CSE 311: Foundations of Computing

Lecture 17: Structural Induction

What's that Doctor McCluckles? Making them ovoid would increase structural integrity and enable a more comfortable delivery? He's right again Professor!
Midterm

• Midterm in class next Wednesday

• Covers material up to ordinary induction (HW5)

• Closed book, closed notes
  – will provide reference sheets

• No calculators
  – arithmetic is intended to be straightforward
  – (only a small point deduction anyway)
Midterm

• 5 problems covering:
  – Propositional Logic
    Including circuits / Boolean algebra / normal forms
  – Predicate Logic/English Translation
  – Modular arithmetic
  – Set theory
  – Induction

• 10 minutes per problem
  – write quickly, don’t get stuck on one problem
  – focus on the overall structure of the solution
Lecture 17: Structural Induction
How to prove $\forall x \in S, P(x)$ is true:

**Base Case:** Show that $P(u)$ is true for all specific elements $u$ of $S$ mentioned in the *Basis step*.

**Inductive Hypothesis:** Assume that $P$ is true for some arbitrary values of each of the *existing named elements* mentioned in the *Recursive step*.

**Inductive Step:** Prove that $P(w)$ holds for each of the new elements $w$ constructed in the *Recursive step* using the named elements mentioned in the Inductive Hypothesis.

**Conclude** that $\forall x \in S, P(x)$.
Last time: Using Structural Induction

- Let \( S \) be given by...
  - **Basis:** \( 6 \in S; \ 15 \in S \)
  - **Recursive:** if \( x, y \in S \) then \( x + y \in S \).

**Claim:** Every element of \( S \) is divisible by \( 3 \).
Last time: Every element of $S$ is divisible by 3.

1. Let $P(x)$ be “$3 \mid x$”. We prove that $P(x)$ is true for all $x \in S$ by structural induction.

2. Base Case: $3 \mid 6$ and $3 \mid 15$ so $P(6)$ and $P(15)$ are true.

3. Inductive Hypothesis: Suppose that $P(x)$ and $P(y)$ are true for some arbitrary $x, y \in S$.

4. Inductive Step: Goal: Show $P(x+y)$.
   
   Since $P(x)$ is true, $3 \mid x$ and so $x=3m$ for some integer $m$ and since $P(y)$ is true, $3 \mid y$ and so $y=3n$ for some integer $n$. Therefore $x+y=3m+3n=3(m+n)$ and thus $3 \mid (x+y)$.
   
   Hence $P(x+y)$ is true.

5. Therefore by induction $3 \mid x$ for all $x \in S$. 

Basis: $6 \in S; 15 \in S$

Recursive: If $x, y \in S$ then $x + y \in S$
More Structural Induction

• Let \( R \) be given by...
  – Basis: \( 12 \in R; \ 15 \in R \)
  – Recursive: if \( x \in R \), then \( x + 6 \in R \) and \( x + 15 \in R \)

• Two base cases and two recursive cases, one existing element.

Claim: \( R \subseteq S \); i.e. every element of \( R \) is also in \( S \).

Proof needs structural induction using definition of \( R \) since statement is of the form \( \forall x \in R \ P(x) \).
Claim: Every element of $R$ is in $S$. ($R \subseteq S$)

1. Let $P(x)$ be “$x \in S$”. We prove that $P(x)$ is true for all $x \in R$ by structural induction.

2. Base Case: (12): $6 \in S$ so $6+6=12 \in S$ by definition of $S$, so $P(12)$
   (15): $15 \in S$, so $P(15)$ is also true

3. Ind. Hyp: Suppose that $P(x)$ is true for some arbitrary $x \in R$

4. Inductive Step: **Goal: Show $P(x+6)$ and $P(x+15)$**
   Since $P(x)$ holds, we have $x \in S$. Since $6 \in S$ from the recursive step of $S$, we get $x + 6 \in S$, so $P(x+6)$ is true, and since $15 \in S$ we get $x + 15 \in S$, so $P(x+15)$ is true.

