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

Last Time

1. Recursion.
2. Recurrences are recursive functions on $\mathbb{N}$.
3. Recursive programs.
4. Recursive sets.
5. Rooted binary trees (RBT).

Today: Proofs with Recursive Objects

1. Two Types of Questions About Recursive Sets
2. Matched Parentheses
3. Structural Induction
   - $\mathbb{N}$
   - Palindromes
   - Arithmetic Expressions
4. Rooted Binary Trees (RBT)

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. $x \in \mathcal{A} \to x + 4 \in \mathcal{A}$.

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

$$x \in \mathcal{A} \ \to \ x \text{ is even} \qquad \text{(T)}$$

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

$$x \text{ is even} \ \to \ x \in \mathcal{A} \qquad \text{(F)}$$

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).

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

- The ancestors have a trait.
- The trait is passed on from parents to children.

Conclusion: Everyone today has that trait.

Matched Parentheses $\mathcal{M}$

Recursive definition of $\mathcal{M}$.

1. $\varepsilon \in \mathcal{M}$. **[basis]**
2. $x, y \in \mathcal{M} \to [x] \bullet y \in \mathcal{M}$. **[constructor]**

The strings in $\mathcal{M}$ are the matched (in the arithmetic sense) parentheses. For example:

$$
\begin{array}{ll}
[\,] & (\text{set } x = \varepsilon,\ y = \varepsilon \text{ to get } [\varepsilon]\varepsilon = [\,]) \\
[[\,]] & (\text{set } x = [\,],\ y = \varepsilon) \\
[\,][\,] & (\text{set } x = \varepsilon,\ y = [\,])
\end{array}
$$

Let's list the strings in $\mathcal{M}$ as they are created,

$$
\mathcal{M} = \{\ \underset{\substack{\uparrow \\ s_1}}{\varepsilon},\ \underset{\substack{\uparrow \\ s_2}}{[\,]},\ \underset{\substack{\uparrow \\ s_3}}{[[\,]]},\ \underset{\substack{\uparrow \\ s_4}}{[\,][\,]},\ \underset{\substack{\uparrow \\ s_5}}{[[\,]][\,]},\ \ldots,\ s_n,\ \ldots\ \}
$$

To get s_n, we apply the constructor to two prior (not necessarily distinct) strings.

Strings in $\mathcal{M}$ are Balanced

Balanced means the number of opening and closing parentheses are equal

The constructor,

$$x, y \in \mathcal{M} \to [x] \bullet y \in \mathcal{M}$$

adds one opening and closing parenthesis.

If the "parent" strings x and y are balanced, then the child $[x] \bullet y$ 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 $P(1) \wedge \cdots \wedge P(n) \to P(n+1)$ (direct proof).

Assume $P(1) \wedge P(2) \wedge \cdots \wedge P(n)$*:* $s_1, \ldots, s_n$ are all balanced.

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

s_{n+1} is the child of two *earlier* strings: $s_{n+1} = [s_k] \bullet s_\ell$ (constructor rule)

s_k, s_ℓ appeared *earlier* than s_{n+1}, so s_k and s_ℓ 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 $\forall n \geq 1$. ■

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 $\mathcal{S}$ be a recursive set. This means you have:

1. Bases cases $s_1, \ldots, s_k$ that are in $\mathcal{S}$.
2. Constructor rules that use elements in $\mathcal{S}$ to create a new element of $\mathcal{S}$.

Let $P(s)$ be a property defined for any element $s \in \mathcal{S}$. To show $P(s)$ for every element in $\mathcal{S}$, you must show:

1: **[Base cases]** $P(s_1), P(s_2), \ldots, P(s_k)$ are T.

2: **[Induction step]** For *every* constructor rule, show:

IF P is T for the parents, THEN P is T for children

3: By structural induction, conclude that $P(s)$ is T for all $s \in \mathcal{S}$. ■

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

Every String in $\mathcal{M}$ is Matched

[[ ]] [ ]

opening:3 closing:3

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".

Question. Is every string of matched parentheses in $\mathcal{M}$?

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

Structural Induction on $\mathbb{N}$

$\mathbb{N} = \{1, 2, 3, \ldots\}$ is a recursively defined set,

1. $1 \in \mathbb{N}$.
2. $x \in \mathbb{N} \to x + 1 \in \mathbb{N}$.

