Some stuff on Induction

Thomas Sewell

NICTA

24 July 2010

Thought I'd talk about induction, since it has direct relevance to functional programming.

Two main messages:

Thought I'd talk about induction, since it has direct relevance to functional programming.

Two main messages:

- Induction is important.

Thought I'd talk about induction, since it has direct relevance to functional programming.

Two main messages:

- Induction is important.
- Induction is hard.

How do you prove something about a loop?

for $(x = 0; x < 12; x++)\{y = y + (x * x);\}$

How do you prove something about a loop?

for $(x = 0; x < 12; x++)\{y = y + (x * x);\}$

We all know that recursion is more interesting than looping ...

What do we mean by recursion?

$\text{sort } xs = \text{let } (ys, zs) = \text{split\_evenly } xs \text{ in merge } (\text{sort } ys)\ (\text{sort } zs)$

What do we mean by recursion?

sort xs = let (ys, zs) = split_evenly xs in merge (sort ys) (sort zs)

Think of the inner sort as the "real" sort function.

What do we mean by recursion?

sort xs = let (ys, zs) = split_evenly xs in merge (sort ys) (sort zs)

Think of the inner sort as the "real" sort function.

This freaks out our inner imperative programmer.

What do we mean by recursion?

sort xs = let (ys, zs) = split_evenly xs in merge (sort ys) (sort zs)

Think of the inner sort as the "real" sort function.

This freaks out our inner imperative programmer.

This is inductive logic.

What about the recursion, in, say, Ben's typing rules:

$$\frac{\Gamma, \Theta \mid -f :: \alpha \to \beta \qquad \Gamma, \Theta \mid -n :: \alpha}{\Gamma, \Theta \mid -f\ n :: \beta}$$

Induction is where theorem proving gets hard, because we need to state a special kind of invariant.

Induction is where theorem proving gets hard, because we need to state a special kind of invariant. Not the kind of invariant we could find through a debugger.

Conclusion

Conclusion:

- Induction is important.
- Induction is hard.