5. Therefore $P(x)$ (i.e., $x \in S$) for all $x \in R$ by induction.

<table>
<thead>
<tr>
<th>Basis: 6 \in S; 15 \in S</th>
<th>Basis: 12 \in R; 15 \in R</th>
</tr>
</thead>
<tbody>
<tr>
<td>Recursive: if $x, y \in S$, then $x + y \in S$</td>
<td>Recursive: if $x \in R$, then $x + 6 \in R$ and $x + 15 \in R</td>
</tr>
</tbody>
</table>
Recursive Definitions

• Recursively defined functions and sets are our mathematical models of code and the data it uses
  – recursively defined sets can be translated into Java classes
  – recursively defined functions can be translated into Java functions
    some (but not all) can be written more cleanly as loops

• Can now do proofs about CS-specific objects
Lists of Integers

- **Basis:** \(\text{nil} \in \text{List}\)
- **Recursive step:**
  
  \[
  \text{if } L \in \text{List} \text{ and } a \in \mathbb{Z}, \\
  \text{then } a :: L \in \text{List}
  \]

Examples:
- \(\text{nil} \rightarrow []\)
- \(1 :: \text{nil} \rightarrow [1]\)
- \(1 :: 2 :: \text{nil} \rightarrow [1, 2]\)
- \(1 :: 2 :: 3 :: \text{nil} \rightarrow [1, 2, 3]\)
Functions on Recursively Defined Sets

Assume that the recursive definition of $S$ gives a unique way to construct every element of $S$.

We can define the values of a function $f$ on $S$ recursively as follows:

**Basis:** Define $f(u)$ for all specific elements $u$ of $S$ mentioned in the *Basis step*.

**Recursive Step:** Define $f(w)$ for each of the new elements $w$ constructed in terms of $f$ applied to each of the existing named elements mentioned in the *Recursive step*.
Functions on Lists

Length:

\[
\begin{align*}
\text{len}(\text{nil}) & := 0 \\
\text{len}(a :: L) & := \text{len}(L) + 1
\end{align*}
\]

for any \( L \in \text{List} \) and \( a \in \Sigma \)

Concatenation:

\[
\begin{align*}
\text{concat}(\text{nil}, R) & := R \\
\text{concat}(a :: L, R) & := a :: \text{concat}(L, R)
\end{align*}
\]

for any \( R \in \text{List} \) and \( L, R \in \text{List} \) and any \( a \in \mathbb{Z} \)

Basis: \( \text{nil} \in \text{List} \)

Recursive step: if \( L \in \text{List} \) and \( a \in \mathbb{Z} \), then \( a :: L \in \text{List} \)
Structural Induction

How to prove $\forall x \in S, P(x)$ is true:

Base Case: Show that $P(u)$ is true for all specific elements $u$ of $S$ mentioned in the Basis step.

Inductive Hypothesis: Assume that $P$ is true for some arbitrary values of each of the existing named elements mentioned in the Recursive step.

Inductive Step: Prove that $P(w)$ holds for each of the new elements $w$ constructed in the Recursive step using the named elements mentioned in the Inductive Hypothesis.

Conclude that $\forall x \in S, P(x)$.
Claim: \( \text{len}(\text{concat}(L, R)) = \text{len}(L) + \text{len}(R) \) for all \( L, R \in \text{List} \)

Let \( D(L) \) be:

\[ \text{len}(\text{concat}(L, R)) = \text{len}(L) + \text{len}(R) \text{ for all } R \in \text{List} \]

**Length:**

- \( \text{len}(\text{nil}) := 0 \)
- \( \text{len}(a :: L) := \text{len}(L) + 1 \)

**Concatenation:**

- \( \text{concat}(\text{nil}, R) := R \)
- \( \text{concat}(a :: L, R) := a :: \text{concat}(L, R) \)
Claim: \( \forall L, R \in \text{List} \) \( \operatorname{len} \left( \text{concat}(L, R) \right) = \operatorname{len}(L) + \operatorname{len}(R) \)

Let \( P(L) \) be “\( \operatorname{len} \left( \text{concat}(L, R) \right) = \operatorname{len}(L) + \operatorname{len}(R) \) for all \( R \in \text{List} \)”.

