CS 457/557 Functional Programming

Lecture 11
Proving Program Properties
Recall the calculation proof method

• Substitution of equals for equals.
• Based on definitions or previously proved theorems.

• For example consider:
  \[(f \circ g) x = f(g x)\] \hspace{1cm} (comp)

  – Notice label on equation

• Now prove that composition is associative, i.e.
  \[((f \circ g) \circ h) x = (f \circ (g \circ h)) x\]

  – Can use known equations in either direction.
Example: Proof by calculation

• Pick one side of the equation and transform using rule \textit{comp} above

\[(f \cdot g) \cdot h =\]
\hspace{10mm} \text{by comp (left to right)}

\[(f \cdot g) (h x) =\]
\hspace{10mm} \text{by comp (left to right)}

\[f (g (h x)) =\]
\hspace{10mm} \text{by comp (right to left)}

\[f ((g \cdot h) x) =\]
\hspace{10mm} \text{by comp (right to left)}

\[(f \cdot (g \cdot h)) x\]
Example With Regions

• Consider the algebra of Shapes (Ch. 8)
• Suppose we have already proved (Hudak p.100-101):
  
  \[ r \ ` \text{Union}` \text{ Empty} = r \]  
  \[ \text{Axiom 4a} \]
  
  \[ r \ ` \text{Intersect}` \text{ univ} = r \]  
  \[ \text{Axiom 4b} \]
  
  \[ r \ ` \text{Union}` \text{ Complement} r = \text{univ} \]  
  \[ \text{Axiom 5a} \]
  
  \[ r \ ` \text{Intersect}` \text{ Complement} r = \text{Empty} \]  
  \[ \text{Axiom 5b} \]
  
  \[ r1 \ ` \text{Union}` (r2 ` \text{Intersect}` r3) = (r1 ` \text{Intersect}` r2) ` \text{Union}` (r1 ` \text{Intersect}` r3) \]  
  \[ \text{Axiom 3b} \]

• Prove: \[ r \ ` \text{Union}` r = r \]
  
  \[ r = (\text{by Axiom 4a}) \]
  
  \[ r \ ` \text{Union}` \text{ Empty} = (\text{by 5b}) \]
  
  \[ r \ ` \text{Union}` (r ` \text{Intersect}` \text{ Complement} r) = (\text{by 3b}) \]
  
  \[ (r ` \text{Union} r) ` \text{Intersect} (r ` \text{Union} \text{ Complement} r) = (r ` \text{Union} r) ` \text{Intersect} \text{ univ} = (\text{by 4b}) \]  
  \[ (\text{by 5a})^\wedge \]
  
  \[ r ` \text{Union} r \]
Proofs by induction over finite lists

- Format over lists
  Let P{x} be some proposition (I.e. P{x} :: Bool)
  i.e. P is an expression with some free variable x :: [a]
    - x has type :: [a]
    - x may occur more than once in P{x}

  e.g.
  \[ \text{length } x = \text{length (reverse } x) \]
  \[ \text{all } p \ x \Rightarrow p \ (\text{head } x) \]
  \[ \text{sum (} x + + y) = \text{sum } x + \text{sum } y \]
  \[ \text{map } f \ (x + + y) = \text{map } f \ x + + \text{map } f \ y \]
  \[ (\text{map } f . \text{map } g) \ x = \text{map } (f . g) \ x \]

- Then to prove P for all finite lists, we:
  1) Prove P { [] }
  2) Assume P{ xs } and then
     Prove P{ x:xs }
Example: relating map and length

- **Definitions and Laws:** (These are things we get to assume are true)
  \[
  \text{length} \; [\;] = 0 \tag{1} \\
  \text{length} \; (x:xs) = 1 + \text{length} \; xs \tag{2} \\
  \text{map} \; f \; [\;] = [\;] \tag{3} \\
  \text{map} \; f \; (x:xs) = f \; x : \text{map} \; f \; xs \tag{4}
  \]

