Required text and reading

Haskell: The Craft of Functional Programming (Third Edition),
Simon Thompson, Addison-Wesley, 2011.

Reading assignment

Monday 24 September 2012  Chapters 1–3 (pp. 1–66)
Monday 1 October 2012    Chapters 4–7 (pp. 67–176)
Monday 8 October 2012    Chapters 8–9 (pp. 177–212)
Monday 15 October 2012   Chapters 10–12 (pp. 213–286)
Monday 22 October 2012   Class test
Monday 29 October 2012   Chapters 13–14 (pp. 287–356)
Monday 5 November 2012   Chapters 15–16 (pp. 357–414)
Monday 12 November 2012  Chapters 17–21 (pp. 415–534)
Part I

Lambda expressions
A failed attempt to simplify

\[
f :: \text{[Int]} \rightarrow \text{Int} \\
f \text{xs} = \text{foldr} \ (+) \ 0 \ (\text{map} \ \text{sqr} \ (\text{filter} \ \text{pos} \ \text{xs})) \\
\text{where} \\
\text{sqr} \ x = x \times x \\
\text{pos} \ x = x > 0
\]

The above \textit{cannot} be simplified to the following:

\[
f :: \text{[Int]} \rightarrow \text{Int} \\
f \text{xs} = \text{foldr} \ (+) \ 0 \ (\text{map} \ (x \times x) \ (\text{filter} \ (x > 0) \ \text{xs}))
\]
A successful attempt to simplify

\[
\begin{align*}
f :: [\text{Int}] & \rightarrow \text{Int} \\
f \; \text{xs} & = \; \text{foldr} \; (+) \; 0 \; (\text{map} \; \text{sqr} \; (\text{filter} \; \text{pos} \; \text{xs})) \\
& \quad \text{where} \\
& \quad \text{sqr} \; x \; = \; x \; \times \; x \\
& \quad \text{pos} \; x \; = \; x \; > \; 0
\end{align*}
\]

The above \textit{can} be simplified to the following:

\[
\begin{align*}
f :: [\text{Int}] & \rightarrow \text{Int} \\
f \; \text{xs} & = \; \text{foldr} \; (+) \; 0 \\
& \quad (\text{map} \; (\lambda x \rightarrow x \; \times \; x) \\
& \quad \; (\text{filter} \; (\lambda x \rightarrow x \; > \; 0) \; \text{xs}))
\end{align*}
\]
Lambda calculus

\[ f :: [\text{Int}] \rightarrow \text{Int} \]
\[ f \text{ xs} = \text{foldr} \ (+) \ 0 \]
\[ (\text{map} \ (\lambda x \rightarrow x \times x)) \]
\[ (\text{filter} \ (\lambda x \rightarrow x > 0) \ \text{xs}) \]

The character \( \backslash \) stands for \( \lambda \), the Greek letter lambda.

Logicians write

\[ \backslash x \rightarrow x > 0 \quad \text{as} \quad \lambda x. \ x > 0 \]

\[ \backslash x \rightarrow x \times x \quad \text{as} \quad \lambda x. \ x \times x. \]

Lambda calculus is due to the logician Alonzo Church (1903–1995).
Evaluating lambda expressions

\((\lambda x \to x > 0) \, 3\)

=  
  let \(x = 3\) in \(x > 0\)

=  
  3 > 0

=  
  True

\((\lambda x \to x * x) \, 3\)

=  
  let \(x = 3\) in \(x \times x\)

=  
  3 \times 3

=  
  9
Lambda expressions and currying

\((\lambda x \rightarrow \lambda y \rightarrow x + y)\) 3 4

= 

\(((\lambda x \rightarrow (\lambda y \rightarrow x + y))\) 3) 4

= 

(let x = 3 in \(\lambda y \rightarrow x + y\)) 4

= 

(\(\lambda y \rightarrow 3 + y\)) 4

= 

(let y = 4 in 3 + y)

= 

3 + 4

= 

7
Evaluating lambda expressions

The general rule for evaluating lambda expressions is

\[(\lambda x. N) \ M\]

\[=\]

\[(\text{let } x = M \text{ in } N)\]

This is sometimes called the \(\beta\) rule (or beta rule).
Part II

Sections
Sections

 (> 0) is shorthand for (\x -> x > 0)

 (2 *) is shorthand for (\x -> 2 * x)

 (+ 1) is shorthand for (\x -> x + 1)

 (2 ^) is shorthand for (\x -> 2 ^ x)