We prove \( P(L) \) for all \( L \in \text{List} \) by structural induction.

**Base Case:** \( \text{null} \)

\[\begin{align*}
\text{concat} \left( \text{null}, R \right) &= R \\
\text{LHS} &= \operatorname{len}(R) \\
\text{RHS} &= \operatorname{len}(\text{null}) + \operatorname{len}(R) \\
0 &= \operatorname{len}(R)
\end{align*}\]

**Length:**

- \( \operatorname{len}(\text{null}) := 0 \)
- \( \operatorname{len}(a :: L) := \operatorname{len}(L) + 1 \)

**Concatenation:**

- \( \text{concat}(\text{null}, R) := R \)
- \( \text{concat}(a :: L, R) := a :: \text{concat}(L, R) \)
Claim: \( \text{len}(\text{concat}(L, R)) = \text{len}(L) + \text{len}(R) \) \ for all \( L, R \in \text{List} \)

Let \( P(L) \) be “\( \text{len}(\text{concat}(L, R)) = \text{len}(L) + \text{len}(R) \) for all \( R \in \text{List} \)”.

We prove \( P(L) \) for all \( L \in \text{List} \) by structural induction.

Base Case (nil): Let \( R \in \text{List} \) be arbitrary. Then,

\[ \text{len}(\text{concat}(\text{nil}, R)) = \text{len}(\text{nil}) + \text{len}(R) = 0 + \text{len}(R) = \text{len}(R) \]

Length:
- \( \text{len}(\text{nil}) := 0 \)
- \( \text{len}(\text{a :: L}) := \text{len}(L) + 1 \)

Concatenation:
- \( \text{concat}(\text{nil}, R) := R \)
- \( \text{concat}(\text{a :: L}, R) := \text{a :: concat}(L, R) \)
**Claim:** \( \text{len}(\text{concat}(L, R)) = \text{len}(L) + \text{len}(R) \) \text{ for all } L, R \in \text{List} \\

Let \( P(L) \) be “\( \text{len}(\text{concat}(L, R)) = \text{len}(L) + \text{len}(R) \) \text{ for all } R \in \text{List} \)”. We prove \( P(L) \) for all \( L \in \text{List} \) by structural induction.

**Base Case** (nil): Let \( R \in \text{List} \) be arbitrary. Then,

\[
\text{len}(\text{concat}(\text{nil}, R)) = \text{len}(R) \quad \text{def of concat}
\]
\[
= 0 + \text{len}(R) \quad \text{def of len}
\]
\[
= \text{len}(\text{nil}) + \text{len}(R) \quad \text{def of len}
\]

Since \( R \) was arbitrary, \( P(\text{nil}) \) holds.
**Claim:** \( \text{len}(\text{concat}(L, R)) = \text{len}(L) + \text{len}(R) \) for all \( L, R \in \text{List} \)

Let \( P(L) \) be “\( \text{len}(\text{concat}(L, R)) = \text{len}(L) + \text{len}(R) \) for all \( R \in \text{List} \)”. We prove \( P(L) \) for all \( L \in \text{List} \) by structural induction.

**Base Case** (nil): Let \( R \in \text{List} \) be arbitrary. Then, \( \text{len}(\text{concat}(\text{nil}, R)) = \text{len}(R) = 0 + \text{len}(R) = \text{len}(\text{nil}) + \text{len}(R) \), showing \( P(\text{nil}) \).

**Inductive Hypothesis:** Assume that \( P(L) \) is true for some arbitrary \( L \in \text{List} \), i.e., \( \text{len}(\text{concat}(L, R)) = \text{len}(L) + \text{len}(R) \) for all \( R \in \text{List} \).
Let $P(L)$ be “$\text{len}(\text{concat}(L, R)) = \text{len}(L) + \text{len}(R)$ for all $R \in \text{List}$”. We prove $P(L)$ for all $L \in \text{List}$ by structural induction.

