COSC 3015: Lecture 13

Lecture given by Prof. Caldwell and scribed by Sunil Kothari

October 7, 2008

1 Trees

Chapter 5 has some really good examples. But, we will move on. And we may move to Higher-Order Perl book. We will start with Chapter 6 on trees. Today's lecture is a sort of review, since many of the things discussed here are in 2300. A binary tree is defined as:
$data\ Btree\ a = Leaf\ a \mid Fork\ (Btree\ a)\ (Btree\ a)$

So, here's an example of a tree:

```
Fork (Leaf 1)
     (Fork (Leaf 2)
           (Leaf 3))
```

```
 Main> :t Leaf
Leaf :: a -> Btree a
```

You get case statement for free when you create a datatype. For example, consider the size function

```
size t =
   case t of
    (Leaf _) -> 1
    (Fork xt yt) -> size xt + size yt

Main> :t size
size :: Num a => Btree b -> a
Main>
```

So, case is like a destructor for the datatype. What's the other way of writing it ?

```
size1 (Leaf _) = 1
size1 (Fork xt yt) = size1 xt + size1 yt

Main> :t size1
size1 :: Num a => Btree b -> a
```

Whenever you have a datatype, you also get a structural induction principle. The induction principle for Btree is

$$\forall x : a.P(\text{Leaf } x) \wedge (\forall xt, yt : (Btree\ a).P(xt) \wedge P(yt) \Rightarrow P(Fork\ xt\ yt)) \\ \Rightarrow \forall t : (Btree\ a).P(t)$$

1. $\forall x : a.P(\text{Leaf}x)$ [base case]
2. $\forall xt, yt : (Btree\ a).P(xt) \wedge P(yt) \Rightarrow P(Fork\ xt\ yt)$ [induction case]

We also have extra case for partial elements of the type

- $P(\perp)$

Definition 1. *A Btree is finite if and only if* $size(t) \neq \perp$.

The book says that any finite path through a syntax tree of a recursive datatype is linear. For example

$$\begin{array}{c} succ \\ | \\ Zero \\ succ \\ | \\ succ \\ | \\ Zero \end{array}$$

Similar is the case with lists.

So, let's look at different trees - something used in 2300.

$$data\ TTree\ a = \text{Leaf} \mid Node2\ a\ (TTree\ a)\ (TTree\ a)$$

Or we can also have a tree with three sub-trees.

$$data\ TTTree\ a = \text{Leaf} \mid Node3\ a\ (TTTree\ a)\ (TTTree\ a)\ (TTTree\ a)$$

Q: Can we make circular structures ?
A: No, we can't. But, we can define Nat as a form of DAG (directed acyclic graph)

$$dataNat' = Zero \mid SNat' \mid SSNat'$$

where, $SS\ k = S\ (S\ k)$.

So, what about the (finite) induction principle for $TTTree\ a$.

1. $P(Leaf)$ [base case]
2. $\forall x : a, \forall xt, yt, zt : (Btree\ a).\ P(xt) \wedge P(yt) \wedge P(zt) \Rightarrow P(Node3\ x\ xt\ yt\ zt)$ [induction case]

If we want infinite induction, we ought to have $P(\perp)$ too.
Let's define *flatten*.

```
flatten:: Btree a ->  [a]
flatten (Leaf x) = [x]
flatten (Fork t t') = flatten t ++ flatten t'
```

$$
\begin{aligned}
& flatten\ (Node\ (Leaf\ 1)\ (Node\ (Leaf\ 2)\ (Leaf\ 3))) \\
= \ & flatten(Leaf 1) {+}{+} flatten(Fork(Leaf 2)(Leaf 3)) \\
= \ & [1] {+}{+} (flatten(Leaf 2) {+}{+} flatten(Leaf 3)) \\
= \ & [1] {+}{+} ([2] {+}{+} [3]) \\
= \ & [1, 2, 3]
\end{aligned}
$$

Theorem 1. $size = (length.flatten)$

Proof. By extensionality we must show $\forall t : Btree\ a, (size\ t) = (length.flatten)\ t$. Continue by structural induction on t. There are two cases:

$P(leaf)$ We must show

$$\forall x : a.P(leaf\ x).$$

i.e. that $\forall x : a.size(leaf\ x) = (length.flatten)(leaf\ x)$
Choose arb. $x \in a$ and show

$$size(leaf\ x) = (length.flatten)(leaf\ x)$$

On the left:

$$size\ (leaf\ x) = 1$$

On the right:

$$
\begin{aligned}
(length.flatten)(leaf\ x) &= length\ (flatten\ (leaf\ x)) \\
&= length\ [x] \\
&= 1
\end{aligned}
$$

$P(Fork\ xs\ ys)$: Assume $P(xs)$ and $P(ys)$ and show $P(Fork\ xs\ ys)$:

$$P(xs) : size\ xs = (length.flatten\ xs)$$
$$P(ys) : size\ ys = (length.flatten\ ys)$$

Show

$$size(fork\ xs\ ys) = length.flatten(fork\ xs\ ys)$$

On the left side

$$\begin{aligned} size(fork\ xs\ ys) &= size\ xs + size\ ys \\ &= (length.flatten\ xs) + (length.flatten\ ys) \\ &= (length\ (flatten\ xs)) + (length\ (flatten\ ys)) \end{aligned}$$

On the right

$$\begin{aligned} length.flatten\ (fork\ xs\ ys) &\overset{\text{compose}}{=} length(flatten\ (fork\ xs\ ys)) \\ &\overset{\text{flatten}}{=} length(flatten\ xs{+}{+}flatten\ ys) \\ &\overset{\text{length-append}}{=} (length\ (flatten\ xs)) + (length\ (flatten\ ys)) \end{aligned}$$

□

Note that "." is a function composition operator.

Lemma 1 (Length-Append)**.** $\forall xs, ys : [a].\ \mid xs{+}{+}ys \mid = \mid xs \mid + \mid ys \mid$

We can define a function *nodes* as:

```
nodes (Leaf _) = 0
nodes (Fork xs ys) = 1 + nodes xs + nodes ys
```

Then, we can prove a theorem

Theorem 2. $\forall xt : Btree\ a.size\ xt = 1 + nodes\ xt$

Proof. By ind. on xt.
P(xt) $\overset{\text{def}}{=}$ size xt = 1+ nodes xt
Again, we have two cases:

Case P(leaf x). size (Leaf x) = 1 + nodes (Leaf x)
L.H.S
size (Leaf x) = 1
R.H.S.
1+ nodes (Leaf x) = 1 + 0 = 1
so the base case holds.

Case P(Fork xs ys). Assume

size xs = 1+ nodes xs
size ys = 1 + nodes ys
Show
size (Fork xs ys) = 1 + nodes xs ys
L.H.S = size (Fork xs ys)
= 1 + nodes xs + 1 + nodes ys
R.H.S
1 + nodes(Fork xs ys)
= 1 + 1 + nodes xs + nodes ys

□