

Proofs with Recursion

Reading

- Malik Magdon-Ismael. Discrete Mathematics and Computing.
 - Chapter 8

Overview

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

Two Types of Questions About a Recursive Set

- Recursive definition of a set \mathcal{A} :

$$0 \in \mathcal{A}$$

$$x \in \mathcal{A} \rightarrow x + 4 \in \mathcal{A}$$

- What is \mathcal{A} ?

$$\mathcal{A} = \{0, 4, 8, 12, 16, \dots\}$$

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

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

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

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

- **Very, very** different statements!
 - Every leopard has 4 legs
 - Is everything with 4 legs a leopard?

Structural Induction

- Structural induction shows every member of a recursive set has a property, question (i)
- Consider the evolution of orcs
 - (Because computer scientists have nothing better to do)
 - The first two orcs had blue eyes
 - When two orcs mate, if they both have blue eyes, then the child has blue eyes
 - Do all orcs have blue eyes?
 - When could a green-eyed orc 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}

$$\varepsilon \in \mathcal{M} \quad \text{[basis]}$$

$$x, y \in \mathcal{M} \rightarrow [x] \bullet y \in \mathcal{M} \quad \text{[constructor]}$$

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

$$[] \quad (\text{set } x = \varepsilon, y = \varepsilon \text{ to get } [\varepsilon]\varepsilon = [])$$

$$[][] \quad (\text{set } x = [], y = \varepsilon)$$

$$[] [] \quad (\text{set } x = \varepsilon, y = [])$$

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

$$\mathcal{M} = \{\varepsilon, [], [], [], [], [], \dots, s_n, \dots\}$$

- 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,

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

- adds one opening and closing parenthesis
- If the “parent” strings x and y are balanced, then the child $[x] \bullet y$ is balanced.
- (Orcs inherit blue eyes. Here, parents pass along balance to the children.)
- Just as all orcs will have blue eyes, all strings in \mathcal{M} will be balanced.

Proof: Strings in \mathcal{M} are Balanced

$$\begin{aligned}\mathcal{M} &= \{\varepsilon, [], [()], [()] [], \dots, s_n, \dots\} \\ &= \{s_1, s_2, s_3, \dots, s_n, \dots\}\end{aligned}$$

- What is the $P(n)$?
 - $P(n)$: string s_n is balanced, i.e., the number of '[' equals the number of ']'
- *Proof.* [Strong induction on n]
 1. [Base case] $P(1)$ claims that $s_1 = \varepsilon$ is balanced. True.
 2. [Induction step] Show that $P(1) \wedge \dots \wedge P(n) \rightarrow P(n + 1)$. Direct proof.
 - Assume $P(1) \wedge \dots \wedge P(n)$: s_1, \dots, s_n are all balanced
 - Show $P(n + 1)$: s_{n+1} is balanced
 - s_{n+1} is the child of two *earlier* strings: $s_{n+1} = [s_k] \bullet s_l$ (**constructor rule**)
 - s_k, s_l appeared earlier than s_{n+1} , so they are balanced (induction hypothesis)
 - Therefore, s_{n+1} is balanced (you add parentheses around a balanced string)
 3. By induction, $P(n)$ is T $\forall n \geq 1$

Proof: Strings in \mathcal{M} are Balanced, cont'd

$$\mathcal{M} = \{s_1, s_2, s_3, \dots, s_n, \dots\}$$

- $P(n)$: string s_n is balanced, i.e., the number of '[' equals the number of ']'
 - But is that the only property?
 - Every '[' is eventually followed by a corresponding ']'
- **Question.** Is every balanced string in \mathcal{M} ?
 - What about][?
- **Exercise.** Prove that $[[] \notin \mathcal{M}$

Structural Induction Overview

- Strong induction with recursively defined sets is called *structural induction*
- Let \mathcal{S} be a recursive set. This means you have:
 - Base cases s_1, \dots, s_k that are in \mathcal{S}
 - 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**] $P(s_1), P(s_2), \dots, P(s_k)$ 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 Opening Parenthesis in \mathcal{M} is Matched

- Example string in \mathcal{M} : $[[[]][[]]$
 - Three opening and three closing parentheses
- Going from left to right:
 - $[$, opening = 1, closing = 0
 - $[$, opening = 2, closing = 0
 - $]$, opening = 2, closing = 1
 - $]$, opening = 2, closing = 2
 - $[$, opening = 3, closing = 2
 - $]$, 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}

- $\mathbb{N} = \{1, 2, 3, \dots\}$ 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 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}

