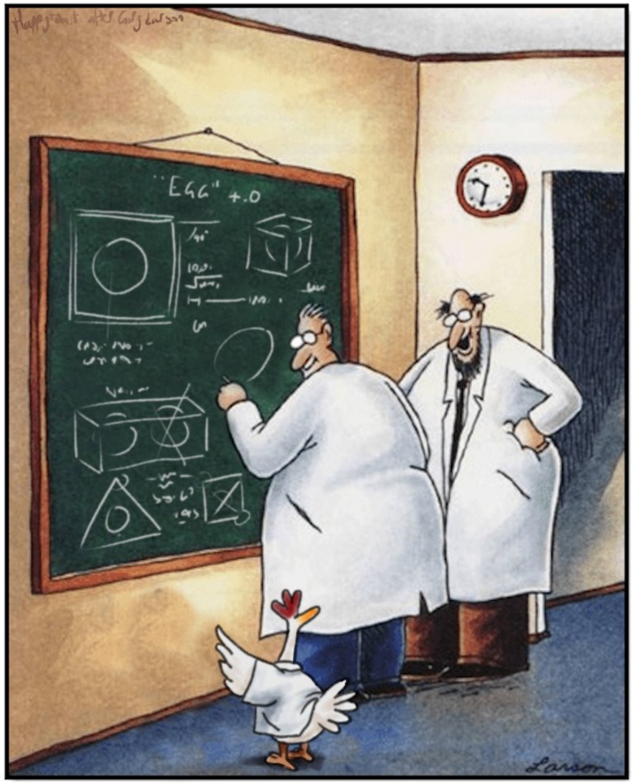
CSE 311: Foundations of Computing

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 $x + y \in S$.

6 15

21 12 30

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 ∈ S by structural induction.

P(6) ✓

P(15) ✓

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

IH: P(x) P(y)

Goal: 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:** 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: 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 $\forall x \in R. P(x)$

$\forall x.\ x \in R \to x \in S$ | $\forall x \in R.\ x \in S$

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: $6 \in S$; $15 \in S$ **Recursive:** if $x, y \in S$, then $x + y \in S$	**Basis:** $12 \in R$; $15 \in R$ **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 ∈ **List**
Recursive step:
  if L ∈ **List and** a ∈ ℤ,
  then a :: L ∈ **List**

Length:

$\text{len}(\text{nil}) := 0$
$\text{len}(a :: L) := \text{len}(L) + 1$ **for any** L ∈ **List and** a ∈ ℤ

Concatenation:

$\text{concat}(\text{nil}, R) := R$ **for any** R ∈ **List**
$\text{concat}(a :: L, R) := a :: \text{concat}(L, R)$ **for any** L, R ∈ **List and any** a ∈ ℤ

Structural Induction

Basis: nil ∈ **List**

Recursive step:

if L ∈ **List and** a ∈ ℤ,

then a :: L ∈ **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: len(concat(L, R)) = len(L) + len(R)   **for all** L, R ∈ **List**

P(X) := ∀R∈List. len(concat(X,R)) = len(X) + len(R)

∀X∈List. P(X)

∀X∈List. ∀R∈List. ...

Length:

len(nil) := 0
len(a :: L) := len(L) + 1

Concatenation:

concat(nil, R) := R
concat(a :: L, R) := a :: 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.**

$P(\text{nil}) = \forall R \in \text{List},\ \text{len}(\text{concat}(\text{nil}, R))$
$= \text{len}(\text{nil}) + \text{len}(R)$

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,**

len(concat(nil, R)) = len(R) by def of concat

= 0 + len(R) algebra
= len(nil) + len(R) by def of len

Length:
len(nil) := 0
len(a :: L) := len(L) + 1

Concatenation:
concat(nil, R) := R
concat(a :: L, R) := a :: 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,**

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

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.**

Goal: P(a::L)

Basis: nil ∈ **List**

Recursive step:

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

then a :: L ∈ **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: **Goal: Show that** $P(a :: L)$ **is true**

Claim: len(concat(L, R)) = len(L) + len(R) **for all** L, R ∈ **List**

Let P(L) **be** "len(concat(L, R)) = len(L) + 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,** len(concat(nil, R)) = len(R) = 0 + len(R) = len(nil) + len(R), **showing** P(nil).