**Base Case** ($\texttt{nil}$): Let $R \in \text{List}$ be arbitrary. Then, $\text{len}(\text{concat}(\texttt{nil}, R)) = \text{len}(R) = 0 + \text{len}(R) = \text{len}(\texttt{nil}) + \text{len}(R)$, showing $P(\texttt{nil})$.

**Inductive Hypothesis:** Assume that $P(L)$ is true for some arbitrary $L \in \text{List}$, i.e., $\text{len}(\text{concat}(L, R)) = \text{len}(L) + \text{len}(R)$ for all $R \in \text{List}$.

**Inductive Step:** Goal: Show that $P(a :: L)$ is true.

$$
\text{len}(\text{concat}(a :: L, R)) = \text{len}(a :: \text{concat}(L, R)) \\
= \text{len}(\text{concat}(L, R)) + 1 \\
= \text{len}(L) + \text{len}(R) + 1 \quad \text{by IH} \\
= \text{len}(a :: L) + \text{len}(R)
$$
**Claim:** \( \text{len}(\text{concat}(L, R)) = \text{len}(L) + \text{len}(R) \) for all \( L, R \in \text{List} \)

Let \( P(L) \) be “\( \text{len}(\text{concat}(L, R)) = \text{len}(L) + \text{len}(R) \) for all \( R \in \text{List} \)” . We prove \( P(L) \) for all \( L \in \text{List} \) by structural induction.

**Base Case** (nil): Let \( R \in \text{List} \) be arbitrary. Then, \( \text{len}(\text{concat}(\text{nil}, R)) = \text{len}(R) = 0 + \text{len}(R) = \text{len}(\text{nil}) + \text{len}(R) \), showing \( P(\text{nil}) \).

**Inductive Hypothesis:** Assume that \( P(L) \) is true for some arbitrary \( L \in \text{List} \), i.e., \( \text{len}(\text{concat}(L, R)) = \text{len}(L) + \text{len}(R) \) for all \( R \in \text{List} \).

**Inductive Step:** Goal: Show that \( P(a :: L) \) is true

Let \( R \in \text{List} \) be arbitrary. Then,

**Length:**
\[
\begin{align*}
\text{len}(\text{nil}) & := 0 \\
\text{len}(a :: L) & := \text{len}(L) + 1
\end{align*}
\]

**Concatenation:**
\[
\begin{align*}
\text{concat}(\text{nil}, R) & := R \\
\text{concat}(a :: L, R) & := a :: \text{concat}(L, R)
\end{align*}
\]
**Claim:** \( \text{len(concat}(L, R)) = \text{len}(L) + \text{len}(R) \) for all \( L, R \in \text{List} \).

Let \( P(L) \) be “\( \text{len(concat}(L, R)) = \text{len}(L) + \text{len}(R) \) for all \( R \in \text{List} \)”.
We prove \( P(L) \) for all \( L \in \text{List} \) by structural induction.

**Base Case** (\( \text{nil} \)): Let \( R \in \text{List} \) be arbitrary. Then, \( \text{len(concat}(\text{nil}, R)) = \text{len}(R) = 0 + \text{len}(R) = \text{len}(\text{nil}) + \text{len}(R) \), showing \( P(\text{nil}) \).

**Inductive Hypothesis**: Assume that \( P(L) \) is true for some arbitrary \( L \in \text{List} \), i.e., \( \text{len(concat}(L, R)) = \text{len}(L) + \text{len}(R) \) for all \( R \in \text{List} \).

**Inductive Step**: **Goal: Show that \( P(a :: L) \) is true**

Let \( R \in \text{List} \) be arbitrary. Then, we can calculate
\[
\text{len(concat}(a :: L, R)) = 1 + \text{len(concat}(L, R)) = 1 + \text{len}(L) + \text{len}(R) = \text{len}(a :: L) + \text{len}(R)
\]
Claim: \( \text{len}(\text{concat}(L, R)) = \text{len}(L) + \text{len}(R) \) for all \( L, R \in \text{List} \)

Let \( P(L) \) be “\( \text{len}(\text{concat}(L, R)) = \text{len}(L) + \text{len}(R) \) for all \( R \in \text{List} \)”.
We prove \( P(L) \) for all \( L \in \text{List} \) by structural induction.

