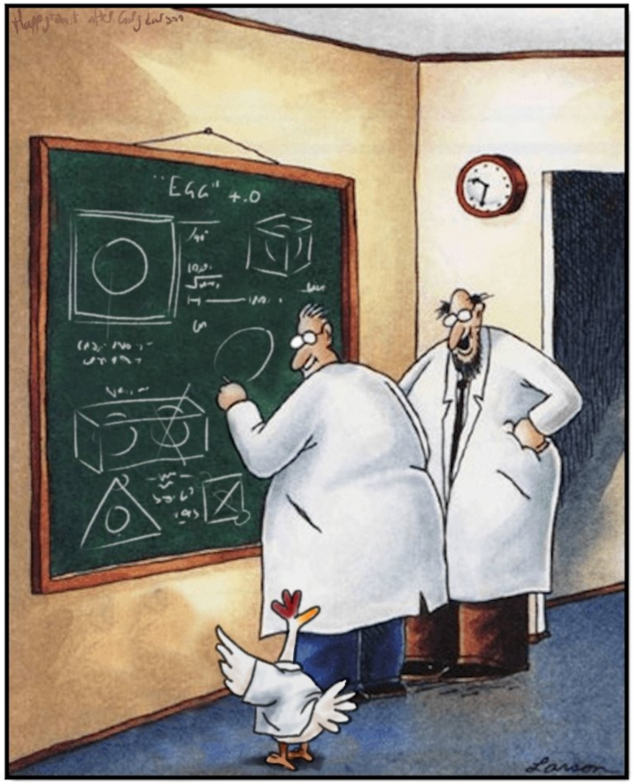
CSE 311: Foundations of Computing

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

CSE 311: Foundations of Computing

Lecture 17: Structural Induction

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 $\forall\, x \in S, P(x)$**

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.

Claim: Every element of S is divisible by 3.

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

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.

1. Let $P(x)$ be "$3|x$". We prove that $P(x)$ is true for all $x \in S$ by structural induction.
2. Base Case: $3|6$ and $3|15$ so $P(6)$ and $P(15)$ are true

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.

1. Let P(x) be "3|x". We prove that P(x) is true for all x ∈ S by structural induction.
2. Base Case: 3|6 and 3|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 ∈ S
4. Inductive Step: Goal: Show P(x+y)

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.

1. **Let** P(x) **be** "3|x". **We prove that** P(x) **is true for all** x ∈ S **by structural induction.**
2. **Base Case:** 3|6 **and** 3|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 ∈ S
4. **Inductive Step:** $\boxed{\textbf{Goal: Show } P(x+y)}$

   Since P(x) **is true,** 3|x **and so** x=3m **for some integer** m **and since** P(y) **is true,** 3|y **and so** y=3n **for some integer** n**.** **Therefore** x+y=3m+3n=3(m+n) **and thus** 3|(x+y).

   Hence P(x+y) **is true.**
5. **Therefore by induction** 3|x **for all** x ∈ 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 ∈ S". **We prove that** P(x) **is true for all** x ∈ R **by structural induction.**
2. **Base Case:** (12): 6 ∈ S **so** 6+6=12 ∈ S **by definition of** S**, so** P(12)
   (15): 15 ∈ S, **so** P(15) **is also true**
3. **Ind. Hyp: Suppose that** P(x) **is true for some arbitrary** x ∈ R
4. **Inductive Step:** **Goal: Show** P(x+6) **and** P(x+15)
   Since P(x) **holds, we have** x ∈ S. **Since** 6 ∈ S **from the recursive step of** S, **we get** x + 6 ∈ S, **so** P(x+6) **is true, and since** 15 ∈ S **we get** x + 15 ∈ S, **so** P(x+15) **is true.**
5. **Therefore** P(x) **(i.e.,** x ∈ S) **for all** x ∈ R **by induction.**

Basis: $6 \in S$; $15 \in S$	**Basis:** $12 \in R$; $15 \in R$
Recursive: if $x, y \in S$, then $x + y \in S$	**Recursive:** if $x \in R$, then $x + 6 \in R$ and $x + 15 \in R$

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:** $\mathrm{nil} \in$ **List**
- **Recursive step:**

  if $\mathrm{L} \in$ **List and** $\mathrm{a} \in \mathbb{Z}$**,**

  then $\mathrm{a} :: \mathrm{L} \in$ **List**

Examples:

