Structural Induction on Trees

Structural Induction on Trees

Structural induction is not limited to lists; it applies to any tree structure.

The general induction principle is the following:

To prove a property $\texttt{P(t)}$ for all trees $\texttt{t}$ of a certain type,

- show that $\texttt{P(l)}$ holds for all leaves $\texttt{l}$ of a tree,
- for each type of internal node $\texttt{t}$ with subtrees $\texttt{s}_1, \ldots, \texttt{s}_n$, show that

  $$\texttt{P}(\texttt{s}_1) \land \ldots \land \texttt{P}(\texttt{s}_n) \textit{ implies } P(t).$$

Example: IntSets

Recall our definition of `IntSet` with the operations `contains` and `incl`:

```
abstract class IntSet {
   def incl(x: Int): IntSet
   def contains(x: Int): Boolean
}

object Empty extends IntSet {
  def contains(x: Int): Boolean = false
  def incl(x: Int): IntSet = NonEmpty(x, Empty, Empty)
}
```

Example: IntSets (2)

```
case class NonEmpty(elem: Int, left: IntSet, right: IntSet) extends IntSet {

  def contains(x: Int): Boolean =
    if (x < elem) left contains x
    else if (x > elem) right contains x
    else true

  def incl(x: Int): IntSet =
    if (x < elem) NonEmpty(elem, left incl x, right)
    else if (x > elem) NonEmpty(elem, left, right incl x)
    else this
}
```

The Laws of IntSet

What does it mean to prove the correctness of this implementation?

One way to define and show the correctness of an implementation consists of proving the laws that it respects.

In the case of `IntSet`, we have the following three laws:

For any set `s`, and elements `x` and `y`:

```
Empty contains x          =  false
(s incl x) contains x     =  true
(s incl x) contains y     =  s contains y        if x != y
```

(In fact, we can show that these laws completely characterize the desired data type).

Proving the Laws of IntSet (1)

How can we prove these laws?

Proposition 1: `Empty contains x = false`.

Proof: According to the definition of `contains` in `Empty`.

Proving the Laws of IntSet (2)

Proposition 2: `(s incl x) contains x = true`

Proof by structural induction on `s`.

$\boxed{\textbf{Base case: } \texttt{Empty}}$

```
(Empty incl x) contains x
```

Proving the Laws of IntSet (2)

Proposition 2: `(s incl x) contains x = true`

Proof by structural induction on `s`.

Base case: `Empty`

```
(Empty incl x) contains x

=   NonEmpty(x, Empty, Empty) contains x   // by definition of Empty.incl
```

Proving the Laws of IntSet (2)

Proposition 2: `(s incl x) contains x = true`

Proof by structural induction on `s`.

Base case: `Empty`

```
(Empty incl x) contains x

=   NonEmpty(x, Empty, Empty) contains x   // by definition of Empty.incl

=   true                                   // by definition of NonEmpty.contains
```

Proving the Laws of IntSet (3)

Induction step: `NonEmpty(x, l, r)`

```
(NonEmpty(x, l, r) incl x) contains x
```

Proving the Laws of IntSet (3)

Induction step: `NonEmpty(x, l, r)`

```
(NonEmpty(x, l, r) incl x) contains x

=   NonEmpty(x, l, r) contains x          //  by definition of NonEmpty.incl
```

Proving the Laws of IntSet (3)

Induction step: `NonEmpty(x, l, r)`

```
(NonEmpty(x, l, r) incl x) contains x

=   NonEmpty(x, l, r) contains x          //  by definition of NonEmpty.incl

=   true                                  //  by definition of NonEmpty.contains
```

Proving the Laws of IntSet (4)

Induction step: `NonEmpty(y, l, r)` **where** `y < x`

```
(NonEmpty(y, l, r) incl x) contains x
```

Proving the Laws of IntSet (4)

Induction step: `NonEmpty(y, l, r)` **where** `y < x`

```
(NonEmpty(y, l, r) incl x) contains x

=   NonEmpty(y, l, r incl x) contains x    // by definition of NonEmpty.incl
```

Proving the Laws of IntSet (4)

Induction step: `NonEmpty(y, l, r)` **where** `y < x`

```
(NonEmpty(y, l, r) incl x) contains x

=   NonEmpty(y, l, r incl x) contains x   // by definition of NonEmpty.incl

=   (r incl x) contains x                 // by definition of NonEmpty.contains
```

Proving the Laws of IntSet (4)

Induction step: `NonEmpty(y, l, r)` **where** `y < x`

```
(NonEmpty(y, l, r) incl x) contains x

=   NonEmpty(y, l, r incl x) contains x    // by definition of NonEmpty.incl

=   (r incl x) contains x                  // by definition of NonEmpty.contains

=   true                                   // by the induction hypothesis
```

Proving the Laws of IntSet (4)

Induction step: `NonEmpty(y, l, r)` **where** `y < x`

```
(NonEmpty(y, l, r) incl x) contains x

=   NonEmpty(y, l, r incl x) contains x    // by definition of NonEmpty.incl

=   (r incl x) contains x                  // by definition of NonEmpty.contains

=   true                                   // by the induction hypothesis
```

Induction step: `NonEmpty(y, l, r)` **where** `y > x` is analogous

Proving the Laws of IntSet (5)

Proposition 3: If `x != y` then

```
(xs incl y) contains x  =  xs contains x.
```

Proof by structural induction on `s`. Assume that `y < x` (the dual case `x < y` is analogous).

