ys$
 $(x:xs) ++ ys = x:(xs ++ ys)$

Case: $bs = (x:xs)$

Assume: $xs ++ (cs ++ ds) = (xs ++ cs) ++ ds$

Monoid - Associativity

Goal : $\forall bs, cs, ds :: [a]$
 $bs ++ (cs ++ ds) = (bs ++ cs) ++ ds$

$(++) :: [a] \rightarrow [a] \rightarrow [a]$
 $[] ++ ys = ys$
 $(x:xs) ++ ys = x:(xs ++ ys)$

Case: $bs = (x:xs)$

Assume: $xs ++ (cs ++ ds) = (xs ++ cs) ++ ds$

$bs ++ (cs ++ ds) = \quad (bs ++ cs) ++ ds$

Monoid - Associativity

Goal : $\forall bs, cs, ds :: [a]$
 $bs ++ (cs ++ ds) = (bs ++ cs) ++ ds$

$(++) :: [a] \rightarrow [a] \rightarrow [a]$
 $[] ++ ys = ys$
 $(x:xs) ++ ys = x:(xs ++ ys)$

Case: $bs = (x:xs)$

Assume: $xs ++ (cs ++ ds) = (xs ++ cs) ++ ds$

$bs ++ (cs ++ ds) = (bs ++ cs) ++ ds$

Monoid - Associativity

Goal : $\forall bs, cs, ds :: [a]$
 $bs ++ (cs ++ ds) = (bs ++ cs) ++ ds$

$(++) :: [a] \rightarrow [a] \rightarrow [a]$
 $[] ++ ys = ys$
 $(x:xs) ++ ys = x:(xs ++ ys)$

Case: $bs = (x:xs)$

Assume: $xs ++ (cs ++ ds) = (xs ++ cs) ++ ds$

$(x:xs) ++ (cs ++ ds) = ((x:xs) ++ cs) ++ ds$

Monoid - Associativity

Goal : $\forall bs, cs, ds :: [a]$
 $bs ++ (cs ++ ds) = (bs ++ cs) ++ ds$

$(++) :: [a] \rightarrow [a] \rightarrow [a]$
 $[] ++ ys = ys$
 $(x:xs) ++ ys = x:(xs ++ ys)$

Case: $bs = (x:xs)$

Assume: $xs ++ (cs ++ ds) = (xs ++ cs) ++ ds$

$(x:xs) ++ (cs ++ ds) = ((x:xs) ++ cs) ++ ds$

Monoid - Associativity

Goal : $\forall bs, cs, ds :: [a]$
 $bs ++ (cs ++ ds) = (bs ++ cs) ++ ds$

$(++) :: [a] \rightarrow [a] \rightarrow [a]$
 $[] ++ ys = ys$
 $(x:xs) ++ ys = x:(xs ++ ys)$

Case: $bs = (x:xs)$

Assume: $xs ++ (cs ++ ds) = (xs ++ cs) ++ ds$

$(x:xs) ++ (cs ++ ds) = ((x:xs) ++ cs) ++ ds$

Monoid - Associativity

Goal : $\forall \text{ bs, cs, ds} :: [\text{a}]$
 $\text{bs} ++ (\text{cs} ++ \text{ds}) = (\text{bs} ++ \text{cs}) ++ \text{ds}$

$(++) :: [\text{a}] \rightarrow [\text{a}] \rightarrow [\text{a}]$
 $[] ++ \text{ys} = \text{ys}$
 $(\text{x}:\text{xs}) ++ \text{ys} = \text{x}:(\text{xs} ++ \text{ys})$

Case: $\text{bs} = (\text{x}:\text{xs})$

Assume: $\text{xs} ++ (\text{cs} ++ \text{ds}) = (\text{xs} ++ \text{cs}) ++ \text{ds}$

$(\text{x}:\text{xs}) ++ (\text{cs} ++ \text{ds}) = ((\text{x}:\text{xs}) ++ \text{cs}) ++ \text{ds}$

Monoid - Associativity

Goal : $\forall \text{ bs, cs, ds} :: [\text{a}]$
 $\text{bs} ++ (\text{cs} ++ \text{ds}) = (\text{bs} ++ \text{cs}) ++ \text{ds}$

$(++) :: [\text{a}] \rightarrow [\text{a}] \rightarrow [\text{a}]$
 $[] ++ \text{ys} = \text{ys}$
 $(\text{x}:\text{xs}) ++ \text{ys} = \text{x}:(\text{xs} ++ \text{ys})$

Case: $\text{bs} = (\text{x}:\text{xs})$

Assume: $\text{xs} ++ (\text{cs} ++ \text{ds}) = (\text{xs} ++ \text{cs}) ++ \text{ds}$

$\text{x}:(\text{xs} ++ (\text{cs} ++ \text{ds})) = ((\text{x}:\text{xs}) ++ \text{cs}) ++ \text{ds}$

Monoid - Associativity

Goal : $\forall bs, cs, ds :: [a]$
 $bs ++ (cs ++ ds) = (bs ++ cs) ++ ds$

$(++) :: [a] \rightarrow [a] \rightarrow [a]$
 $[] ++ ys = ys$
 $(x:xs) ++ ys = x:(xs ++ ys)$

Case: $bs = (x:xs)$

Assume: $xs ++ (cs ++ ds) = (xs ++ cs) ++ ds$

$x:(xs ++ (cs ++ ds)) = ((x:xs) ++ cs) ++ ds$

Monoid - Associativity

Goal : $\forall bs, cs, ds :: [a]$
 $bs ++ (cs ++ ds) = (bs ++ cs) ++ ds$

$(++) :: [a] \rightarrow [a] \rightarrow [a]$
 $[] ++ ys = ys$
 $(x:xs) ++ ys = x:(xs ++ ys)$