Base Case (nil): Let \( R \in \text{List} \) be arbitrary. Then, \( \text{len}(\text{concat}(\text{nil}, R)) = \text{len}(R) = 0 + \text{len}(R) = \text{len}(\text{nil}) + \text{len}(R) \), showing \( P(\text{nil}) \).

Inductive Hypothesis: Assume that \( P(L) \) is true for some arbitrary \( L \in \text{List} \), i.e., \( \text{len}(\text{concat}(L, R)) = \text{len}(L) + \text{len}(R) \) for all \( R \in \text{List} \).

Inductive Step: Goal: Show that \( P(a :: L) \) is true

Let \( R \in \text{List} \) be arbitrary. Then, we can calculate

\[
\text{len}(\text{concat}(a :: L, R)) = \text{len}(a :: \text{concat}(L, R)) = 1 + \text{len}(\text{concat}(L, R)) = 1 + \text{len}(L) + \text{len}(R) = \text{len}(a :: L) + \text{len}(R)
\]

Since \( R \) was arbitrary, we have shown \( P(a :: L) \).

By induction, we have shown the claim holds for all \( L \in \text{List} \).
Rooted Binary Trees

- **Basis:** is a rooted binary tree

- **Recursive step:**

  If $T_1$ and $T_2$ are rooted binary trees,

  then $T_1$ and $T_2$ also is a rooted binary tree.
Defining Functions on Rooted Binary Trees

- size(•) := 1

- size(T₁, T₂) := 1 + size(T₁) + size(T₂)

- height(•) := 0

- height(T₁, T₂) := 1 + max{height(T₁), height(T₂)}
Last time: Structural Induction

How to prove $\forall x \in S, P(x)$ is true:

**Base Case:** Show that $P(u)$ is true for all specific elements $u$ of $S$ mentioned in the **Basis step**

**Inductive Hypothesis:** Assume that $P$ is true for some arbitrary values of each of the existing named elements mentioned in the **Recursive step**

**Inductive Step:** Prove that $P(w)$ holds for each of the new elements $w$ constructed in the **Recursive step** using the named elements mentioned in the Inductive Hypothesis

**Conclude** that $\forall x \in S, P(x)$
Claim: For every rooted binary tree $T$, $\text{size}(T) \leq 2^{\text{height}(T)} + 1 - 1$
Claim: For every rooted binary tree $T$, $\text{size}(T) \leq 2^{\text{height}(T)} + 1 - 1$

1. Let $P(T)$ be “$\text{size}(T) \leq 2^{\text{height}(T)} + 1 - 1$”. We prove $P(T)$ for all rooted binary trees $T$ by structural induction.

\begin{align*}
\text{size}(\bullet) & \coloneqq 1 \\
\text{size}(T_1 T_2) & \coloneqq 1 + \text{size}(T_1) + \text{size}(T_2) \\
\text{height}(\bullet) & \coloneqq 0 \\
\text{height}(T_1 T_2) & \coloneqq 1 + \max\{\text{height}(T_1), \text{height}(T_2)\}
\end{align*}
**Claim:** For every rooted binary tree \( T \), \( \text{size}(T) \leq 2^{\text{height}(T)} + 1 - 1 \)

1. Let \( P(T) \) be “\( \text{size}(T) \leq 2^{\text{height}(T)} + 1 - 1 \)”.
   We prove \( P(T) \) for all rooted binary trees \( T \) by structural induction.

2. Base Case: \( \text{size}(\bullet) = 1 \), \( \text{height}(\bullet) = 0 \), and \( 2^{0+1} - 1 = 2^1 - 1 = 1 \) so \( P(\bullet) \) is true.
Claim: For every rooted binary tree $T$, $\text{size}(T) \leq 2^{\text{height}(T)} + 1 - 1$

1. Let $P(T)$ be “$\text{size}(T) \leq 2^{\text{height}(T)} + 1 - 1$”. We prove $P(T)$ for all rooted binary trees $T$ by structural induction.

