Meeting 08: Structural Induction

Questions

1. Concrete vs. Abstract Syntax
   - Ambiguity

3. Precedence

2. Proving ambiguity (Brad)

Concrete

11

strings

Abstract

4

trees

Show a grammar is ambiguous to give two parse trees for the same string

\[ e ::= 1 | e + e | e \cdot e \]

\[ e ::= 1 | + (e, e) \]
Abstract Syntax Trees

\[
\begin{align*}
\text{e ::= } & \ 1 \mid e + e \mid e \times e \\
\text{treat } + & \text{ as left associative} \\
\text{treat } \times & \text{ as left associative} \\
(1 + 1) + 1 & \\
\text{(1} \times 1\text{) } & \text{x has higher precedence than +} \\
\text{Still ambiguous} & \\
\end{align*}
\]
sealed abstract class MyOption[T]
case class MySome[T](v: T) extends MyOption[T]

Case class My None extends MyOption[T]

f(): Option[Int]

x: Option[Int] match {  
  case None => ... error handling ...  
  case Some(v) => v ....  
}
def deref(x: Option[Int]): T =  
x match {  
  case None => throw new NoneException  
  case Some(v) => v  
}
\[
\begin{array}{c}
\text{Node (Node (EC, 2, EC), 1, Node (EC, 3, EC))} \\
\Rightarrow \text{List(2, 1, 3)}
\end{array}
\]

Structural Induction
Program Correctness

factorial

**TOTAL CORRECTNESS**

Theorem: For any integer $n$, $n \neq 0$

$$\text{factorial}(Ln) \rightarrow *2n!0.$$  

**PARTIAL CORRECTNESS**

Like to prove the first half

"it does what it is supposed"

"assuming it terminates"

Theorem: For any integer $n$, 
if
$\text{factorial}(Ln) \rightarrow *$
then $n = h n!$.
A list \( l \) is a list \([T]\) value (for some \( T \))

\[ l = \text{Nil} \]

or \( l = h :: e \) where \( h \) is a value for type \( T \) and \( e \)

is a value of type \([T]\)

For any list value \( l \), something about \( l \)

Method: Proof by structural induction \( P(l) \)

Base Case: \( l = \text{Nil} \) on \( l \).

Show \( P(\text{Nil}) \)

Inductive Case: \( l = h :: e \) for some \( h, e \)

Assume \( P(e) \)

To show \( P(h :: e) \)
Natural Numbers
= Unary Number

sealed abstract class Nat
case class Zero extends Nat
case class Succ(n: Nat) extends Nat