Case: $bs = (x:xs)$

Assume: $xs ++ (cs ++ ds) = (xs ++ cs) ++ ds$

$x:(xs ++ (cs ++ ds)) = ((x:xs) ++ cs) ++ ds$

Monoid - Associativity

Goal : $\forall \text{ bs, cs, ds} :: [\text{a}]$
 $\text{bs} ++ (\text{cs} ++ \text{ds}) = (\text{bs} ++ \text{cs}) ++ \text{ds}$

$(++) :: [\text{a}] \rightarrow [\text{a}] \rightarrow [\text{a}]$
 $[] ++ \text{ys} = \text{ys}$
 $(\text{x}:\text{xs}) ++ \text{ys} = \text{x}:(\text{xs} ++ \text{ys})$

Case: $\text{bs} = (\text{x}:\text{xs})$

Assume: $\text{xs} ++ (\text{cs} ++ \text{ds}) = (\text{xs} ++ \text{cs}) ++ \text{ds}$

$\text{x}:(\text{xs} ++ (\text{cs} ++ \text{ds})) = ((\text{x}:\text{xs}) ++ \text{cs}) ++ \text{ds}$

Monoid - Associativity

Goal : $\forall bs, cs, ds :: [a]$
 $bs ++ (cs ++ ds) = (bs ++ cs) ++ ds$

$(++) :: [a] \rightarrow [a] \rightarrow [a]$
 $[] ++ ys = ys$
 $(x:xs) ++ ys = x:(xs ++ ys)$

Case: $bs = (x:xs)$

Assume: $xs ++ (cs ++ ds) = (xs ++ cs) ++ ds$

$x:(xs ++ (cs ++ ds)) = (x:(xs ++ cs)) ++ ds$

Monoid - Associativity

Goal : $\forall bs, cs, ds :: [a]$
 $bs ++ (cs ++ ds) = (bs ++ cs) ++ ds$

$(++) :: [a] \rightarrow [a] \rightarrow [a]$
 $[] ++ ys = ys$
 $(x:xs) ++ ys = x:(xs ++ ys)$

Case: $bs = (x:xs)$

Assume: $xs ++ (cs ++ ds) = (xs ++ cs) ++ ds$

$x:(xs ++ (cs ++ ds)) = (x:(xs ++ cs)) ++ ds$

Monoid - Associativity

Goal : $\forall bs, cs, ds :: [a]$
 $bs ++ (cs ++ ds) = (bs ++ cs) ++ ds$

$(++) :: [a] \rightarrow [a] \rightarrow [a]$
 $[] ++ ys = ys$
 $(x:xs) ++ ys = x:(xs ++ ys)$

Case: $bs = (x:xs)$

Assume: $xs ++ (cs ++ ds) = (xs ++ cs) ++ ds$

$x:(xs ++ (cs ++ ds)) = (x:(xs ++ cs)) ++ ds$

Monoid - Associativity

Goal : $\forall bs, cs, ds :: [a]$
 $bs ++ (cs ++ ds) = (bs ++ cs) ++ ds$

$(++) :: [a] \rightarrow [a] \rightarrow [a]$
 $[] ++ ys = ys$
 $(x:xs) ++ ys = x:(xs ++ ys)$

Case: $bs = (x:xs)$

Assume: $xs ++ (cs ++ ds) = (xs ++ cs) ++ ds$

$x:(xs ++ (cs ++ ds)) = (x:(xs ++ cs)) ++ ds$

Monoid - Associativity

Goal : $\forall bs, cs, ds :: [a]$
 $bs ++ (cs ++ ds) = (bs ++ cs) ++ ds$

$(++) :: [a] \rightarrow [a] \rightarrow [a]$
 $[] ++ ys = ys$
 $(x:xs) ++ ys = x:(xs ++ ys)$

Case: $bs = (x:xs)$

Assume: $xs ++ (cs ++ ds) = (xs ++ cs) ++ ds$

$x:(xs ++ (cs ++ ds)) = x:((xs ++ cs) ++ ds)$

Monoid - Associativity

Goal : $\forall bs, cs, ds :: [a]$
 $bs ++ (cs ++ ds) = (bs ++ cs) ++ ds$

$(++) :: [a] \rightarrow [a] \rightarrow [a]$
 $[] ++ ys = ys$
 $(x:xs) ++ ys = x:(xs ++ ys)$

Case: $bs = (x:xs)$

Assume: $xs ++ (cs ++ ds) = (xs ++ cs) ++ ds$

$x:(xs ++ (cs ++ ds)) = x:((xs ++ cs) ++ ds)$

Monoid - Associativity

Goal : $\forall bs, cs, ds :: [a]$
 $bs ++ (cs ++ ds) = (bs ++ cs) ++ ds$

$(++) :: [a] \rightarrow [a] \rightarrow [a]$
 $[] ++ ys = ys$
 $(x:xs) ++ ys = x:(xs ++ ys)$

Case: $bs = (x:xs)$

Assume: $xs ++ (cs ++ ds) = (xs ++ cs) ++ ds$

$x:(xs ++ (cs ++ ds)) = x:((xs ++ cs) ++ ds)$

Monoid - Associativity

Goal : $\forall bs, cs, ds :: [a]$
 $bs ++ (cs ++ ds) = (bs ++ cs) ++ ds$

$(++) :: [a] \rightarrow [a] \rightarrow [a]$
 $[] ++ ys = ys$
 $(x:xs) ++ ys = x:(xs ++ ys)$

