Foundations of Computer Science
Lecture 8
Proofs with Recursive Objects

Structural Induction: Induction on Recursively Defined Objects
Proving an object is *not* in a recursive set
Examples: sets, sequences, trees

Two Types of Questions About a Recursive Set

\[ \mathcal{A} = \{0, 4, 8, 12, 16, \ldots\} \]

Recursive definition of \( \mathcal{A} \).
1. \( 0 \in \mathcal{A} \).
2. \( x \in \mathcal{A} \rightarrow x + 4 \in \mathcal{A} \).

(i) What is in \( \mathcal{A} \)? Is some feature common to every element of \( \mathcal{A} \)? Is everything in \( \mathcal{A} \) even?

\[ x \in \mathcal{A} \rightarrow x \text{ is even} \quad (T) \]

(ii) Is everything with some property in \( \mathcal{A} \)? Is every even number in \( \mathcal{A} \)?

\[ x \text{ is even} \rightarrow x \in \mathcal{A} \quad (F) \]

Very very different statements!

Every leopard has 4 legs.
Everything with 4 legs is a leopard?

Structural induction shows every member of a recursive set has a property, question (i).

Last Time

- Recursion.
- Recurrences are recursive functions on \( \mathbb{N} \).
- Recursive programs.
- Recursive sets.
- Rooted binary trees (RBT).
Orks and blue Eyes

- The first two Orks had blue eyes.
- When two Orks mate, if they both have blue eyes, then the child has blue eyes.

Do all Orks have blue eyes?

When could a green-eyed ork have arisen?

**Structural Induction**
1. The ancestors have a trait.
2. The trait is passed on from parents to children.

Conclusion: Everyone today has that trait.

---

**Strings in $\mathcal{M}$ are Balanced**

Balanced means the number of opening and closing parentheses are equal.

The constructor, $x, y \in \mathcal{M} \rightarrow [x] \cdot y \in \mathcal{M}$, adds one opening and closing parenthesis.

If the “parent” strings $x$ and $y$ are balanced, then the child $[x] \cdot y$ is balanced.

(Orks inherit blue eyes. Here, parents pass along balance to the children.)

Just as all Orks will have blue eyes, all strings in $\mathcal{M}$ will be balanced.

---

**Proof: Strings in $\mathcal{M}$ are Balanced**

$\mathcal{M} = \{s_1, s_2, s_3, s_4, s_5, \ldots, s_n, \ldots\}$

$P(n)$: string $s_n$ is balanced, i.e., the number of ‘[’ equals the number of ‘]’.

**Proof.** Strong induction on $n$.
1. **[Base case]** The base case is $s_1 = \varepsilon$ which is clearly balanced, so $P(1)$ is T.
2. **[Induction step]** Show $P(1) \land \cdots \land P(n) \rightarrow P(n+1)$ (direct proof).
   
   Assume $P(1) \land P(2) \land \cdots \land P(n)$: $s_1, s_2, \ldots, s_n$ are all balanced.

   Show $P(n+1)$: $s_{n+1}$ is balanced.

   $s_{n+1}$ is the child of two earlier strings: $s_{n+1} = [s_i] \cdot s_i$ (constructor rule):

   $s_i, s_i$ appeared earlier than $s_{n+1}$, so $s_i$ and $s_i$ are balanced (induction hypothesis).

   Therefore $s_{n+1}$ is balanced (because you add one opening and closing parenthesis).

3. By induction, $P(n)$ is T for $n \geq 1$.

**Question.** Is every balanced string in $\mathcal{M}$?

**Exercise.** Prove that $[[]] \notin \mathcal{M}$.
Structural Induction

Strong induction with recursively defined sets is called **structural induction**.

Let \( S \) be a recursive set. This means you have:
- **Bases cases** \( s_1, \ldots, s_k \) that are in \( S \).
- **Constructor rules** that use elements in \( S \) to create a new element of \( S \).

Let \( P(s) \) be a property defined for any element \( s \in S \). To show \( P(s) \) for every element in \( S \), you must show:
1. **[Base cases]** \( P(s_1), P(s_2), \ldots, P(s_k) \) are true.
2. **[Induction step]** For every constructor rule, show:
   - If \( P \) is true for the parents, then \( P \) is true for children.
3. By structural induction, conclude that \( P(s) \) is true for all \( s \in S \). ■

- **MUST** show for every base case.
- **MUST** show for every constructor rule.
- Structural induction can be used with any recursive set.

Structural Induction on \( \mathbb{N} \)

\( \mathbb{N} = \{1, 2, 3, \ldots\} \) is a recursively defined set,
- \( 1 \in \mathbb{N} \).
- \( x \in \mathbb{N} \rightarrow x + 1 \in \mathbb{N} \).

Consider any property of the natural numbers, for example
\[ P(n) : 5^n - 1 \text{ is divisible by 4}. \]

Structural induction to prove \( P(n) \) holds for every \( n \in \mathbb{N} \):
1. **[Prove for all base cases]** Only one base case \( P(1) \).
2. **[Prove every constructor rule preserves \( P(n) \)]** Only one constructor:
   - If \( P \) is true for \( x \) (the parent), then \( P \) is true for \( x + 1 \) (the child).
3. By structural induction, \( P(n) \) is true for all \( n \in \mathbb{N} \). ■

That’s just ordinary induction! ☺

Every String in \( \mathcal{M} \) is Matched

Going from left to right:
\[ [ [ ] ] [ ] \]
- Opening: 3
- Closing: 3

Opening is always at least closing: parentheses are arithmetically matched. **Important Exercise.** Prove this by structural induction.