 (^ 2) is shorthand for (\x -> x ^ 2)
Sections

\[ f :: [\text{Int}] \rightarrow \text{Int} \]
\[ f \ xs \ = \ \text{foldr} \ (+) \ 0 \]
\[ \quad (\text{map} \ (\lambda x \rightarrow x \times x) \]
\[ \quad \quad \quad (\text{filter} \ (\lambda x \rightarrow x > 0) \ xs)) \]

\[ f :: [\text{Int}] \rightarrow \text{Int} \]
\[ f \ xs \ = \ \text{foldr} \ (+) \ 0 \ (\text{map} \ (^2) \ (\text{filter} \ (> \ 0) \ xs)) \]
Composition

\[(.) :: (b \to c) \to (a \to b) \to (a \to c)\]

\[(f \ . \ g) \ x \ = \ f \ (g \ x)\]
Evaluation composition

\( (\cdot) :: (b \to c) \to (a \to b) \to (a \to c) \)
\( (f \cdot g) \; x = f \; (g \; x) \)

\[
\begin{align*}
\text{sqr} :: \text{Int} \to \text{Int} \\
\text{sqr} \; x &= x \times x \\
\text{pos} :: \text{Int} \to \text{Bool} \\
\text{pos} \; x &= x > 0
\end{align*}
\]

\[
\begin{align*}
(p\text{os} \cdot \text{sqr}) \; 3
&= \\
\text{pos} \; (\text{sqr} \; 3)
&= \\
\text{pos} \; 9
&= \\
\text{True}
\end{align*}
\]
Compare and contrast

possqr :: Int -> Bool
possqr x = pos (sqr x)

possqr 3
= 
pos (sqr 3)
= 
pos 9
= True

possqr :: Int -> Bool
possqr = pos . sqr

possqr 3
= 
(pos . sqr) 3
= 
pos (sqr 3)
= 
pos 9
= True
Composition is associative

\[(f . g) . h = f . (g . h)\]

\[\begin{align*}
((f . g) . h) \ x &= \\
(f . g) (h \ x) &= \\
f (g (h \ x)) &= \\
f ((g . h) x) &= \\
(f . (g . h)) \ x &= 
\end{align*}\]
Thinking functionally

\[ f :: [\text{Int}] \rightarrow \text{Int} \]
\[ f \; xs \; = \; \text{foldr} \; (+) \; 0 \; (\text{map} \; (^\; 2) \; (\text{filter} \; (> \; 0) \; xs)) \]

\[ f :: [\text{Int}] \rightarrow \text{Int} \]
\[ f \; = \; \text{foldr} \; (+) \; 0 \; \circ \; \text{map} \; (^\; 2) \; \circ \; \text{filter} \; (> \; 0) \]
Applying the function

\[ f :: \text{[Int]} \rightarrow \text{Int} \]
\[ f = \text{foldr} (+) 0 . \text{map} (\wedge 2) . \text{filter} (> 0) \]

\[ f [1, -2, 3] = \]
\[ (\text{foldr} (+) 0 . \text{map} (\wedge 2) . \text{filter} (> 0)) [1, -2, 3] = \]
\[ \text{foldr} (+) 0 (\text{map} (\wedge 2) (\text{filter} (> 0) [1, -2, 3])) = \]
\[ \text{foldr} (+) 0 (\text{map} (\wedge 2) [1, 3]) = \]
\[ \text{foldr} (+) 0 [1, 9] = \]
\[ 10 \]
Part IV

Variables and binding
Variables

\[
x = 2 \\
y = x+1 \\
z = x+y*y
\]

*Main> z
11
Variables—binding

\[ x = 2 \]
\[ y = x+1 \]
\[ z = x+y*y \]

*Main> z
11

**Binding occurrence**

**Bound occurrence**

Scope of binding
Variables—binding

\[
\begin{align*}
  x &= 2 \\
  y &= x + 1 \\
  z &= x + y \times y
\end{align*}
\]

*Main> z
11

**Binding occurrence**

**Bound occurrence**

Scope of binding
Variables—binding

\[
\begin{align*}
x & = 2 \\
y & = x + 1 \\
z & = x + y \times y
\end{align*}
\]

*Main> z
11

**Binding occurrence**

**Bound occurrence**

Scope of binding
Variables—renaming

xavier = 2
yolanda = xavier+1
zeuss = xavier+yolanda*yolanda

*Main> zeuss
11
Part V

Functions and binding
Functions—binding

\[
  f \, x = g \, x \, (x+1) \\
  g \, x \, y = x+y*y
\]