Base case: `Empty`

```
(Empty incl y) contains x                    // to show: =  Empty contains x
```

Proving the Laws of IntSet (5)

Proposition 3: If `x != y` then

```
(xs incl y) contains x  =  xs contains x.
```

Proof by structural induction on `s`. Assume that `y < x` (the dual case `x < y` is analogous).

Base case: `Empty`

```
(Empty incl y) contains x                    // to show: =  Empty contains x

 =   NonEmpty(y, Empty, Empty) contains x    // by definition of Empty.incl
```

Proving the Laws of IntSet (5)

Proposition 3: If `x != y` then

```
(xs incl y) contains x  =  xs contains x.
```

Proof by structural induction on `s`. Assume that `y < x` (the dual case `x < y` is analogous).

Base case: `Empty`

```
(Empty incl y) contains x                 // to show: =  Empty contains x

=   NonEmpty(y, Empty, Empty) contains x   // by definition of Empty.incl

=   Empty contains x                       // by definition of NonEmpty.contains
```

Proving the Laws of IntSet (6)

For the inductive step, we need to consider a tree `NonEmpty(z, l, r)`. We distinguish five cases:

1. `z = x`
2. `z = y`
3. `z < y < x`
4. `y < z < x`
5. `y < x < z`

First Two Cases: `z = x`, `z = y`

Induction step: `NonEmpty(x, l, r)`

```
(NonEmpty(x, l, r) incl y) contains x  // to show: =  NonEmpty(x, l, r) contains x
```

First Two Cases: `z = x, z = y`

Induction step: `NonEmpty(x, l, r)`

```
(NonEmpty(x, l, r) incl y) contains x  // to show: =  NonEmpty(x, l, r) contains x

=   NonEmpty(x, l incl y, r) contains x    // by definition of NonEmpty.incl
```

First Two Cases: `z = x, z = y`

Induction step: `NonEmpty(x, l, r)`

```
(NonEmpty(x, l, r) incl y) contains x  // to show: =  NonEmpty(x, l, r) contains x

=   NonEmpty(x, l incl y, r) contains x    // by definition of NonEmpty.incl

=   true                                    // by definition of NonEmpty.contains
```

First Two Cases: `z = x, z = y`

Induction step: `NonEmpty(x, l, r)`

```
(NonEmpty(x, l, r) incl y) contains x  // to show: =  NonEmpty(x, l, r) contains x

=   NonEmpty(x, l incl y, r) contains x    // by definition of NonEmpty.incl

=   true                                   // by definition of NonEmpty.contains

=   NonEmpty(x, l, r) contains x           // by definition of NonEmpty.contains
```

First Two Cases: `z = x, z = y`

Induction step: `NonEmpty(x, l, r)`

```
(NonEmpty(x, l, r) incl y) contains x  // to show: =  NonEmpty(x, l, r) contains x

=   NonEmpty(x, l incl y, r) contains x    // by definition of NonEmpty.incl

=   true                                   // by definition of NonEmpty.contains

=   NonEmpty(x, l, r) contains x           // by definition of NonEmpty.contains
```

Induction step: `NonEmpty(y, l, r)`

```
(NonEmpty(y, l, r) incl y) contains x  // to show: =  NonEmpty(y, l, r) contains x
```

First Two Cases: `z = x, z = y`

Induction step: `NonEmpty(x, l, r)`

```
(NonEmpty(x, l, r) incl y) contains x  // to show: =  NonEmpty(x, l, r) contains x

=   NonEmpty(x, l incl y, r) contains x     // by definition of NonEmpty.incl

=   true                                    // by definition of NonEmpty.contains

=   NonEmpty(x, l, r) contains x            // by definition of NonEmpty.contains
```

Induction step: `NonEmpty(y, l, r)`

```
(NonEmpty(y, l, r) incl y) contains x  // to show: =  NonEmpty(y, l, r) contains x

=    NonEmpty(y, l, r) contains x           // by definition of NonEmpty.incl
```

Case `z < y`

Induction step: `NonEmpty(z, l, r)` **where** `z < y < x`

```
(NonEmpty(z, l, r) incl y) contains x  // to show: =  NonEmpty(z, l, r) contains x
```

Case z < y

Induction step: `NonEmpty(z, l, r)` **where** `z < y < x`

```
(NonEmpty(z, l, r) incl y) contains x  // to show: =  NonEmpty(z, l, r) contains x

=   NonEmpty(z, l, r incl y) contains x    // by definition of NonEmpty.incl
```