Case: $bs = (x:xs)$

Assume: $xs ++ (cs ++ ds) = (xs ++ cs) ++ ds$

$x:(xs ++ (cs ++ ds)) = x:((xs ++ cs) ++ ds)$

Monoid - Associativity

Goal : $\forall bs, cs, ds :: [a]$
 $bs ++ (cs ++ ds) = (bs ++ cs) ++ ds$

$(++) :: [a] \rightarrow [a] \rightarrow [a]$
 $[] ++ ys = ys$
 $(x:xs) ++ ys = x:(xs ++ ys)$

Case: $bs = (x:xs)$

Assume: $xs ++ (cs ++ ds) = (xs ++ cs) ++ ds$

$x:((xs ++ cs) ++ ds) = x:((xs ++ cs) ++ ds)$

Monoid - Associativity

Goal : $\forall bs, cs, ds :: [a]$
 $bs ++ (cs ++ ds) = (bs ++ cs) ++ ds$

$(++) :: [a] \rightarrow [a] \rightarrow [a]$
 $[] ++ ys = ys$
 $(x:xs) ++ ys = x:(xs ++ ys)$

Case: $bs = (x:xs)$

Assume: $xs ++ (cs ++ ds) = (xs ++ cs) ++ ds$

$x:((xs ++ cs) ++ ds) = x:((xs ++ cs) ++ ds)$

Monoid - Associativity

Goal : $\forall bs, cs, ds :: [a]$
 $bs ++ (cs ++ ds) = (bs ++ cs) ++ ds$

$(++) :: [a] \rightarrow [a] \rightarrow [a]$
 $[] ++ ys = ys$
 $(x:xs) ++ ys = x:(xs ++ ys)$

Case: $bs = (x:xs)$ ✓

Assume: $xs ++ (cs ++ ds) = (xs ++ cs) ++ ds$

$x:((xs ++ cs) ++ ds) = x:((xs ++ cs) ++ ds)$

Monoid - Associativity - Bird style

$bs \ ++ \ (cs \ ++ \ ds) \ = \ (bs \ ++ \ cs) \ ++ \ ds$

$bs = x:xs$

$(x:xs) \ ++ \ (cs \ ++ \ ds) \ = \ ((x:xs) \ ++ \ cs) \ ++ \ ds$

second rule for ++

$x:(xs \ ++ \ (cs \ ++ \ ds)) \ = \ ((x:xs) \ ++ \ cs) \ ++ \ ds$

second rule for ++

$x:(xs \ ++ \ (cs \ ++ \ ds)) \ = \ (x:(xs \ ++ \ cs)) \ ++ \ ds$

second rule for ++

$x:(xs \ ++ \ (cs \ ++ \ ds)) \ = \ x:((xs \ ++ \ cs) \ ++ \ ds)$

inductive hypothesis

$x:((xs \ ++ \ cs) \ ++ \ ds) \ = \ x:((xs \ ++ \ cs) \ ++ \ ds)$

Functor - Identity

Goal : \forall bs :: [a]
map id bs = bs

map :: (a -> b) -> [a] -> [b]
map f [] = []
map f (x:xs) = f x : map f xs

Cases: bs = [] and bs = (x:xs)

Functor - Identity

Goal : \forall bs :: [a]
map id bs = bs

map :: (a -> b) -> [a] -> [b]
map f [] = []
map f (x:xs) = f x : map f xs

Case: bs = []

Functor - Identity

Goal : \forall bs :: [a]
map id bs = bs

map :: (a -> b) -> [a] -> [b]
map f [] = []
map f (x:xs) = f x : map f xs

Case: bs = []

map id bs = bs

Functor - Identity

Goal : \forall bs :: [a]
map id bs = bs

map :: (a -> b) -> [a] -> [b]
map f [] = []
map f (x:xs) = f x : map f xs

Case: bs = []

map id bs = bs

Functor - Identity

Goal : \forall bs :: [a]
map id bs = bs

map :: (a -> b) -> [a] -> [b]
map f [] = []
map f (x:xs) = f x : map f xs

Case: bs = []

map id [] = []

Functor - Identity

Goal : \forall bs :: [a]
map id bs = bs

map :: (a -> b) -> [a] -> [b]
map f [] = []
map f (x:xs) = f x : map f xs

Case: bs = []

map id [] = []

Functor - Identity

Goal : \forall bs :: [a]
map id bs = bs

map :: (a -> b) -> [a] -> [b]
map f [] = []
map f (x:xs) = f x : map f xs

Case: bs = []

map id [] = []

Functor - Identity

Goal : \forall bs :: [a]
map id bs = bs

map :: (a -> b) -> [a] -> [b]
map f [] = []
map f (x:xs) = f x : map f xs

Case: bs = []

map id [] = []

Functor - Identity

Goal : \forall bs :: [a]
map id bs = bs

map :: (a -> b) -> [a] -> [b]
map f [] = []
map f (x:xs) = f x : map f xs

Case: bs = []

[] = []

Functor - Identity

Goal : \forall bs :: [a]
map id bs = bs

map :: (a -> b) -> [a] -> [b]
map f [] = []
map f (x:xs) = f x : map f xs

Case: bs = []

[] = []

Functor - Identity

Goal : \forall bs :: [a]
map id bs = bs

map :: (a -> b) -> [a] -> [b]
map f [] = []
map f (x:xs) = f x : map f xs

Case: **bs = []** ✓

[] = []

Functor - Identity

Goal : \forall bs :: [a]
map id bs = bs

map :: (a -> b) -> [a] -> [b]
map f [] = []
map f (x:xs) = f x : map f xs

Case: bs = (x:xs)