Key step is to show that constructor preserves “matchedness”.

**Hard Exercise.** Prove this (see Exercise 8.3).

Palindromes \( \mathcal{P} \)

“Was it a rat I saw”

\[ (0110)^n = 00110 \quad \text{not a palindrome} \]
\[ (0110)^n = 0110 \quad \text{palindrome} \]

Recursive definition of palindromes \( \mathcal{P} \):
- There are three base cases: \( \epsilon \in \mathcal{P} \), \( 0 \in \mathcal{P} \), \( 1 \in \mathcal{P} \).
- There are two constructor rules:
  1. \( x \in \mathcal{P} \rightarrow 0 \bullet x \bullet 0 \in \mathcal{P} \);
  2. \( x \in \mathcal{P} \rightarrow 1 \bullet x \bullet 1 \in \mathcal{P} \).

Constructor rules preserves palindromicity:
- \( (0 \bullet 0110 \bullet 0)^n = 001100 \)
- \( (1 \bullet 0110 \bullet 1)^n = 101101 \)

Therefore, we can prove by structural induction that all strings in \( \mathcal{P} \) are palindromes.

**Hard Exercise.** Prove that all palindromes are in \( \mathcal{P} \) (Exercise 8.7).

Creator: Malik Magdon-Ismail
Proofs with Recursive Objects: 12/15
Structural Induction on \( \mathbb{N} \)
Fact known to all kindergartners: $((1 + 1 + 1) \times (1 + 1 + 1 + 1 + 1)) = 15,$ value $^2((1 + 1 + 1) \times (1 + 1 + 1 + 1 + 1)) = 15.$

A recursive set of well formed arithmetic expression strings $A_{\text{eco}}$:

- One base case: $1 \in A_{\text{eco}}$.
- There are two constructor rules: (i) $x \in A_{\text{eco}} \rightarrow (x + 1 + 1) \in A_{\text{eco}}$; (ii) $x, y \in A_{\text{eco}} \rightarrow (x \times y) \in A_{\text{eco}}$.

$$1 \rightarrow (1 + 1 + 1) \rightarrow ((1 + 1 + 1) + 1 + 1)$$
$$((1 \times 1) + (1 + 1 + 1))$$

The constructors add 2 to the parent or multiply the parents. If the parents have odd value, then so does the child. Constructors preserve “oddness” → all strings in $A_{\text{eco}}$ have odd value.

**Rooted Binary Tree with $n \geq 1$ Vertices Have $n - 1$ Edges**

- The empty tree $\varepsilon$ is an RBT.
- Disjoint RBTs $T_1, T_2$ give a new RBT by linking their roots to a new root.

$P(T)$ : if $T$ is a rooted binary tree with $n \geq 1$ vertices, then $T$ has $n - 1$ links.

1. **[Base case]** $P(\varepsilon)$ is vacuously $\top$ because $\varepsilon$ is not a tree with $n \geq 1$ vertices.
2. **[Induction step]** Consider the constructors with parent RBTs $T_1$ and $T_2$.
   - Parents: $T_1$ with $n_1$ vertices and $\ell_1$ edges and $T_2$ with $n_2$ vertices and $\ell_2$ edges.
   - Child: $T$ with $n$ vertices and $\ell$ edges.

   **Case 1:** $T_1 = T_2 = \varepsilon$. Child is a single node with $n = 1$, $\ell = 0$, and $\ell = n - 1. \checkmark$

   **Case 2:** $T_1 = \varepsilon$, $T_2 \neq \varepsilon$. The child one more node and one more link, $n = n_2 + 1$ and $\ell = \ell_2 + 1 \implies n_2 - 1 + 1 = n_2 = n - 1. \checkmark$

   **Case 3:** $T_1 \neq \varepsilon$, $T_2 = \varepsilon$. (Similar to case 2) $n = n_1 + 1$ and $\ell = \ell_1 + 1 \implies n_1 - 1 + 1 = n_1 = n - 1. \checkmark$

   **Case 4:** $T_1 \neq \varepsilon$, $T_2 \neq \varepsilon$. Now, $n = n_1 + n_2 + 1$ and there are two new links, so $\ell = \ell_1 + \ell_2 + 2 \implies n_1 - 1 + n_2 - 1 + 2 = n_1 + n_2 = n - 1. \checkmark$

   Constructor always preserves property $P$.

3. By structural induction, $P(T)$ is true $\forall T \in \text{RBT}$. 

**Checklist for Structural Induction**

- **Analogy:** if the first ancestors had blue eyes, and blue eyes are inherited from one generation to the next, then all of society will have blue eyes.
- You have a recursively defined set $S$.
- You want to prove a property $P$ for all members of $S$.
- Does the property $P$ hold for the base cases?
- Is the property $P$ preserved by all the constructor rules?
- Structural induction is not how to prove all objects with property $P$ are in $S$.  

**Rooted Binary Tree with**

$$\varepsilon \Rightarrow$$

$1 \Rightarrow$