- **Proposition:** (This is what we are trying to prove)
  \[P\{[xs]\}: \text{length} \; (\text{map} \; f \; xs) = \text{length} \; xs\]

- **Proof Structure:**
  1) **Prove** \(P\{[]\}:: \text{length} \; (\text{map} \; f \; []) = \text{length} \; []\)
  2) **Assume** \(P\{xs\}:: \text{(as well as the definitions and laws)} \text{length} \; (\text{map} \; f \; xs) = \text{length} \; xs\)
     Then **Prove** \(P\{x:xs\}:: \text{length} \; (\text{map} \; f \; (x:xs)) = \text{length} \; (x:xs)\)
Proof

1) **Prove:** \( \text{length} (\text{map} \ f \ \text{[]} ) = \text{length} \ [\ ] \)
   
   \( \text{length} (\text{map} \ f \ []) = \ (\text{by} \ 3:\ \text{map} \ f \ [] = [] \) \)
   
   \( \text{length} [\ ] \)

2) **Assume:** \( \text{length}(\text{map} \ f \ \text{xs}) = \text{length} \ \text{xs} \)

**Prove:** \( \text{length}(\text{map} \ f \ (\text{x}:\text{xs})) = \text{length} \ (\text{x}:\text{xs}) \)

   \( \text{length} (\text{map} \ f \ (\text{x}:\text{xs})) = \ (\text{by} \ 4:\ \text{map} \ f \ (\text{x}:\text{xs}) = f \ \text{x}: \text{map} \ f \ \text{xs}) \)
   
   \( \text{length} (f \ \text{x}: (\text{map} \ f \ \text{xs})) = \ (\text{by} \ 2:\ \text{length} \ (\text{x}:\text{xs}) = 1 + (\text{length} \ \text{xs}) ) \)
   
   \( 1 \ + \ \text{length}(\text{map} \ f \ \text{xs}) = \ (\text{by} \ \text{IH}) \)
   
   \( 1 \ + \ \text{length} \ \text{xs} = \ (\text{by} \ 2:\ \text{length} \ (\text{x}:\text{xs}) = 1 \ + \ \text{length} \ \text{xs} ) \)
   
   \( \text{length} \ (\text{x}:\text{xs}) \)
Example: Relating \text{sum} and ++

- **Definitions and Laws:** (These are things we get to assume are true)
  
  \begin{align*}
  \text{sum} \; [] &= 0 \quad (1) \\
  \text{sum} \; (x:xs) &= x + (\text{sum} \; xs) \quad (2) \\
  [] ++ ys &= ys \quad (3) \\
  (x:xs) ++ ys &= x:(xs ++ ys) \quad (4)
  \end{align*}

- **Proposition:** (This is what we are trying to prove)
  
  \[ P\{xs\} = \text{sum} \; (xs ++ ys) = \text{sum} \; xs + \text{sum} \; ys \]
  
  – why do we do induction on the first argument of ++?

- **Proof Structure:**
  
  – 1) **Prove** \( P\{[]\} \):
    
    \[ \text{sum} \; ([] ++ ys) = \text{sum} \; [] + \text{sum} \; ys \]
  
  – 2) **Assume** \( P\{xs\} \): (as well as the definitions and laws)
    
    \[ \text{sum} \; (xs ++ ys) = \text{sum} \; xs + \text{sum} \; ys \]
    
    **Then Prove** \( P\{x:xs\} \):
    
    \[ \text{sum} \; ((x:xs) ++ ys) = \text{sum} \; (x:xs) + \text{sum} \; ys \]
Proof

1) **Prove:** \( \text{sum ([]} + + \text{ys}) = \text{sum []} + \text{sum ys} \)
   - \( \text{sum ([]} + + \text{ys}) = (\text{by 3: } [] + + \text{ys} = \text{ys}) \)
   - \( \text{sum ys} = (\text{arithmetic: } 0 + n = n) \)
   - \( 0 + \text{sum ys} = (\text{by 1: } \text{sum []} = 0) \)
   - \( \text{sum []} + \text{sum ys} \)