*Main> f 2
11
Functions—binding

\[ f \ x = g \ x \ (x+1) \]
\[ g \ x \ y = x+y*y \]

*Main> f 2
11

**Binding occurrence**

*Bound occurrence*

Scope of binding
Functions—binding

\[ f \ x = g \ x \ (x+1) \]
\[ g \ x \ y = x+y*y \]

*Main> f 2
11

**Binding occurrence**

*Bound occurrence*

Scope of binding

There are two *unrelated* uses of \( x \)!
Functions—binding

\[ f \ x = g \ x \ (x+1) \]
\[ g \ x \ y = x+y*y \]

*Main> f 2
11

**Binding occurrence**

**Bound occurrence**

Scope of binding
Functions—binding

\[ f \ x = g \ x \ (x+1) \]
\[ g \ x \ y = x+y\cdot y \]

*Main> \ f \ 2
11

**Binding occurrence**

*Bound occurrence*

Scope of binding
Functions—binding

\[ f \ x = g \ x \ (x+1) \]
\[ g \ x \ y = x+y*y \]

*Main> f 2
11

**Binding occurrence**
*Bound occurrence*
Scope of binding
Functions—formal and actual parameters

\[
\begin{align*}
    f \ x &= g \ x \ (x+1) \\
    g \ x \ y &= x+y\times y
\end{align*}
\]

*Main> f 2
11

**Formal parameter**

**Actual parameter**
Functions—formal and actual parameters

\[ f \ x = g \ x \ (x+1) \]
\[ g \ x \ y = x+y \times y \]

*Main> f 2
11

Formal parameter
Actual parameter
Functions—formal and actual parameters

\[ f \ x = g \ x \ (x+1) \]
\[ g \ x \ y = x+y*y \]

*Main> f 2
11

Formal parameter
Actual parameter
Functions—renaming

Fred Xavier = George Xavier (xavier+1)
George Xerox Yolanda = Xerox+Yolanda*Yolanda

*Main> fred 2
11

Different uses of x renamed to xavier and xerox.
Part VI

Variables in a where clause and binding
Variables in a where clause

\[ f \ x = z \]
\[ \text{where} \]
\[ y = x + 1 \]
\[ z = x + y * y \]

*Main> f 2
11
Variables in a where clause—binding

\[
f \ x \ = \ z
\]
\[
\text{where}
\]
\[
y \ = \ x + 1
\]
\[
z \ = \ x + y \times y
\]

*Main> f 2
11

**Binding occurrence**

**Bound occurrence**

Scope of binding
Variables in a where clause—binding

\[
\begin{align*}
f \ x &= z \\
\text{where} \\
\ y &= x+1 \\
\ z &= x+y*y
\end{align*}
\]

*Main> f 2
11

**Binding occurrence**

*Bound occurrence*

Scope of binding
Variables in a where clause—binding

\[
f \ x = \ z \\
\quad \text{where} \\
\quad \ y = \ x + 1 \\
\quad \ z = \ x + y \times y
\]

*Main> \ f \ 2
11

**Binding occurrence**

*Bound occurrence*

Scope of binding
Variables in a where clause—binding

```haskell
f x = z
  where
    y = x+1
    z = x+y*y

*Main> f 2
11
```

**Binding occurrence**

**Bound occurrence**

Scope of binding
Variables in a where clause—hole in scope

```haskell
f x = z
   where
     y = x + 1
     z = x + y * y

y = 5

*Main> y
5
```

**Binding occurrence**

**Bound occurrence**

Scope of binding
Part VII

Functions in a where clause and binding
Functions in a where clause

\[ f \ x = g \ (x+1) \]
\[ \text{where} \]
\[ g \ y = x+y \times y \]

*Main> f 2
11
Functions in a where clause—binding

\[
f \ x = g (x + 1) \\
\text{where} \\
g \ y = x + y \times y
\]

\*Main> f 2
11

**Binding occurrence**

**Bound occurrence**

Scope of binding

Variable \( x \) is still in scope within \( g \)!
Functions in a where clause—binding

\[ f \ x = g \ (x+1) \]
\[
\text{where}
\]
\[ g \ y = x + y \cdot y \]

*Main> f 2
11

**Binding occurrence**

**Bound occurrence**

Scope of binding
Functions in a where clause—binding

```haskell
f x = g (x+1)
    where
    g y = x+y*y
```