- Here's a nerdy palindrome: "Was it a rat I saw"
 - The same sequence of letters forwards and backwards
- Binary sequences

$$(01100)^R = 00110 \quad (\text{not a palindrome})$$

$$(0110)^R = 0110 \quad (\text{a palindrome})$$

- Recursive definition of palindromes \mathcal{P}
 - There are three base cases: $\varepsilon \in \mathcal{P}, 0 \in \mathcal{P}, 1 \in \mathcal{P}$
 - There are two constructor rules: (i) $x \in \mathcal{P} \rightarrow 0 \cdot x \cdot 0 \in \mathcal{P}$;
(ii) $x \in \mathcal{P} \rightarrow 1 \cdot x \cdot 1 \in \mathcal{P}$

- Constructor *rules* preserves palindromicity:

$$(0 \cdot 0110 \cdot 0)^R = 001100$$

$$(1 \cdot 0110 \cdot 1)^R = 101101$$

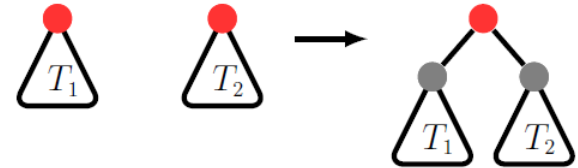
- Therefore, we can prove by structural induction that all strings in \mathcal{P} are palindromes
- **Hard Exercise.** Prove that all palindromes are in \mathcal{P} (Exercise 8.7).

Arithmetic Expressions

- Fact known to all first-graders: $((1+1+1) \times (1+1+1+1+1))=15$
 - i.e., $\text{value}((1+1+1) \times (1+1+1+1+1))=15$
- A recursive set of well formed arithmetic expression strings \mathcal{A}_{ODD} :
 - One base case: $1 \in \mathcal{A}_{ODD}$
 - There are two constructor rules: (i) $x \in \mathcal{A}_{ODD} \rightarrow (x + 1 + 1) \in \mathcal{A}_{ODD}$;
(ii) $x, y \in \mathcal{A}_{ODD} \rightarrow (x \times y) \in \mathcal{A}_{ODD}$
 - For example,
$$\begin{array}{ccc} 1 & \rightarrow & (1 + 1 + 1) & \rightarrow & ((1 + 1 + 1) + 1 + 1) \\ & & (1 \times 1) & & ((1 \times 1) + 1 + 1) \end{array}$$
- 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” \rightarrow all strings in \mathcal{A}_{ODD} have odd value

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

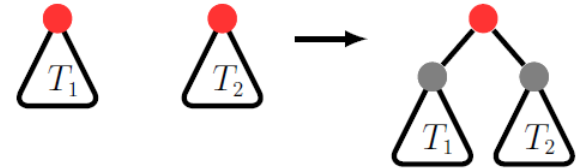
- The empty tree ε is an RBT.
- Disjoint RBTs T_1, T_2 give a new RBT by linking their roots to a new root.



- $P(T)$: 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 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 $n = 1, l = 0 = n - 1$
 - Case 2: $T_1 = \varepsilon; T_2 \neq \varepsilon$.
 - The child has one more node, $n = n_2 + 1$, and one more link:
 $l = l_2 + 1$
 $= n_2 - 1 + 1 = n_2 = n - 1$ [**induction hypothesis**]

Rooted Binary Trees with $n \geq 1$ Vertices Have $n - 1$ Edges, cont'd

- The empty tree ε 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 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 3: $T_1 \neq \varepsilon; T_2 = \varepsilon$. (Similar to Case 2.)

$$n = n_1 + 1 \text{ AND}$$

$$l = l_1 + 1$$

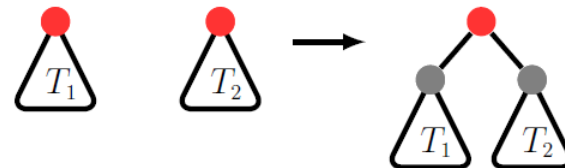
$$= n_1 - 1 + 1 = n_1 = n - 1 \text{ [induction hypothesis]}$$
 - Case 4: $T_1 \neq \varepsilon; T_2 \neq \varepsilon$.
 - Now, $n = n_1 + n_2 + 1$ and there are two new links, so

$$l = l_1 + l_2 + 2$$

$$= n_1 - 1 + n_2 - 1 + 2 = n_1 + n_2 = n - 1 \text{ [induction hypothesis]}$$

Rooted Binary Trees with $n \geq 1$ Vertices Have $n - 1$ Edges, cont'd

- The empty tree ε 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 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.
 - 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}