2) **Assume:** \( \text{sum (xs ++ ys)} = \text{sum xs + sum ys} \)
   **Prove:** \( \text{sum ((x:xs) ++ ys)} = \text{sum (x:xs) + sum ys} \)
   - \( \text{sum ((x:xs) ++ ys)} = (\text{by 4: } (x:xs) ++ \text{ys} = x:\text{(xs ++ ys)}) \)
   - \( \text{sum (x:(xs++ys))} = (\text{by 2: } \text{sum (x:xs)} = x + (\text{sum xs}) \) \)
   - \( x + \text{sum(xs++ys)} = (\text{by IH}) \)
   - \( x + (\text{sum xs + sum ys}) = (\text{associativity of } +: (p + q) + r = p + (q + r)) \)
   - \( (x + \text{sum xs}) + \text{sum ys} = (\text{by 2: } \text{sum (x:xs)} = x + (\text{sum xs}) \) \)
   - \( \text{sum (x:xs) + sum ys} \)
Proof by induction using Case Analysis

• Prove by induction:
  \[ P\{xs\} == (\text{takeWhile } p \ xs) ++ (\text{dropWhile } p \ xs) = xs \]

• Where:
  (1) \[
  \[] ++ ys = ys
  \]
  (2) \[
  (x:xs) ++ ys = x : (xs ++ ys)
  \]
  (3) \[
  \text{dropWhile } p \ [] = []
  \]
  (4) \[
  \text{dropWhile } p \ (x:xs) = \begin{cases} 
  \text{dropWhile } p \ xs & \text{if } p \ x \\
  x :: xs & \text{else}
  \end{cases}
  \]
  (5) \[
  \text{takeWhile } p \ [] = []
  \]
  (6) \[
  \text{takeWhile } p \ (x:xs) = \begin{cases} 
  x : (\text{takeWhile } p \ xs) & \text{if } p \ x \\
  [] & \text{else}
  \end{cases}
  \]
Base and Inductive cases

- **Base case:** \( P \{ \ [ ] \} \)
  
  \[
  \text{(takeWhile } p [ ] \text{)} ++ \ (\text{dropWhile } p [ ]) = \ (\text{by 3}, 5) \\
  [ ] ++ [ ] = (\text{by 1}) \\
  [ ]
  \]

- **Induction Step:**
  \( P\{ys\} \Rightarrow P\{y:ys\} \)

  **Assume:**
  \[
  \text{(takeWhile } p \ ys \text{)} ++ \ (\text{dropWhile } p \ ys) = ys
  \]

  **Prove:**
  \[
  \text{(takeWhile } p \ (y:ys) \text{)} ++ \ (\text{dropWhile } p \ (y:ys)) = (y : ys)
  \]
Split Proof

\[(\text{takeWhile} \ p \ (y:ys)) \ ++ \ (\text{dropWhile} \ p \ (y:ys)) \ = \ (\text{by} \ 4,6)\]

\[
\begin{align*}
\text{(if } p \ y \ \text{then} \ y : (\text{takeWhile} \ p \ ys) \\
\quad \text{else } [] \ ) \ ++ \\
\quad \text{(if } p \ y \ \text{then} \ (\text{dropWhile} \ p \ ys) \\
\quad \quad \text{else } y : ys)
\end{align*}
\]