– nil	[]
– $1 :: \mathrm{nil}$	[1]
– $2 :: 1 :: \mathrm{nil}$	[2, 1]
– $3 :: 2 :: 1 :: \mathrm{nil}$	[3, 2, 1]

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

Basis: nil $\in$ **List**

Recursive step:

if L $\in$ **List and** a $\in \mathbb{Z}$,

then a :: L $\in$ **List**

Length:

len(nil) := 0

len(a :: L) := len(L) + 1 **for any** L $\in$ **List and** a $\in \mathbb{Z}$

Concatenation:

concat(nil, R) := R **for any** R $\in$ **List**

concat(a :: L, R) := a :: concat(L, R) **for any** L, R $\in$ **List and any** a $\in \mathbb{Z}$

Structural Induction

Basis: $\mathrm{nil} \in$ **List**

Recursive step:

if $\mathrm{L} \in$ **List and** $\mathrm{a} \in \mathbb{Z}$,

then $\mathrm{a} :: \mathrm{L} \in$ **List**

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** $L, R \in$ **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: $\text{len}(\text{concat}(L, R)) = \text{len}(L) + \text{len}(R)$ **for all** $L, R \in$ **List**

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

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: $\text{len}(\text{concat}(L, R)) = \text{len}(L) + \text{len}(R)$ **for all** $L, R \in$ **List**

Let P(L) **be** "$\text{len}(\text{concat}(L, R)) = \text{len}(L) + \text{len}(R)$ **for all** R ∈ **List**" .
We prove P(L) **for all** L ∈ **List by structural induction.**

Base Case (nil): **Let** R ∈ **List be arbitrary. Then,**

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: $\text{len}(\text{concat}(\text{L}, \text{R})) = \text{len}(\text{L}) + \text{len}(\text{R})$ **for all** $\text{L}, \text{R} \in$ **List**

Let P(L) **be** "$\text{len}(\text{concat}(\text{L}, \text{R})) = \text{len}(\text{L}) + \text{len}(\text{R})$ **for all** R ∈ **List**".
We prove P(L) **for all** L ∈ **List by structural induction.**

Base Case (nil): **Let** R ∈ **List be arbitrary. Then,**

$$
\begin{aligned}
\text{len}(\text{concat}(\text{nil}, \text{R})) &= \text{len}(\text{R}) && \textbf{def of concat} \\
&= 0 + \text{len}(\text{R}) \\
&= \text{len}(\text{nil}) + \text{len}(\text{R}) && \textbf{def of len}
\end{aligned}
$$

Since R **was arbitrary,** P(nil) **holds.**

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

Let P(L) **be** "$\text{len}(\text{concat}(L, R)) = \text{len}(L) + \text{len}(R)$ **for all** R ∈ **List**" .
We prove P(L) **for all** L ∈ **List by structural induction.**

Base Case (nil): **Let** R ∈ **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(nil).

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

Basis: $\text{nil} \in$ **List**

Recursive step:

if $L \in$ **List and** $a \in \mathbb{Z}$,

then $a :: L \in$ **List**

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

Let P(L) **be** "$\text{len}(\text{concat}(L, R)) = \text{len}(L) + \text{len}(R)$ **for all** R ∈ **List**".
We prove P(L) **for all** L ∈ **List by structural induction.**

Base Case (nil): **Let** R ∈ **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(nil).

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

Inductive Step: $\boxed{\textbf{Goal: Show that } P(a :: L) \textbf{ is true}}$

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

Let P(L) **be** "$\text{len}(\text{concat}(L, R)) = \text{len}(L) + \text{len}(R)$ **for all** R ∈ **List**".
We prove P(L) **for all** L ∈ **List by structural induction.**

Base Case (nil): **Let** R ∈ **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(nil).

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

Inductive Step: $\boxed{\textbf{Goal: Show that } P(a :: L) \textbf{ is true}}$

Let R ∈ **List be arbitrary. Then,**

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: $\text{len}(\text{concat}(L, R)) = \text{len}(L) + \text{len}(R)$ **for all** $L, R \in$ **List**

Let P(L) **be** "$\text{len}(\text{concat}(L, R)) = \text{len}(L) + \text{len}(R)$ **for all** R ∈ **List**".
We prove P(L) **for all** L ∈ **List by structural induction.**

Base Case (nil): **Let** R ∈ **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(nil).