*Main> f 2
11

**Binding occurrence**

**Bound occurrence**

Scope of binding
Functions in a where clause—binding

\[
f \ x = \ g \ (x+1)  
\]
where
\[
g \ y = x+y*y  
\]

*Main> f 2
11

**Binding occurrence**

*Bound occurrence*

Scope of binding
Functions in a where clause—hole in scope

\[
\begin{align*}
f(x) &= g(x+1) \\
    \text{where} & \\
    g(y) &= x+y*y \\
    g(z) &= z*z*z \\
\end{align*}
\]

*Main> g 2
8

**Binding occurrence**

**Bound occurrence**

Scope of binding
Functions in a where clause—pathological case

\[
f \ x = f \ (x+1) \\
\quad \text{where} \\
\quad f \ y = x + y \cdot y
\]

\[
* \text{Main>} \ f \ 2 \\
11
\]

**Binding occurrence**

**Bound occurrence**

Scope of binding
Functions in a where clause—pathological case

\[
\begin{align*}
\textbf{f} & \quad x = f (x+1) \\
\text{where} & \\
\textbf{f} & \quad y = x+y*y
\end{align*}
\]

*Main> \textbf{f} 2
11

**Binding occurrence**

*Bound occurrence*

Scope of binding
Functions in a where clause—formals and actuals

\[
\begin{align*}
  f \; x &= g \; (x+1) \\
  \quad \text{where} \\
  g \; y &= x+y\times y
\end{align*}
\]

*Main> f 2
11

**Formal parameter**

**Actual parameter**
Functions in a where clause—formals and actuals

\[
f \ x = g \ (x+1) \\
\text{where} \\
g \ y = x+y*y
\]

*Main> f 2
11

Formal parameter
Actual parameter
Part VIII

Lambda expressions and binding
A wrong attempt to simplify

\[
\begin{align*}
  f & \colon [\text{Int}] \to [\text{Int}] \\
  f \, \text{xs} & = \text{map} \left( x \times x \right) \left( \text{filter} \left( x > 0 \right) \text{xs} \right)
\end{align*}
\]

This makes no sense—no binding occurrence of variable!
Lambda expressions

\[ f :: [\text{Int}] \rightarrow [\text{Int}] \]
\[ f \ x s = \]
\[ \text{map} \ (\lambda x \rightarrow x \times x) \ (\text{filter} \ (\lambda x \rightarrow x > 0) \ x s) \]

The character \( \lambda \) stands for \( \lambda \), the Greek letter *lambda*.

Logicians write

\[ (\lambda x \rightarrow x \times x) \quad \text{as} \quad (\lambda x. \ x \times x) \]
\[ (\lambda x \rightarrow x > 0) \quad \text{as} \quad (\lambda x. \ x > 0) \]
Lambda expressions—binding

\[ f :: \textbf{[Int]} \rightarrow \textbf{[Int]} \]
\[ f \; \textbf{xs} \; = \; \textbf{map} \; (\backslash x \rightarrow x \times x) \; (\textbf{filter} \; (\backslash x \rightarrow x > 0) \; \textbf{xs}) \]

**Binding occurrence**

*Bound occurrence*

Scope of binding
Lambda expressions—binding

\[ f :: [Int] \rightarrow [Int] \]
\[ f \; xs \; = \; \text{map } (\lambda \; x \rightarrow \; x \times x) \; (\text{filter } (\lambda \; x \rightarrow \; x > 0) \; xs) \]

**Binding occurrence**

*Bound occurrence*

Scope of binding
Part IX

Lambda expressions explain binding
Lambda expressions explain binding

A variable binding can be rewritten using a lambda expression and an application:

\[(N \text{ where } x = M)\]
\[=\]
\[(\lambda x. N) \ M\]
\[=\]
\[(\text{let } x = M \text{ in } N)\]

A function binding can be written using an application on the left or a lambda expression on the right:

\[(M \text{ where } f \ x = N)\]
\[=\]
\[(M \text{ where } f = \lambda x. N)\]
Lambda expressions and binding constructs

\[
\begin{align*}
f & \ 2 \\
\text{where} \quad f & \ x \ = \ x + y \cdot y \\
\text{where} \quad y & \ = \ x + 1 \\
& = \\
\text{f} & \ 2 \\
\text{where} \quad f & \ = \ \lambda x \ . \ (x + y \cdot y \text{ where } y \ = \ x + 1) \\
& = \\
\text{f} & \ 2 \\
\text{where} \quad f & \ = \ \lambda x \ . \ ((\lambda y \ . \ x + y \cdot y) \ (x + 1)) \\
& = \\
(\lambda f \ . \ f \ 2) & \ (\lambda x \ . \ ((\lambda y \ . \ x + y \cdot y) \ (x + 1)))
\end{align*}
\]
Evaluating lambda expressions