- Now, either \((p \ y) = \text{True}\) or \((p \ y) = \text{False}\)
- So split problem by doing a case analysis
Case 1: Assume: \( p \ y = \text{True} \)

\[
\begin{align*}
\text{(if } p \ y \text{ then } y : (\text{takeWhile } p \ ys) \text{ else } []) ++ \\
\text{(if } p \ y \text{ then } (\text{dropWhile } p \ ys) \text{ else } y:ys) &= \text{(by case assumption)}
\end{align*}
\]

\[
\begin{align*}
(y : (\text{takeWhile } p \ ys)) ++ (\text{dropWhile } p \ ys) &= \text{(by 2)}
\end{align*}
\]

\[
\begin{align*}
y : ((\text{takeWhile } p \ ys) ++ (\text{dropWhile } p \ ys)) &= \text{(by I.H.)}
\end{align*}
\]

\[
y : ys
\]
Case 2: Assume: \( p y = \text{False} \)

\[
\text{if } p y \text{ then } y : (\text{takeWhile } p \text{ ys}) \text{ else } [] \text{ }++
\]
\[
\text{if } p y \text{ then } (\text{dropWhile } p \text{ ys}) \text{ else } y:ys \text{ } = \text{ (by case assumption)}
\]

\[
[] ++ (y:ys) = \text{(by 1)}
\]

\( y:ys \)
Structural Induction over Trees

data Bintree a = Lf a
   \mid (Bintree a) :/\: (Bintree a)
\rightarrow
» Note all infix constructors start with a colon (:

• Assume the following definitions and facts:

  sumtree :: Bintree a -> Int

  (1) \text{sumtree} \ (Lf \ x) = x

  (2) \text{sumtree} \ (a :/\: b) = (\text{sumtree} \ a) + (\text{sumtree} \ b)

  flatten :: Bintree a -> [a]

  (3) \text{flatten} \ (Lf \ x) = [x]

  (4) \text{flatten} \ (a :/\: b) = (\text{flatten} \ a) ++ (\text{flatten} \ b)

  (5) \text{sum} \ [] = 0

  (6) \text{sum} \ (x:xs) = x + (\text{sum} \ xs)

  (7) \textit{Lemma:} \text{sum}(xs ++ ys) = (\text{sum} \ xs) + (\text{sum} \ ys)
Proofs on Trees

To prove a proposition $P(t)$ about all trees $t$, must prove it for each tree constructor, assuming it is true for all smaller trees.

So, to prove $P(t)$ on a Bintree, we must:

- Prove $P(Lf x)$
- Prove that $P(a) \land P(b) \Rightarrow P(a :/\: b)$

Example: Prove $P(t)$: $\text{sum} (\text{flatten } t) = \text{sumtree } t$

**case 1: Prove $P(Lf x)$: $\text{sum} (\text{flatten} \ (Lf \ x)) = \text{sumtree} \ (Lf \ x)$**

\[
\begin{align*}
\text{sum} (\text{flatten} \ (Lf \ x)) &= (\text{by 3: flatten} \ (Lf \ x) = [x]) \\
\text{sum} \ [x] &= (\text{by 6: sum} \ (x:xs) = x + \text{sum} \ xs) \\
x + (\text{sum} \ []) &= (\text{by 5: sum} \ [] = 0) \\
x + 0 &= (\text{by arithmetic: } x + 0 = x) \\
x &= (\text{by 1: sumtree}(Lf \ x) = x) \\
\text{sumtree} \ (Lf \ x)
\end{align*}
\]
Case 2

case 2: Prove $P\{a\} \land P\{b\} \Rightarrow P\{a :/\: b\}$

Assume:
1) $P\{a\}$: $\text{sum}(\text{flatten } a) = \text{sumtree } a$
2) $P\{b\}$: $\text{sum}(\text{flatten } b) = \text{sumtree } b$

Prove: $P\{a :/\: b\}$: 

$\text{sum}(\text{flatten } (a :/\: b)) = \text{sumtree}(a :/\: b)$

$\text{sum}(\text{flatten } (a :/\: b)) =$
by 4

$\text{sum } ((\text{flatten } a) + + (\text{flatten } b)) =$
by lemma: 7

$\text{sum}(\text{flatten } a) + \text{sum}(\text{flatten } b)$
by I.H. (twice)
$(\text{sumtree } a) + (\text{sumtree } b)$
by 2
$= \text{sumtree } (a :/\: b)$