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

Inductive Step: $\boxed{\textbf{Goal: Show that } P(a :: L) \textbf{ is true}}$

Let R ∈ **List be arbitrary. Then, we can calculate**

$\text{len}(\text{concat}(a :: L, R))$	$= \text{len}(a :: \text{concat}(L, R))$	**def of** concat
	$= 1 + \text{len}(\text{concat}(L, R))$	**def of** len
	$= 1 + \text{len}(L) + \text{len}(R)$	**IH**
	$= \text{len}(a :: L) + \text{len}(R)$	**def of** len

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

Let P(L) **be** "$\text{len}(\text{concat}(L, R)) = \text{len}(L) + \text{len}(R)$ **for all** R ∈ **List**".
We prove P(L) **for all** L ∈ **List by structural induction.**

Base Case (nil): **Let** R ∈ **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(nil).

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

Inductive Step: $\boxed{\textbf{Goal: Show that } P(a :: L) \textbf{ is true}}$

Let R ∈ **List be arbitrary. Then, we can calculate**

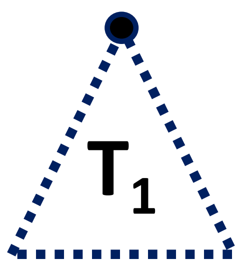
$$
\begin{aligned}
\text{len}(\text{concat}(a :: L, R)) &= \text{len}(a :: \text{concat}(L, R)) && \textbf{def of } \text{concat} \\
&= 1 + \text{len}(\text{concat}(L, R)) && \textbf{def of } \text{len} \\
&= 1 + \text{len}(L) + \text{len}(R) && \textbf{IH} \\
&= \text{len}(a :: L) + \text{len}(R) && \textbf{def of } \text{len}
\end{aligned}
$$

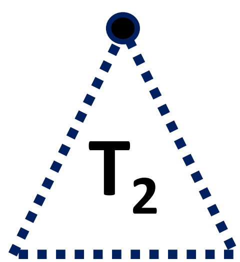
Since R **was arbitrary, we have shown** P(a :: L).

By induction, we have shown the claim holds for all L ∈ **List**.

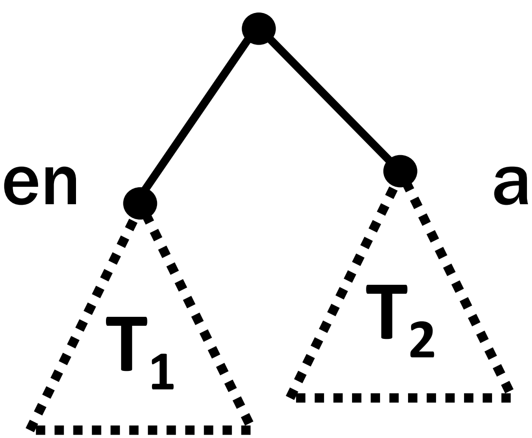
Rooted Binary Trees

- **Basis:** • is a rooted binary tree
- **Recursive step:**

If 

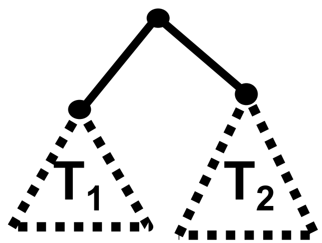
and 

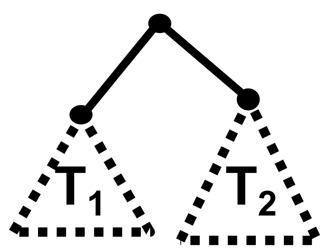
are rooted binary trees,

then 

also is a rooted binary tree.

Defining Functions on Rooted Binary Trees

- size(•) := 1
- size (

) := 1 + size($\mathbf{T_1}$) + size($\mathbf{T_2}$)
- height(•) := 0
- height (

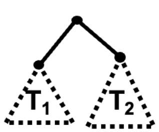
) := 1 + max{height($\mathbf{T_1}$), height($\mathbf{T_2}$)}

Claim: **For every rooted binary tree T,** size(**T**) $\leq 2^{\text{height}(\mathbf{T})+1} - 1$

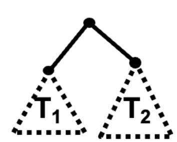
Claim: For every rooted binary tree T, $\text{size}(\mathbf{T}) \leq 2^{\text{height}(\mathbf{T}) + 1} - 1$