\((\lambda f \rightarrow f \: 2) \: (\lambda x \rightarrow ((\lambda y \rightarrow x+y \cdot y) \: (x+1)))\)  
=  
\((\lambda x \rightarrow ((\lambda y \rightarrow x+y \cdot y) \: (x+1))) \: 2\)  
=  
\((\lambda y \rightarrow 2+y \cdot y) \: (2+1)\)  
=  
\((\lambda y \rightarrow 2+y \cdot y) \: 3\)  
=  
\(2+3 \cdot 3\)  
=  
11
Part X

Additional material:
Lambda expressions and binding, revisited
Lambda expressions—binding

$$(\lambda f \rightarrow f \ 2) \ (\lambda x \rightarrow ((\lambda y \rightarrow x+y*y) \ (x+1)))$$

**Binding occurrence**

**Bound occurrence**

Scope of binding
Lambda expressions—binding

\((\ f \ \to \ f \ 2\) \ (\ x \ \to \ ((\ y \ \to \ x\!+\!y\!\times\!y\) \ (\ x\!+\!1\))))\)

Binding occurrence

Bound occurrence

Scope of binding
Lambda expressions—binding

\((\lambda f \rightarrow f \ 2) \ (\lambda x \rightarrow ((\lambda y \rightarrow x + y \times y) \ (x + 1)))\)

**Binding occurrence**

**Bound occurrence**

Scope of binding
Lambda expressions—formals and actuals

$(\lambda f \to f \ 2) \ (\lambda x \to ((\lambda y \to x+y\times y) \ (x+1)))$

**Formal parameter**

**Actual parameter**
Lambda expressions—formals and actuals

\[
(\lambda x \rightarrow (\lambda y \rightarrow x+y \times y) (x+1))
\]

Formal parameter

Actual parameter
Lambda expressions—formals and actuals

(\y \to 2+y*y) (2+1)

Formal parameter
Actual parameter
Part XI

Additional material: Comprehensions and binding
Comprehensions

\[ f :: [\text{Int}] \rightarrow [\text{Int}] \]
\[ f \text{ xs} = [ x^2 | x \leftarrow \text{xs}, x > 0 ] \]

*Main> f [1, -2, 3]
[1, 9]
Comprehensions—binding

\[ f :: [\text{Int}] \rightarrow [\text{Int}] \]
\[ f \, \text{xs} = [ x \times x \mid x \leftarrow \text{xs}, x > 0 ] \]

*Main> f [1,-2,3]
[1,9]

**Binding occurrence**

**Bound occurrence**

Scope of binding
Comprehensions—binding

\[ f :: [\text{Int}] \rightarrow [\text{Int}] \]
\[ f \text{ xs } = \left[ x^2 \mid x \leftarrow \text{xs}, x > 0 \right] \]

*Main> f [1,-2,3]
[1,9]

**Binding occurrence**

*Bound occurrence*

Scope of binding
Comprehensions—pathological case

\[
f :: [\text{Int}] \rightarrow [\text{Int}]
f x = [ x^2 | x <- x, x > 0 ]
\]

*Main> f [1,-2,3]
[1,9]

**Binding occurrence**

**Bound occurrence**

Scope of binding – Note hole in scope!
Squares of Positives—pathological case

\[ f :: [\text{Int}] \rightarrow [\text{Int}] \]
\[ f \ x = [ \ x \times x \mid x \leftarrow x, \ x > 0 ] \]

*Main> f [1,-2,3]
[1,9]

**Binding occurrence**

**Bound occurrence**

Scope of binding
List comprehension with two qualifiers

\[ f \ n \ = \ [ \ (i, j) \mid i \gets [1..n], \ j \gets [i..n] ] \]

*Main> \ f 3
[(1, 1), (1, 2), (1, 3), (2, 2), (2, 3), (3, 3)]
List comprehension with two qualifiers—binding

\[ f \ n = [ (i, j) \mid i \leftarrow [1..n], j \leftarrow [i..n] ] \]