Assume: map id xs = xs

Functor - Identity

Goal : \forall bs :: [a]
map id bs = bs

map :: (a -> b) -> [a] -> [b]
map f [] = []
map f (x:xs) = f x : map f xs

Case: bs = (x:xs)

Assume: map id xs = xs

map id bs = bs

Functor - Identity

Goal : \forall bs :: [a]
map id bs = bs

map :: (a -> b) -> [a] -> [b]
map f [] = []
map f (x:xs) = f x : map f xs

Case: bs = (x:xs)

Assume: map id xs = xs

map id bs = bs

Functor - Identity

Goal : \forall bs :: [a]
map id bs = bs

map :: (a -> b) -> [a] -> [b]
map f [] = []
map f (x:xs) = f x : map f xs

Case: bs = (x:xs)

Assume: map id xs = xs

map id (x : xs) = (x : xs)

Functor - Identity

Goal : $\forall bs :: [a]$
map id bs = bs

map :: (a -> b) -> [a] -> [b]
map f [] = []
map f (x:xs) = f x : map f xs

Case: bs = (x:xs)

Assume: map id xs = xs

map id (x : xs) = (x : xs)

Functor - Identity

Goal : \forall bs :: [a]
map id bs = bs

map :: (a -> b) -> [a] -> [b]
map f [] = []
map f (x:xs) = f x : map f xs

Case: bs = (x:xs)

Assume: map id xs = xs

map id (x : xs) = (x : xs)

Functor - Identity

Goal : \forall bs :: [a]
map id bs = bs

map :: (a -> b) -> [a] -> [b]
map f [] = []
map f (x:xs) = f x : map f xs

Case: bs = (x:xs)

Assume: map id xs = xs

map id (x : xs) = (x : xs)

Functor - Identity

Goal : \forall bs :: [a]
map id bs = bs

map :: (a -> b) -> [a] -> [b]
map f [] = []
map f (x:xs) = f x : map f xs

Case: bs = (x:xs)

Assume: map id xs = xs

id x : map id xs = (x : xs)

Functor - Identity

Goal : \forall bs :: [a]
map id bs = bs

map :: (a -> b) -> [a] -> [b]
map f [] = []
map f (x:xs) = f x : map f xs

Case: bs = (x:xs)

Assume: map id xs = xs

id x : map id xs = (x : xs)

Functor - Identity

Goal : \forall bs :: [a]
map id bs = bs

map :: (a -> b) -> [a] -> [b]
map f [] = []
map f (x:xs) = f x : map f xs

Case: bs = (x:xs)

Assume: map id xs = xs

x : map id xs = (x : xs)

Functor - Identity

Goal : \forall bs :: [a]
map id bs = bs

map :: (a -> b) -> [a] -> [b]
map f [] = []
map f (x:xs) = f x : map f xs

Case: bs = (x:xs)

Assume: map id xs = xs

x : map id xs = (x : xs)

Functor - Identity

Goal : \forall bs :: [a]
map id bs = bs

map :: (a -> b) -> [a] -> [b]
map f [] = []
map f (x:xs) = f x : map f xs

Case: bs = (x:xs)

Assume: map id xs = xs

x : map id xs = (x : xs)

Functor - Identity

Goal : \forall bs :: [a]
map id bs = bs

map :: (a -> b) -> [a] -> [b]
map f [] = []
map f (x:xs) = f x : map f xs

Case: bs = (x:xs)

Assume: map id xs = xs

x : map id xs = (x : xs)

Functor - Identity

Goal : $\forall bs :: [a]$
map id bs = bs

map :: (a -> b) -> [a] -> [b]
map f [] = []
map f (x:xs) = f x : map f xs

Case: bs = (x:xs)

Assume: map id xs = xs

x : xs = (x : xs)

Functor - Identity

Goal : $\forall bs :: [a]$
map id bs = bs

map :: (a -> b) -> [a] -> [b]
map f [] = []
map f (x:xs) = f x : map f xs

Case: **bs = (x:xs)**

Assume: map id xs = xs

(x : xs) = (x : xs)

Functor - Identity

Goal : $\forall bs :: [a]$
map id bs = bs

map :: (a -> b) -> [a] -> [b]
map f [] = []
map f (x:xs) = f x : map f xs

Case: **bs = (x:xs)** ✓

Assume: map id xs = xs

(x : xs) = (x : xs)

Functor - Composition

Goal : $\forall bs :: [a], f :: a \rightarrow b, g :: b \rightarrow c$
map g (map f bs) = map (g . f) bs

map :: (a -> b) -> [a] -> [b]
map f [] = []
map f (x:xs) = f x : map f xs

Cases: bs = [] and bs = (x:xs)

Functor - Composition

Goal : $\forall bs :: [a], f :: a \rightarrow b, g :: b \rightarrow c$
map g (map f bs) = map (g . f) bs

map :: (a -> b) -> [a] -> [b]
map f [] = []
map f (x:xs) = f x : map f xs

Case: **bs = []**

Functor - Composition

Goal : $\forall bs :: [a], f :: a \rightarrow b, g :: b \rightarrow c$
 $\text{map } g (\text{map } f bs) = \text{map } (g . f) bs$

$\text{map} :: (a \rightarrow b) \rightarrow [a] \rightarrow [b]$
 $\text{map } f [] = []$
 $\text{map } f (x:xs) = f x : \text{map } f xs$