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

size(•) ::= 1

size () ::= 1 + size($\mathbf{T_1}$) + size($\mathbf{T_2}$)

height(•) ::= 0

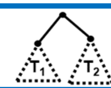
height () ::= 1 + max{height($\mathbf{T_1}$), height($\mathbf{T_2}$)}

Claim: For every rooted binary tree T, $\text{size}(\mathbf{T}) \leq 2^{\text{height}(\mathbf{T}) + 1} - 1$

1. **Let** $P(\mathbf{T})$ **be** "$\text{size}(T) \leq 2^{\text{height}(T)+1} - 1$". **We prove** $P(\mathbf{T})$ **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, $\text{size}(\mathbf{T}) \leq 2^{\text{height}(\mathbf{T})+1} - 1$

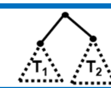
1. **Let** $P(\mathbf{T})$ **be** "$\text{size}(T) \leq 2^{\text{height}(T)+1} - 1$". **We prove** $P(\mathbf{T})$ **for all rooted binary trees T by structural induction.**
2. **Base Case:** $\text{size}(\bullet)=1$, $\text{height}(\bullet)=0$, **and** $2^{0+1}-1=2^1-1=1$ **so** $P(\bullet)$ **is true.**
3. **Inductive Hypothesis: Suppose that** $P(\mathbf{T_1})$ **and** $P(\mathbf{T_2})$ **are true for some rooted binary trees $\mathbf{T_1}$ and $\mathbf{T_2}$, i.e.,** $\text{size}(\mathbf{T}_k) \leq 2^{\text{height}(\mathbf{T}_k)+1} - 1$ **for** $k=1,2$
4. **Inductive Step:**

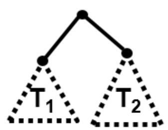
Goal: Prove P(

).

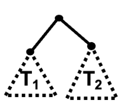
Claim: For every rooted binary tree T, size(T) ≤ $2^{\text{height}(T)+1} - 1$

1. **Let** P(**T**) **be** "size(T) ≤ $2^{\text{height}(T)+1}-1$". **We prove** P(**T**) **for all rooted binary trees T by structural induction.**
2. **Base Case:** size(•)=1, height(•)=0, **and** $2^{0+1}-1=2^1-1=1$ **so** P(•) **is true.**
3. **Inductive Hypothesis: Suppose that** P($\mathbf{T_1}$) **and** P($\mathbf{T_2}$) **are true for some rooted binary trees** $\mathbf{T_1}$ **and** $\mathbf{T_2}$**, i.e.,** size($\mathbf{T}_k$) ≤ $2^{\text{height}(\mathbf{T}_k)+1} - 1$ **for** k=1,2
4. **Inductive Step:**

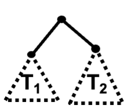
Goal: Prove P().

size()

size(•) ::= 1

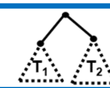
size () ::= 1 + size($\mathbf{T_1}$) + size($\mathbf{T_2}$)

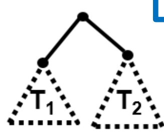
height(•) ::= 0

height () ::= 1 + max{height($\mathbf{T_1}$), height($\mathbf{T_2}$)}

≤ 2^height()+1 – 1

Claim: For every rooted binary tree T, size(T) $\leq 2^{\text{height}(T)+1} - 1$

1. **Let** P(**T**) **be** "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:** size(•)=1, height(•)=0, **and** $2^{0+1}-1=2^1-1=1$ **so** P(•) **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.,** size(T_k) $\leq 2^{\text{height}(T_k)+1} - 1$ **for** k=1,2
4. **Inductive Step:** **Goal: Prove** P().

By def, size(

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

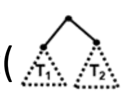
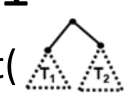
$\leq 1 + 2^{\text{height}(T_1)+1} - 1 + 2^{\text{height}(T_2)+1} - 1$

by IH for T_1 and T_2

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

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

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

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

which is what we wanted to show.

