Meeting 09: Structural Induction

HW2 "Hardness" 2 weeks (4 credits)
- Time: 14.3 hrs avg, 9.5 std dev, 12 median
- Hardness: 5.1 avg, 0.9 std dev, 5 median

HW1 "Hardness" 1 week
- Time: 4.7 avg, 2.8 std dev, 4 median
- Hardness: 3.7 avg, 0.9 std dev, 4 median

Suggestions
- Preface new topics
- Programming: some more, some less
- Pace: increase?
- Readings
  - Interactive discussion
  - Posting lecture notes

Why Scala?
- Scala: Twitter uses it!
- Grammars = what you can write
  - Pattern matching + ASTs = process strings/ XML
  - Regex nightmare
- Induction

Nice: Program Correctness (B:11)

Alex: Thinking Inductively

= Thinking Recursively

- Higher Functions: Map/Reduce

Hadoop

---

Structural Induction

\[
\begin{align*}
\text{l} & \in \text{List}[	ext{T}] \\
\text{0} & \quad \text{l} = \text{NIL} \quad (\text{l} \text{ is the empty list}) \\
\text{2} & \quad \text{l} = \text{h} \cdot \text{t} \quad (\text{l} \text{ can be decomposed into head element } h \text{ and tail } t) \\
& \quad \text{for some } h : T, t : \text{List}[T]
\end{align*}
\]

"Thinking Recursive w/ AST"
def append[CTJ][x1:List[CTJ], y1:List[CTJ]]:List[CTJ]
(l:Nil) x≤ match 2
  Case Nil => y1
  Case xh::x≥ => h:: append(x≥, y1)

3 pattern binding

3 match 5
  Case a::(b::t) =>

(Scala)

Theorem: For all List values x≥, y≥
append(x≥, y≥) →* v
for some value v.

Proof By structural induction on x≥
Base Case: \( xl = \text{Nil} \)

Assume some arbitrary list value \( y' \).

\[
\text{append} (\text{Nil}, y) \rightarrow y'
\]

by the definition of \text{append}

Here’s the value

\( v \)

\( \text{cons} \)

Inductive Case: \( xl = xh :: xt \)

Assume some arb. list value \( yl \) for some values \( xh, xt : \text{List} \).

Inductive Hypothesis: For all list values \( yl' \).

\[
\text{append} (xt, yl') \rightarrow v'
\]

for some value \( v' \)

\( v' \)

\( \text{cons} \)

1. \[
\text{append} (xh :: xt, yl) \rightarrow xh :: \text{append} (xt, yl)
\]

by the definition of \text{append}

2. \[
\text{append} (xt, yl) \rightarrow v' \text{ for some } v'
\]

by the i.h. on \( xt \) with \( yl' = yl \)
3. $x h:: append (x, y) \rightarrow x h:: v'$

by (2)

Here's our value

\[
\text{rev} \ (x: \text{List}\ C T) : \text{List}\ C T
\]

\[
\text{rev}\ \text{Iter} \ (x: \text{List}\ C T, \ acc: \text{List}\ C T) : \text{List}\ C T
\]

For list values $x, v, a$

Theorem: If $\text{rev}\ \text{Iter} \ (x, a) \rightarrow \emptyset \ \checkmark$

then $\text{append} (\text{rev} (x), a) \rightarrow \emptyset \ \checkmark$