Case: $bs = []$

$\text{map } g (\text{map } f bs) = \text{map } (g . f) bs$

Functor - Composition

Goal : $\forall bs :: [a], f :: a \rightarrow b, g :: b \rightarrow c$
 $\text{map } g (\text{map } f bs) = \text{map } (g \circ f) bs$

$\text{map} :: (a \rightarrow b) \rightarrow [a] \rightarrow [b]$
 $\text{map } f [] = []$
 $\text{map } f (x:xs) = f x : \text{map } f xs$

Case: $bs = []$

$\text{map } g (\text{map } f bs) = \text{map } (g \circ f) bs$

Functor - Composition

Goal : $\forall bs :: [a], f :: a \rightarrow b, g :: b \rightarrow c$
map g (map f bs) = map (g . f) bs

map :: (a -> b) -> [a] -> [b]
map f [] = []
map f (x:xs) = f x : map f xs

Case: **bs = []**

map g (map f []) = map (g . f) []

Functor - Composition

Goal : $\forall bs :: [a], f :: a \rightarrow b, g :: b \rightarrow c$
map g (map f bs) = map (g . f) bs

map :: (a -> b) -> [a] -> [b]
map f [] = []
map f (x:xs) = f x : map f xs

Case: **bs = []**

map g (map f []) = map (g . f) []

Functor - Composition

Goal : $\forall bs :: [a], f :: a \rightarrow b, g :: b \rightarrow c$
map g (map f bs) = map ($g \cdot f$) bs

map :: (a -> b) -> [a] -> [b]

map f [] = []

map f (x:xs) = f x : map f xs

Case: $bs = []$

map g (map f []) = map ($g \cdot f$) []

Functor - Composition

Goal : $\forall bs :: [a], f :: a \rightarrow b, g :: b \rightarrow c$
map g (map f bs) = map (g . f) bs

map :: (a -> b) -> [a] -> [b]

map f [] = []

map f (x:xs) = f x : map f xs

Case: bs = []

map g (map f []) = map (g . f) []

Functor - Composition

Goal : $\forall bs :: [a], f :: a \rightarrow b, g :: b \rightarrow c$
map g (map f bs) = map (g . f) bs

map :: (a -> b) -> [a] -> [b]

map f [] = []

map f (x:xs) = f x : map f xs

Case: bs = []

map g (map f []) = []

Functor - Composition

Goal : $\forall bs :: [a], f :: a \rightarrow b, g :: b \rightarrow c$
map g (map f bs) = map (g . f) bs

map :: (a -> b) -> [a] -> [b]
map f [] = []
map f (x:xs) = f x : map f xs

Case: bs = []

map g (map f []) = []

Functor - Composition

Goal : $\forall bs :: [a], f :: a \rightarrow b, g :: b \rightarrow c$
map g (map f bs) = map (g . f) bs

map :: (a -> b) -> [a] -> [b]
map f [] = []
map f (x:xs) = f x : map f xs

Case: bs = []

map g (map f []) = []

Functor - Composition

Goal : $\forall bs :: [a], f :: a \rightarrow b, g :: b \rightarrow c$
map g (map f bs) = map (g . f) bs

map :: (a -> b) -> [a] -> [b]

map f [] = []

map f (x:xs) = f x : map f xs

Case: bs = []

map g (map f []) = []

Functor - Composition

Goal : $\forall bs :: [a], f :: a \rightarrow b, g :: b \rightarrow c$
map g (map f bs) = map (g . f) bs

map :: (a -> b) -> [a] -> [b]

map f [] = []

map f (x:xs) = f x : map f xs

Case: bs = []

map g [] = []

Functor - Composition

Goal : $\forall bs :: [a], f :: a \rightarrow b, g :: b \rightarrow c$
 $\text{map } g (\text{map } f bs) = \text{map } (g \circ f) bs$

$\text{map} :: (a \rightarrow b) \rightarrow [a] \rightarrow [b]$
 $\text{map } f [] = []$
 $\text{map } f (x:xs) = f x : \text{map } f xs$

Case: $bs = []$

$\text{map } g \quad [] = \quad []$

Functor - Composition

Goal : $\forall bs :: [a], f :: a \rightarrow b, g :: b \rightarrow c$
map g (map f bs) = map (g . f) bs

map :: (a -> b) -> [a] -> [b]

map f [] = []

map f (x:xs) = f x : map f xs

Case: bs = []

map g [] = []

Functor - Composition

Goal : $\forall bs :: [a], f :: a \rightarrow b, g :: b \rightarrow c$
map g (map f bs) = map (g . f) bs

map :: (a -> b) -> [a] -> [b]

map f [] = []

map f (x:xs) = f x : map f xs

Case: bs = []

map g [] = []

Functor - Composition

Goal : $\forall bs :: [a], f :: a \rightarrow b, g :: b \rightarrow c$
 $\text{map } g (\text{map } f bs) = \text{map } (g \circ f) bs$

$\text{map} :: (a \rightarrow b) \rightarrow [a] \rightarrow [b]$

$\text{map } f [] = []$

$\text{map } f (x:xs) = f x : \text{map } f xs$

Case: $bs = []$

$[] = []$

Functor - Composition

Goal : $\forall bs :: [a], f :: a \rightarrow b, g :: b \rightarrow c$
map g (map f bs) = map (g . f) bs

map :: (a -> b) -> [a] -> [b]
map f [] = []
map f (x:xs) = f x : map f xs

Case: **bs** = []

[] = []

Functor - Composition

Goal : $\forall bs :: [a], f :: a \rightarrow b, g :: b \rightarrow c$
 $\text{map } g (\text{map } f bs) = \text{map } (g \circ f) bs$