2. Base Case: $\text{size}(\bullet) = 1$, $\text{height}(\bullet) = 0$, and $2^{0+1} - 1 = 2^1 - 1 = 1$ so $P(\bullet)$ is true.

3. Inductive Hypothesis: Suppose that $P(T_1)$ and $P(T_2)$ are true for some rooted binary trees $T_1$ and $T_2$, i.e., $\text{size}(T_k) \leq 2^{\text{height}(T_k)} + 1 - 1$ for $k = 1, 2$

4. Inductive Step: Goal: Prove $P(\text{rooted binary tree})$. 

By defn, $\text{size}(\text{rooted binary tree}) = 1 + \text{size}(T_1) + \text{size}(T_2) \leq 1 + 2^{\text{height}(T_1)} + 1 - 1 + 2^{\text{height}(T_2)} + 1 - 1$ by IH for $T_1$ and $T_2$.

$\leq 2^{\text{height}(T_1)} + 1 + 2^{\text{height}(T_2)} + 1 - 1 \leq 2^{\text{height}(\text{rooted binary tree})} + 1 - 1$ which is what we wanted to show.

5. So, the $P(T)$ is true for all rooted bin. trees by structural induction.
**Claim:** For every rooted binary tree $T$, $\text{size}(T) \leq 2^{\text{height}(T)} + 1 - 1$

1. Let $P(T)$ be “$\text{size}(T) \leq 2^{\text{height}(T)} + 1 - 1$”. We prove $P(T)$ for all rooted binary trees $T$ by structural induction.

2. **Base Case:** $\text{size}(\bullet) = 1$, $\text{height}(\bullet) = 0$, and $2^{0+1-1} = 2^1 - 1 = 1$ so $P(\bullet)$ is true.

3. **Inductive Hypothesis:** Suppose that $P(T_1)$ and $P(T_2)$ are true for some rooted binary trees $T_1$ and $T_2$, i.e., $\text{size}(T_k) \leq 2^{\text{height}(T_k)} + 1 - 1$ for $k=1,2$.

4. **Inductive Step:**
   
   **Goal:** Prove $P(T)$. 

   \[
   \text{size}(\begin{array}{c}
   \text{size}(T_1) \\
   \text{size}(T_2)
   \end{array}) \leq 2^{\text{height}(T) + 1} - 1
   \]
**Claim:** For every rooted binary tree $T$, $\text{size}(T) \leq 2^{\text{height}(T)} + 1 - 1$

1. Let $P(T)$ be “$\text{size}(T) \leq 2^{\text{height}(T)}+1-1$”. We prove $P(T)$ for all rooted binary trees $T$ by structural induction.

2. **Base Case:** $\text{size}(\bullet)=1$, $\text{height}(\bullet)=0$, and $2^{0+1}-1=2^1-1=1$ so $P(\bullet)$ is true.

3. **Inductive Hypothesis:** Suppose that $P(T_1)$ and $P(T_2)$ are true for some rooted binary trees $T_1$ and $T_2$, i.e., $\text{size}(T_k) \leq 2^{\text{height}(T_k)} + 1 - 1$ for $k=1,2$.

4. **Inductive Step:**

   **Goal:** Prove $P(\text{ } )$.

   By def, $\text{size}(\text{ } ) = 1 + \text{size}(T_1) + \text{size}(T_2)$

   $\leq 1 + 2^{\text{height}(T_1)} + 1 - 1 + 2^{\text{height}(T_2)} + 1 - 1$

   by IH for $T_1$ and $T_2$

   $\leq 2^{\text{height}(T_1)} + 2^{\text{height}(T_2)} + 1 - 1$

   $\leq 2 \cdot \max(2^{\text{height}(T_1)} + 1, 2^{\text{height}(T_2)} + 1) - 1$

   $\leq 2(2^{\max(\text{height}(T_1), \text{height}(T_2))} + 1) - 1$

   $\leq 2(2^{\text{height}(\text{ } )} - 1 \leq 2^{\text{height}(\text{ } )} + 1 - 1$

   which is what we wanted to show.

5. So, the $P(T)$ is true for all rooted binary trees by structural induction.