Case z < y

Induction step: `NonEmpty(z, l, r)` **where** `z < y < x`

```
(NonEmpty(z, l, r) incl y) contains x  // to show: =  NonEmpty(z, l, r) contains x

=   NonEmpty(z, l, r incl y) contains x    // by definition of NonEmpty.incl

=   (r incl y) contains x                  // by definition of NonEmpty.contains
```

Case z < y

Induction step: `NonEmpty(z, l, r)` **where** `z < y < x`

```
(NonEmpty(z, l, r) incl y) contains x  // to show: =  NonEmpty(z, l, r) contains x

=   NonEmpty(z, l, r incl y) contains x    // by definition of NonEmpty.incl

=   (r incl y) contains x                  // by definition of NonEmpty.contains

=   r contains x                           // by the induction hypothesis
```

Case z < y

Induction step: `NonEmpty(z, l, r)` **where** `z < y < x`

```
(NonEmpty(z, l, r) incl y) contains x  // to show: =  NonEmpty(z, l, r) contains x

=   NonEmpty(z, l, r incl y) contains x    // by definition of NonEmpty.incl

=   (r incl y) contains x                  // by definition of NonEmpty.contains

=   r contains x                           // by the induction hypothesis

=   NonEmpty(z, l, r) contains x           // by definition of NonEmpty.contains
```

Case y < z < x

Induction step: `NonEmpty(z, l, r)` **where** `y < z < x`

```
(NonEmpty(z, l, r) incl y) contains x  // to show: =  NonEmpty(z, l, r) contains x
```

Case y < z < x

Induction step: `NonEmpty(z, l, r)` **where** `y < z < x`

```
(NonEmpty(z, l, r) incl y) contains x  // to show: =  NonEmpty(z, l, r) contains x

=   NonEmpty(z, l incl y, r) contains x    // by definition of NonEmpty.incl
```

Case y < z < x

Induction step: `NonEmpty(z, l, r)` **where** `y < z < x`

```
(NonEmpty(z, l, r) incl y) contains x  // to show: =  NonEmpty(z, l, r) contains x

=   NonEmpty(z, l incl y, r) contains x    // by definition of NonEmpty.incl

=   r contains x                           // by definition of NonEmpty.contains
```

Case y < z < x

Induction step: `NonEmpty(z, l, r)` **where** `y < z < x`

```
(NonEmpty(z, l, r) incl y) contains x  // to show: =  NonEmpty(z, l, r) contains x

=   NonEmpty(z, l incl y, r) contains x    // by definition of NonEmpty.incl

=   r contains x                           // by definition of NonEmpty.contains

=   NonEmpty(z, l, r) contains x           // by definition of NonEmpty.contains
```

Case `x < z`

Induction step: `NonEmpty(z, l, r)` **where** `y < x < z`

```
(NonEmpty(z, l, r) incl y) contains x  // to show: =  NonEmpty(z, l, r) contains x
```

Case `x < z`

Induction step: `NonEmpty(z, l, r)` **where** `y < x < z`

```
(NonEmpty(z, l, r) incl y) contains x  // to show: =  NonEmpty(z, l, r) contains x

=   NonEmpty(z, l incl y, r) contains x    // by definition of NonEmpty.incl
```

Case x < z

Induction step: `NonEmpty(z, l, r)` **where** `y < x < z`

```
(NonEmpty(z, l, r) incl y) contains x  // to show: =  NonEmpty(z, l, r) contains x

=   NonEmpty(z, l incl y, r) contains x    // by definition of NonEmpty.incl

=   (l incl y) contains x                  // by definition of NonEmpty.contains
```

Case x < z

Induction step: `NonEmpty(z, l, r)` **where** `y < x < z`

```
(NonEmpty(z, l, r) incl y) contains x  // to show: =  NonEmpty(z, l, r) contains x

=   NonEmpty(z, l incl y, r) contains x    // by definition of NonEmpty.incl

=   (l incl y) contains x                  // by definition of NonEmpty.contains

=   l contains x                           // by the induction hypothesis
```

Case x < z

Induction step: `NonEmpty(z, l, r)` **where** `y < x < z`

```
(NonEmpty(z, l, r) incl y) contains x  // to show: =  NonEmpty(z, l, r) contains x

=   NonEmpty(z, l incl y, r) contains x    // by definition of NonEmpty.incl

=   (l incl y) contains x                  // by definition of NonEmpty.contains

=   l contains x                           // by the induction hypothesis

=   NonEmpty(z, l, r) contains x           // by definition of NonEmpty.contains
```

These are all the cases, so the proposition is established.

Exercise (Hard)

Suppose we add a function `union` to `IntSet`:

```
abstract class IntSet { ...
  def union(other: IntSet): IntSet
}
object Empty extends IntSet { ...
  def union(other: IntSet) = other
}
class NonEmpty(x: Int, l: IntSet, r: IntSet) extends IntSet { ...
  def union(other: IntSet): IntSet = (l union (r union (other))) incl x
}
```

Exercise (Hard)

The correctness of `union` can be translated into the following law:

Proposition 4:

```
(xs union ys) contains x  =  xs contains x || ys contains x
```

Show proposition 4 by using structural induction on `xs`.