$\text{map} :: (a \rightarrow b) \rightarrow [a] \rightarrow [b]$
 $\text{map } f [] = []$
 $\text{map } f (x:xs) = f x : \text{map } f xs$

Case: $bs = []$ ✓

$[] = []$

Functor - Composition

Goal : $\forall bs :: [a], f :: a \rightarrow b, g :: b \rightarrow c$
map g (map f bs) = map (g . f) bs

map :: (a -> b) -> [a] -> [b]
map f [] = []
map f (x:xs) = f x : map f xs

Case: **bs = (x:xs)**

Assume: map g (map f xs) = map (g . f) xs

Functor - Composition

Goal : $\forall bs :: [a], f :: a \rightarrow b, g :: b \rightarrow c$
 $\text{map } g (\text{map } f bs) = \text{map } (g . f) bs$

$\text{map} :: (a \rightarrow b) \rightarrow [a] \rightarrow [b]$
 $\text{map } f [] = []$
 $\text{map } f (x:xs) = f x : \text{map } f xs$

Case: $bs = (x:xs)$

Assume: $\text{map } g (\text{map } f xs) = \text{map } (g . f) xs$

$$\text{map } g (\text{map } f bs) = \text{map } (g . f) bs$$

Functor - Composition

Goal : $\forall bs :: [a], f :: a \rightarrow b, g :: b \rightarrow c$
 $\text{map } g (\text{map } f bs) = \text{map } (g \cdot f) bs$

$\text{map} :: (a \rightarrow b) \rightarrow [a] \rightarrow [b]$
 $\text{map } f [] = []$
 $\text{map } f (x:xs) = f x : \text{map } f xs$

Case: $bs = (x:xs)$

Assume: $\text{map } g (\text{map } f xs) = \text{map } (g \cdot f) xs$

$$\text{map } g (\text{map } f \text{ } bs) = \text{map } (g \cdot f) \text{ } bs$$

Functor - Composition

Goal : $\forall bs :: [a], f :: a \rightarrow b, g :: b \rightarrow c$
 $\text{map } g (\text{map } f bs) = \text{map } (g \circ f) bs$

$\text{map} :: (a \rightarrow b) \rightarrow [a] \rightarrow [b]$
 $\text{map } f [] = []$
 $\text{map } f (x:xs) = f x : \text{map } f xs$

Case: $bs = (x:xs)$

Assume: $\text{map } g (\text{map } f xs) = \text{map } (g \circ f) xs$

$$\text{map } g (\text{map } f (x:xs)) = \text{map } (g \circ f) (x:xs)$$

Functor - Composition

Goal : $\forall bs :: [a], f :: a \rightarrow b, g :: b \rightarrow c$
 $\text{map } g (\text{map } f bs) = \text{map } (g \cdot f) bs$

$\text{map} :: (a \rightarrow b) \rightarrow [a] \rightarrow [b]$
 $\text{map } f [] = []$
 $\text{map } f (x:xs) = f x : \text{map } f xs$

Case: $bs = (x:xs)$

Assume: $\text{map } g (\text{map } f xs) = \text{map } (g \cdot f) xs$

$$\text{map } g (\text{map } f (x:xs)) = \text{map } (g \cdot f) (x:xs)$$

Functor - Composition

Goal : $\forall bs :: [a], f :: a \rightarrow b, g :: b \rightarrow c$
 $\text{map } g (\text{map } f bs) = \text{map } (g \cdot f) bs$

$\text{map} :: (a \rightarrow b) \rightarrow [a] \rightarrow [b]$
 $\text{map } f [] = []$
 $\text{map } f (x:xs) = f x : \text{map } f xs$

Case: $bs = (x:xs)$

Assume: $\text{map } g (\text{map } f xs) = \text{map } (g \cdot f) xs$

$$\text{map } g (\text{map } f (x:xs)) = \text{map } (g \cdot f) (x:xs)$$

Functor - Composition

Goal : $\forall bs :: [a], f :: a \rightarrow b, g :: b \rightarrow c$
 $\text{map } g (\text{map } f bs) = \text{map } (g \cdot f) bs$

$\text{map} :: (a \rightarrow b) \rightarrow [a] \rightarrow [b]$
 $\text{map } f [] = []$
 $\text{map } f (x:xs) = f x : \text{map } f xs$

Case: $bs = (x:xs)$

Assume: $\text{map } g (\text{map } f xs) = \text{map } (g \cdot f) xs$

$$\text{map } g (\text{map } f (x:xs)) = \text{map } (g \cdot f) (x:xs)$$

Functor - Composition

Goal : $\forall bs :: [a], f :: a \rightarrow b, g :: b \rightarrow c$
 $\text{map } g (\text{map } f bs) = \text{map } (g \cdot f) bs$

$\text{map} :: (a \rightarrow b) \rightarrow [a] \rightarrow [b]$
 $\text{map } f [] = []$
 $\text{map } f (x:xs) = f x : \text{map } f xs$

Case: $bs = (x:xs)$

Assume: $\text{map } g (\text{map } f xs) = \text{map } (g \cdot f) xs$

$$\text{map } g (\text{map } f (x:xs)) = (g \cdot f) x : \text{map } (g \cdot f) xs$$

Functor - Composition

Goal : $\forall bs :: [a], f :: a \rightarrow b, g :: b \rightarrow c$
 $\text{map } g (\text{map } f bs) = \text{map } (g \cdot f) bs$

$\text{map} :: (a \rightarrow b) \rightarrow [a] \rightarrow [b]$
 $\text{map } f [] = []$
 $\text{map } f (x:xs) = f x : \text{map } f xs$