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 T for x (the parent), THEN P is T for $x + 1$ (the child).

3: By structural induction, $P(n)$ is T $\forall n \in \mathbb{N}$. ■

That's just ordinary induction! 🙂

Palindromes $\mathcal{P}$

"Was it a rat I saw"

$$
\begin{array}{rcll}
(01100)^{\mathrm{R}} & = & 00110 & \text{not a palindrome} \\
(0110)^{\mathrm{R}} & = & 0110 & \text{palindrome}
\end{array}
$$

Recursive definition of palindromes $\mathcal{P}$

1. There are three base cases: $\varepsilon \in \mathcal{P},\ 0 \in \mathcal{P},\ 1 \in \mathcal{P}$.
2. There are two constructor rules: (i) $x \in \mathcal{P} \to 0\bullet x\bullet 0 \in \mathcal{P}$;
   (ii) $x \in \mathcal{P} \to 1\bullet x\bullet 1 \in \mathcal{P}$.

Constructor *rules* preserves palindromicity:

(i) $(0\bullet \underbrace{0110}_{x} \bullet 0)^{\mathrm{R}} = 001100$

(ii) $(1\bullet 0110\bullet 1)^{\mathrm{R}} = 101101$

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).

Arithmetic Expressions

Fact known to all kindergartners: $((1+1+1)\times(1+1+1+1+1)) = 15$,

$$\texttt{value}\big(((1+1+1)\times(1+1+1+1+1))\big) = 15,$$

A recursive set of well formed arithmetic expression strings $\mathcal{A}_{\text{ODD}}$:

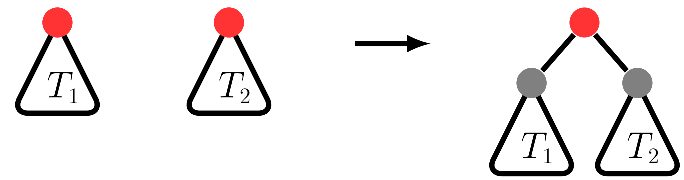
1. One base case: $1 \in A_{\text{ODD}}$.
2. There are two constructor rules: (i) $x \in \mathcal{A}_{\text{ODD}} \to (x+1+1) \in \mathcal{A}_{\text{ODD}}$;
   (ii) $x, y \in \mathcal{A}_{\text{ODD}} \to (x \times y) \in \mathcal{A}_{\text{ODD}}$.

$$\begin{array}{lllll} 1 \to & (1+1+1) & \to & ((1+1+1)+1+1) \\ & (1\times 1) & & ((1\times 1)+1+1) \\ & & & \vdots \end{array}$$

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" $\to$ all strings in $\mathcal{A}_{\text{ODD}}$ have odd value.

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

1. The empty tree ε is an RBT.
2. 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 T because ε 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 ℓ_1 edges and T_2 with n_2 vertices and ℓ_2 edges.
Child: T with n vertices and ℓ edges.

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

Case 2: $T_1 = \varepsilon; T_2 \neq \varepsilon$. The child one more node and one more link, $n = n_2 + 1$ and

$$\ell = \ell_2 + 1 \stackrel{\text{IH}}{=} n_2 - 1 + 1 = n_2 = n - 1.✓$$

Case 3: $T_1 \neq \varepsilon; T_2 = \varepsilon$. (Similar to case 2.) $n = n_1 + 1$ and $\ell = \ell_1 + 1 \stackrel{\text{IH}}{=} n_1 - 1 + 1 = n_1 = n - 1$.✓

Case 4: $T_1 \neq \varepsilon; T_2 \neq \varepsilon$. Now, $n = n_1 + n_2 + 1$ and there are two new links, so

$$\ell = \ell_1 + \ell_2 + 2 \stackrel{\text{IH}}{=} n_1 - 1 + n_2 - 1 + 2 = n_1 + n_2 = n - 1.✓$$

Constructor always preserves property P.

3: By structural induction, $P(T)$ is true $\forall T \in$ 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 $\mathcal{S}$.
- ✓ You want to prove a property P for all members of $\mathcal{S}$.
- ✓ 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 $\mathcal{S}$.