*Main> \ f \ 3
[(1,1), (1,2), (1,3), (2,2), (2,3), (3,3)]

**Binding occurrence**

**Bound occurrence**

Scope of binding
List comprehension with two qualifiers—binding

\[ f \ n \ = \ \left[ (i,j) \mid i \ <\ [1..n], \ j \ <\ [i..n] \right] \]

*Main> f 3
[(1,1),(1,2),(1,3),(2,2),(2,3),(3,3)]

**Binding occurrence**

*Bound occurrence*

Scope of binding
List comprehension with two qualifiers—binding

\[ f \ n \ = \ [ \ (i, j) \mid i \ <- \ [1..n], \ j \ <- \ [i..n] \] \]

*Main> f 3
[(1,1),(1,2),(1,3),(2,2),(2,3),(3,3)]

**Binding occurrence**

**Bound occurrence**

Scope of binding
Part XII

Additional material:
Higher-order functions and binding
Higher-order functions

\[
f :: [\text{Int}] \to [\text{Int}]
f \text{xs} = \text{map} \ (\text{sqr} \ \text{filter pos} \ \text{xs})
\]
where
\[
\text{sqr} \ x = x \times x
\text{pos} \ x = x > 0
\]

*Main> f [1,-2,3]
[1,9]
Higher order functions—binding

\[ f \] \[ xs = \text{map} \ \text{sqr} \ (\text{filter} \ \text{pos} \ \text{xs}) \]
where
\[ \text{sqr} \ x = x \times x \]
\[ \text{pos} \ x = x > 0 \]

*Main> \[ f [1,-2,3] \]
\[ [1,9] \]

**Binding occurrence**

**Bound occurrence**

Scope of binding
Higher-order functions—binding

\[
\text{f } \mathbf{xs} \; = \; \text{map } \text{sqr} \; (\text{filter } \text{pos} \; \mathbf{xs})
\]

where

\[
\text{sqr } x \; = \; x \times x
\]

\[
\text{pos } x \; = \; x > 0
\]

*Main> f [1,-2,3]
[1,9]

**Binding occurrence**

*Bound occurrence*

Scope of binding
Higher-order functions—binding

\[
f \, xs = \text{map} \, \text{sqr} \, (\text{filter} \, \text{pos} \, xs)
\]
where

\[
\text{sqr} \, x = x \times x
\]
\[
\text{pos} \, x = x > 0
\]

*Main> f [1, -2, 3]
[1, 9]

**Binding occurrence**

**Bound occurrence**

Scope of binding
Higher-order functions—binding

\[ f \; \text{x}s \; = \; \text{map} \; \text{sqr} \; (\text{filter} \; \text{pos} \; \text{x}s) \]
\[ \text{where} \]
\[ \text{sqr} \; \text{x} \; = \; \text{x} * \text{x} \]
\[ \text{pos} \; \text{x} \; = \; \text{x} > 0 \]

*Main> f [1,-2,3]
[1,9]

**Binding occurrence**

**Bound occurrence**

Scope of binding
Higher-order functions—binding

\[
f \text{xs} = \text{map } \text{sqr} \ (\text{filter } \text{pos } \text{xs})
\]

where

\[
\text{sqr } x = x \times x
\]
\[
\text{pos } x = x > 0
\]

*Main> f [1,-2,3]
[1,9]

**Binding occurrence**

**Bound occurrence**

Scope of binding
Higher-order functions—binding

\[
f \text{xs} = \text{map } \text{sqr} \ (\text{filter } \text{pos} \ \text{xs})
\]

where

\[
\text{sqr } x = x \times x \\
\text{pos } x = x > 0
\]

*Main> f [1,-2,3]
[1,9]

**Binding occurrence**

**Bound occurrence**

Scope of binding
Higher-order functions—binding

```
f xs = map sqr (filter pos xs)
    where
      sqr x = x * x
      pos x = x > 0

*Main> f [1,-2,3]
[1,9]
```

**Binding occurrence**—not shown (in standard prelude)

**Bound occurrence**

Scope of binding
Higher-order functions—binding

\[ f \; xs = \; \text{map} \; \text{sqr} \; (\text{filter} \; \text{pos} \; xs) \]
where
\[ \text{sqr} \; x = \; x^2 \]
\[ \text{pos} \; x = \; x > 0 \]

*Main> \; f \; [1,-2,3]
[1,9]

**Binding occurrence**—not shown (in standard prelude)

**Bound occurrence**

Scope of binding