Case: $bs = (x:xs)$

Assume: $\text{map } g (\text{map } f xs) = \text{map } (g \cdot f) xs$

$$\text{map } g (\text{map } f (x:xs)) = (g \cdot f) x : \text{map } (g \cdot f) xs$$

Functor - Composition

Goal : $\forall bs :: [a], f :: a \rightarrow b, g :: b \rightarrow c$
 $\text{map } g (\text{map } f bs) = \text{map } (g . f) bs$

$\text{map} :: (a \rightarrow b) \rightarrow [a] \rightarrow [b]$
 $\text{map } f [] = []$
 $\text{map } f (x:xs) = f x : \text{map } f xs$

Case: $bs = (x:xs)$

Assume: $\text{map } g (\text{map } f xs) = \text{map } (g . f) xs$

$$\text{map } g (\text{map } f (x:xs)) = (g . f) x : \text{map } (g . f) xs$$

Functor - Composition

Goal : $\forall bs :: [a], f :: a \rightarrow b, g :: b \rightarrow c$
 $\text{map } g (\text{map } f bs) = \text{map } (g \cdot f) bs$

$\text{map} :: (a \rightarrow b) \rightarrow [a] \rightarrow [b]$
 $\text{map } f [] = []$
 $\text{map } f (x:xs) = f x : \text{map } f xs$

Case: $bs = (x:xs)$

Assume: $\text{map } g (\text{map } f xs) = \text{map } (g \cdot f) xs$

$$\text{map } g (\text{map } f (x:xs)) = (g \cdot f) x : \text{map } (g \cdot f) xs$$

Functor - Composition

Goal : $\forall bs :: [a], f :: a \rightarrow b, g :: b \rightarrow c$
 $\text{map } g (\text{map } f bs) = \text{map } (g \cdot f) bs$

$\text{map} :: (a \rightarrow b) \rightarrow [a] \rightarrow [b]$
 $\text{map } f [] = []$
 $\text{map } f (x:xs) = f x : \text{map } f xs$

Case: $bs = (x:xs)$

Assume: $\text{map } g (\text{map } f xs) = \text{map } (g \cdot f) xs$

$$\text{map } g (f x : \text{map } f xs) = (g \cdot f) x : \text{map } (g \cdot f) xs$$

Functor - Composition

Goal : $\forall bs :: [a], f :: a \rightarrow b, g :: b \rightarrow c$
 $\text{map } g (\text{map } f bs) = \text{map } (g \cdot f) bs$

$\text{map} :: (a \rightarrow b) \rightarrow [a] \rightarrow [b]$
 $\text{map } f [] = []$
 $\text{map } f (x:xs) = f x : \text{map } f xs$

Case: $bs = (x:xs)$

Assume: $\text{map } g (\text{map } f xs) = \text{map } (g \cdot f) xs$

$\text{map } g (f x : \text{map } f xs) = (g \cdot f) x : \text{map } (g \cdot f) xs$

Functor - Composition

Goal : $\forall bs :: [a], f :: a \rightarrow b, g :: b \rightarrow c$
 $\text{map } g (\text{map } f bs) = \text{map } (g \cdot f) bs$

$\text{map} :: (a \rightarrow b) \rightarrow [a] \rightarrow [b]$
 $\text{map } f [] = []$
 $\text{map } f (x:xs) = f x : \text{map } f xs$

Case: $bs = (x:xs)$

Assume: $\text{map } g (\text{map } f xs) = \text{map } (g \cdot f) xs$

$$\text{map } g (f x : \text{map } f xs) = (g \cdot f) x : \text{map } (g \cdot f) xs$$

Functor - Composition

Goal : $\forall bs :: [a], f :: a \rightarrow b, g :: b \rightarrow c$
 $\text{map } g (\text{map } f bs) = \text{map } (g \cdot f) bs$

$\text{map} :: (a \rightarrow b) \rightarrow [a] \rightarrow [b]$

$\text{map } f [] = []$

$\text{map } f (x:xs) = f x : \text{map } f xs$

Case: $bs = (x:xs)$

Assume: $\text{map } g (\text{map } f xs) = \text{map } (g \cdot f) xs$

$\text{map } g (f x : \text{map } f xs) = (g \cdot f) x : \text{map } (g \cdot f) xs$

Functor - Composition

Goal : $\forall bs :: [a], f :: a \rightarrow b, g :: b \rightarrow c$
 $\text{map } g (\text{map } f bs) = \text{map } (g \cdot f) bs$

$\text{map} :: (a \rightarrow b) \rightarrow [a] \rightarrow [b]$
 $\text{map } f [] = []$
 $\text{map } f (x:xs) = f x : \text{map } f xs$

Case: $bs = (x:xs)$

Assume: $\text{map } g (\text{map } f xs) = \text{map } (g \cdot f) xs$

$g (f x) : \text{map } g (\text{map } f xs) = (g \cdot f) x : \text{map } (g \cdot f) xs$

Functor - Composition

Goal : $\forall bs :: [a], f :: a \rightarrow b, g :: b \rightarrow c$
 $\text{map } g (\text{map } f bs) = \text{map } (g \circ f) bs$

$\text{map} :: (a \rightarrow b) \rightarrow [a] \rightarrow [b]$
 $\text{map } f [] = []$
 $\text{map } f (x:xs) = f x : \text{map } f xs$

Case: $bs = (x:xs)$

Assume: $\text{map } g (\text{map } f xs) = \text{map } (g \circ f) xs$

$g (f x) : \text{map } g (\text{map } f xs) = (g \circ f) x : \text{map } (g \circ f) xs$

