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:

\[(f \cdot g) \circ x = f(g \circ x)\]  \hspace{1cm} \text{(comp)}

- Notice label on equation

Now prove that composition is associative, i.e.

\[((f \cdot g) \cdot h) \circ x = (f \cdot (g \cdot h)) \circ x\]

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

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

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

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

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

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

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

• Consider the algebra of Shapes (Ch. 8)
• Suppose we have already proved (Hudak p.100-101):

  \[ r \text{ `Union` Empty} = r \]  (Axiom 4a)
  \[ r \text{ `Intersect` univ} = r \]  (Axiom 4b)
  \[ r \text{ `Union` Complement r} = \text{univ} \]  (Axiom 5a)
  \[ r \text{ `Intersect` Complement r} = \text{Empty} \]  (Axiom 5b)
  \[ r1 \text{ `Union` (} r2 \text{ `Intersect` r3}) = (r1 \text{ `Union` r2) `Intersect` (r1 `Union` r3) \] (Axiom 3b)

• Prove: \[ r \text{ `Union` r} = r \]

  \[ r = \]  (by Axiom 4a)
  \[ r \text{ `Union` Empty} = \]  (by 5b)
  \[ r \text{ `Union` (} r \text{ `Intersect` Complement r}) = \]  (by 3b)
  \[ (r \text{ `Union` r) `Intersect` (r `Union` Complement r}) = \]  (by 5a)^
  \[ (r \text{ `Union` r) `Intersect` univ} = \]  (by 4b)

  \[ r \text{ `Union` r} \]
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.
  
  length x = length (reverse x)
  all p x => p (head x)
  sum (x ++ y) = sum x + sum y
  map f (x ++ y) = map f x ++ map f y
  (map f . map g) x = 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)
  \[ \text{length } [] = 0 \]  (1)
  \[ \text{length } (x:xs) = 1 + \text{length } xs \]  (2)
  \[ \text{map } f [] = [] \]  (3)
  \[ \text{map } f (x:xs) = f x: \text{map } f xs \]  (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 \( P\{xs\}: \) (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{length} [\] \)

\[
\text{length} (\text{map} \ f \ [\]) = \quad (\text{by 3:} \ \text{map} \ f \ [\] = [\]) \\
\text{length} [\]
\]

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

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

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

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

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

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

1) Prove: \( \text{sum } ([ ] + y) = \text{sum } [ ] + \text{sum } y \)
   
   \[
   \text{sum } ([ ] + y) = \quad \text{(by 3: \([ ] + y = y\))}
   \]
   
   \[
   \text{sum } y = \quad \text{(arithmetic: } 0 + n = n\text{)}
   \]
   
   \[
   0 + \text{sum } y = \quad \text{(by 1: \text{sum } [ ] = 0)}
   \]
   
   \[
   \text{sum } [ ] + \text{sum } y
   \]

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

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

• Where:
  (1) \([\ ] ++ ys = ys\)
  (2) \((x:xs) ++ ys = x : (xs ++ ys)\)

  (3) \(\text{dropWhile } p \ [\ ] = [\] \)
  (4) \(\text{dropWhile } p \ (x:xs) = \)
      \[
      \text{if } p \ x \text{ then } (\text{dropWhile } p \ x\ s) \\
      \text{else } x::xs
      \]

  (5) \(\text{takeWhile } p \ [\ ] = [\] \)
  (6) \(\text{takeWhile } p \ (x:xs) = \)
      \[
      \text{if } p \ x \text{ then } x:(\text{takeWhile } p \ x\ s) \\
      \text{else } [\]
      \]
Base and Inductive cases

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

• Induction Step:
  P{ys} => P{y:ys}

Assume:
  (takeWhile p ys) ++ (dropWhile p ys) = ys

Prove:
  (takeWhile p (y:ys)) ++ (dropWhile p (y:ys)) = (y : ys)
Split Proof

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

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

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

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

\[
(y : (\text{takeWhile } p \ ys)) ++ (\text{dropWhile } p \ ys) = \text{ (by 2)}
\]

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

\[
y : ys
\]
Case 2: Assume: $\ p \ y = \ False$

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

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

$y:ys$
Structural Induction over Trees

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

• Assume the following definitions and facts:

  sumtree :: Bintree a -> Int
(1) sumtree (Lf x) = x
(2) sumtree (a :/\: b) = (sumtree a) + (sumtree b)
  flatten :: Bintree a -> [a]
(3) flatten (Lf x) = [x]
(4) flatten (a :/\: b )=(flatten a) ++ (flatten b)

(5) sum [] = 0
(6) sum (x:xs) = x + (sum xs)
(7) Lemma: sum(xs ++ ys) = (sum xs) + (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 $P\{\text{Lf } x\}$
- Prove that $P\{a\} \land P\{b\} \Rightarrow P\{a : /\ : b\}$

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

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

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

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

\[
x + (\text{sum } []) = (\text{by 5: sum } [] = 0)
\]

\[
x + 0 = (\text{by arithmetic: } x + 0 = x)
\]

\[
x = (\text{by 1: sumtree(Lf } x) = x)
\]

\[
\text{sumtree (Lf } x)
\]
Case 2

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

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

Prove: $P\{a :/\!: b\}$: $\text{sum(flinten (a :/\!: b))} = \text{sumtree(a :/\!: b)}$

\[
\text{sum(flinten (a :/\!: b))} = \\
\quad \text{by 4}
\]
\[
\text{sum ((fliten a) + (fliten b))} = \\
\quad \text{by lemma: 7}
\]
\[
\text{sum(flinten a) + sum(flinten b)} = \\
\quad \text{by I.H. (twice)}
\]
\[
(\text{sumtree a}) + (\text{sumtree b}) = \\
\quad \text{by 2}
\]
\[
\text{sumtree (a :/\!: b)}
\]