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

Strings

- An *alphabet* Σ is any finite set of characters

- The set Σ^* of *strings* over the alphabet Σ
  - example: $\{0,1\}^*$ is the set of *binary strings*

    0, 1, 00, 01, 10, 11, 000, 001, … and “”

- Σ^* is defined recursively by
  - Basis: $\varepsilon \in \Sigma^*$ (ε is the empty string, i.e., “”)
  - Recursive: if $w \in \Sigma^*$, $a \in \Sigma$, then $wa \in \Sigma^*$

Palindromes

Palindromes are strings that are the same when read backwards and forwards

Basis:

ε is a palindrome
any $a \in \Sigma$ is a palindrome

Recursive step:

If p is a palindrome,
then apa is a palindrome for every $a \in \Sigma$

Functions on Recursively Defined Sets (on Σ^*)

Length:

$len(\varepsilon) := 0$

$len(wa) := len(w) + 1$ for $w \in \Sigma^*$, $a \in \Sigma$

Concatenation:

$x \bullet \varepsilon := x$ for $x \in \Sigma^*$

$x \bullet wa := (x \bullet w)a$ for $x \in \Sigma^*$, $a \in \Sigma$

Reversal:

$\varepsilon^R := \varepsilon$

$(wa)^R := a \bullet w^R$ for $w \in \Sigma^*$, $a \in \Sigma$

Number of c's in a string:

$\#_c(\varepsilon) := 0$

$\#_c(wc) := \#_c(w) + 1$ for $w \in \Sigma^*$

$\#_c(wa) := \#_c(w)$ for $w \in \Sigma^*$, $a \in \Sigma$, $a \neq c$

separate cases for
c vs $a \neq c$

Last time: Structural Induction

Basis: $\varepsilon \in \Sigma^*$

Recursive Steps:

if $w \in \Sigma^*$ and $a \in \Sigma$,

then $wa \in \Sigma^*$

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: $\text{len}(x \bullet y) = \text{len}(x) + \text{len}(y)$ **for all** $x, y \in \Sigma^*$

Let $P(y)$ **be** "$\text{len}(x \bullet y) = \text{len}(x) + \text{len}(y)$ for all $x \in \Sigma^*$" .
We prove $P(y)$ **for all** $y \in \Sigma^*$ **by structural induction.**

Base Case ($y = \varepsilon$): **Let** $x \in \Sigma^*$ **be arbitrary. Then,** $\text{len}(x \bullet \varepsilon) = \text{len}(x) = \text{len}(x) + \text{len}(\varepsilon)$ **since** $\text{len}(\varepsilon)=0$. **Since** x **was arbitrary,** $P(\varepsilon)$ **holds.**

Inductive Hypothesis: Assume that $P(w)$ **is true for some arbitrary** $w \in \Sigma^*$**, i.e.,** $\text{len}(x \bullet w) = \text{len}(x) + \text{len}(w)$ for all x

Claim: $\text{len}(x \bullet y) = \text{len}(x) + \text{len}(y)$ **for all** $x, y \in \Sigma^*$

Let $P(y)$ **be** "$\text{len}(x \bullet y) = \text{len}(x) + \text{len}(y)$ for all $x \in$
We prove $P(y)$ **for all** $y \in \Sigma^*$ **by structural induc**

Does this look familiar?

Base Case $(y = \varepsilon)$: **Let** $x \in \Sigma^*$ **be arbitrary. Then,** $\text{len}(x \bullet \varepsilon) = \text{len}(x) = \text{len}(x) + \text{len}(\varepsilon)$ **since** $\text{len}(\varepsilon)=0$. **Since** x **was arbitrary,** $P(\varepsilon)$ **holds.**

Inductive Hypothesis: Assume that $P(w)$ **is true for some arbitrary** $w \in \Sigma^*$, **i.e.,** $\text{len}(x \bullet w) = \text{len}(x) + \text{len}(w)$ for all x

Inductive Step: **Goal: Show that** $P(wa)$ **is true for every** $a \in \Sigma$

Let $a \in \Sigma$ **and** $x \in \Sigma^*$. **Then** $\text{len}(x \bullet wa) = \text{len}((x \bullet w)a)$ **by def of** $\bullet$
$= \text{len}(x \bullet w)+1$ **by def of len**
$= \text{len}(x)+\text{len}(w)+1$ **by I.H.**
$= \text{len}(x)+\text{len}(wa)$ **by def of len**

Therefore, $\text{len}(x \bullet wa)= \text{len}(x)+\text{len}(wa)$ **for all** $x \in \Sigma^*$, **so** $P(wa)$ **is true.**

So, by induction $\text{len}(x \bullet y) = \text{len}(x) + \text{len}(y)$ **for all** $x, y \in \Sigma^*$