Functor - Composition

Goal : $\forall bs :: [a], f :: a \rightarrow b, g :: b \rightarrow c$
map g (map f bs) = map (g . f) bs

map :: (a -> b) -> [a] -> [b]
map f [] = []
map f (x:xs) = f x : map f xs

Case: **bs = (x:xs)**

Assume: map g (map f xs) = map (g . f) xs

(g . f) x : map g (map f xs) = (g . f) x : map (g . f) xs

Functor - Composition

Goal : $\forall bs :: [a], f :: a \rightarrow b, g :: b \rightarrow c$
 $\text{map } g (\text{map } f bs) = \text{map } (g . f) bs$

$\text{map} :: (a \rightarrow b) \rightarrow [a] \rightarrow [b]$
 $\text{map } f [] = []$
 $\text{map } f (x:xs) = f x : \text{map } f xs$

Case: $bs = (x:xs)$

Assume: $\text{map } g (\text{map } f xs) = \text{map } (g . f) xs$

$(g . f) x : \text{map } g (\text{map } f xs) = (g . f) x : \text{map } (g . f) xs$

Functor - Composition

Goal : $\forall bs :: [a], f :: a \rightarrow b, g :: b \rightarrow c$
 $\text{map } g (\text{map } f bs) = \text{map } (g . f) bs$

$\text{map} :: (a \rightarrow b) \rightarrow [a] \rightarrow [b]$
 $\text{map } f [] = []$
 $\text{map } f (x:xs) = f x : \text{map } f xs$

Case: $bs = (x:xs)$

Assume: $\text{map } g (\text{map } f xs) = \text{map } (g . f) xs$

$(g . f) x : \text{map } g (\text{map } f xs) = (g . f) x : \text{map } (g . f) xs$

Functor - Composition

Goal : $\forall bs :: [a], f :: a \rightarrow b, g :: b \rightarrow c$
 $\text{map } g (\text{map } f bs) = \text{map } (g . f) bs$

$\text{map} :: (a \rightarrow b) \rightarrow [a] \rightarrow [b]$
 $\text{map } f [] = []$
 $\text{map } f (x:xs) = f x : \text{map } f xs$

Case: $bs = (x:xs)$

Assume: $\text{map } g (\text{map } f xs) = \text{map } (g . f) xs$

$(g . f) x : \text{map } g (\text{map } f xs) = (g . f) x : \text{map } (g . f) xs$

Functor - Composition

Goal : $\forall bs :: [a], f :: a \rightarrow b, g :: b \rightarrow c$
 $\text{map } g (\text{map } f bs) = \text{map } (g \cdot f) bs$

$\text{map} :: (a \rightarrow b) \rightarrow [a] \rightarrow [b]$
 $\text{map } f [] = []$
 $\text{map } f (x:xs) = f x : \text{map } f xs$

Case: $bs = (x:xs)$

Assume: $\text{map } g (\text{map } f xs) = \text{map } (g \cdot f) xs$

$(g \cdot f) x : \text{map } (g \cdot f) xs = (g \cdot f) x : \text{map } (g \cdot f) xs$

Functor - Composition

Goal : $\forall bs :: [a], f :: a \rightarrow b, g :: b \rightarrow c$
 $\text{map } g (\text{map } f bs) = \text{map } (g . f) bs$

$\text{map} :: (a \rightarrow b) \rightarrow [a] \rightarrow [b]$
 $\text{map } f [] = []$
 $\text{map } f (x:xs) = f x : \text{map } f xs$

Case: $bs = (x:xs)$

Assume: $\text{map } g (\text{map } f xs) = \text{map } (g . f) xs$

$(g . f) x : \text{map } (g . f) xs = (g . f) x : \text{map } (g . f) xs$

Functor - Composition

Goal : $\forall bs :: [a], f :: a \rightarrow b, g :: b \rightarrow c$
 $\text{map } g (\text{map } f bs) = \text{map } (g . f) bs$

$\text{map} :: (a \rightarrow b) \rightarrow [a] \rightarrow [b]$
 $\text{map } f [] = []$
 $\text{map } f (x:xs) = f x : \text{map } f xs$

Case: $bs = (x:xs) \checkmark$

Assume: $\text{map } g (\text{map } f xs) = \text{map } (g . f) xs$

$(g . f) x : \text{map } (g . f) xs = (g . f) x : \text{map } (g . f) xs$

Coq

To the Coqmobile!

Coq extraction - append

```
module Append where

import qualified Prelude

data List a =
  Nil
  | Cons a (List a)

append :: (List a1) -> (List a1) -> List a1
append xs ys =
  case xs of {
    Nil -> ys;
    Cons h t -> Cons h (append t ys)}
```

Coq extraction - map

```
module Map where

import qualified Prelude

data List a =
  Nil
  | Cons a (List a)

map :: (a1 -> a2) -> (List a1) -> List a2
map f l =
  case l of {
    Nil -> Nil;
    Cons a t -> Cons (f a) (map f t)}
```

Conclusion

- Equational reasoning is easy and useful
- Theorem provers can help if you want to be really sure
- Very few excuses for law-breaking typeclass instances

Other cool stuff

- Bottom, partial data and strictness
- Monad laws
 - Similar to the monoid laws
 - Just a monoid in the category of endofunctors after all
- Fold fusion
 - when does $f \cdot \text{fold } g \ a = \text{fold } h \ b$?
- Program synthesis
- Theorems for free

Books

- How To Prove It - Velleman
- Introduction to Functional Programming with Haskell - Bird
- Interactive Theorem Proving and Program Development - Bertot
- Software Foundations - Pierce

Slides and code

<https://github.com/dalaing/bfpg-2013-03>