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

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

Let R ∈ **List be arbitrary. Then,**

len(concat(a::L, R)) = len(a:: concat(L, R)) by def of concat

= len(concat(L, R)) + 1 by def of len

= len(L) + len(R) + 1 by IH

Length:

len(nil) := 0
len(a :: L) := len(L) + 1

Concatenation:

concat(nil, R) := R
concat(a :: L, R) := a :: 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: **Goal: Show that** $P(a :: L)$ **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 ∈ **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: **Goal: Show that P(a :: L) is true**

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

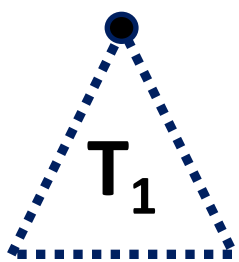
$\text{len}(\text{concat}(a :: L, R))$	$= \text{len}(a :: \text{concat}(L, R))$	**def of** concat
	$= 1 + \text{len}(\text{concat}(L, R))$	**def of** len
	$= 1 + \text{len}(L) + \text{len}(R)$	**IH**
	$= \text{len}(a :: L) + \text{len}(R)$	**def of** len

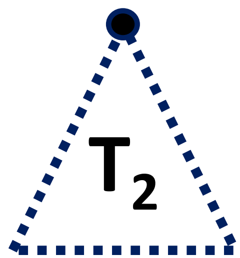
Since R **was arbitrary, we have shown** P(a :: L).

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

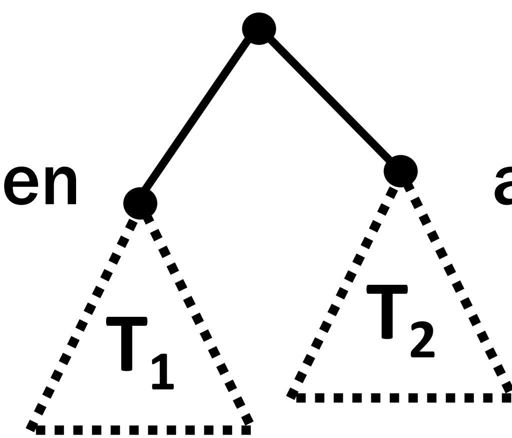
Rooted Binary Trees

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

If 

and 

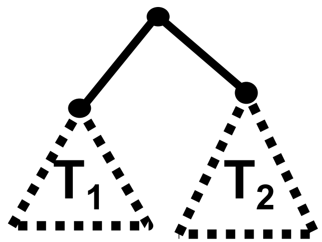
are rooted binary trees,

then 

also is a rooted binary tree.

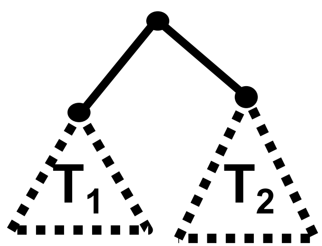
Defining Functions on Rooted Binary Trees

- 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**) ≤ $2^{\text{height}(\mathbf{T}) + 1} - 1$

$\forall x \in \text{Tree}.\ \text{size}(x) \leq 2^{\text{height}(x)+1} - 1$

$P(x) :=$

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

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.

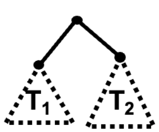
Goal: P(•) = size(•) $\leq 2^{h(\bullet)+1} - 1$

size(•) = 1

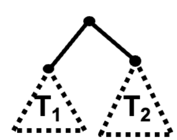
$\leq$

1

size(•) ::= 1

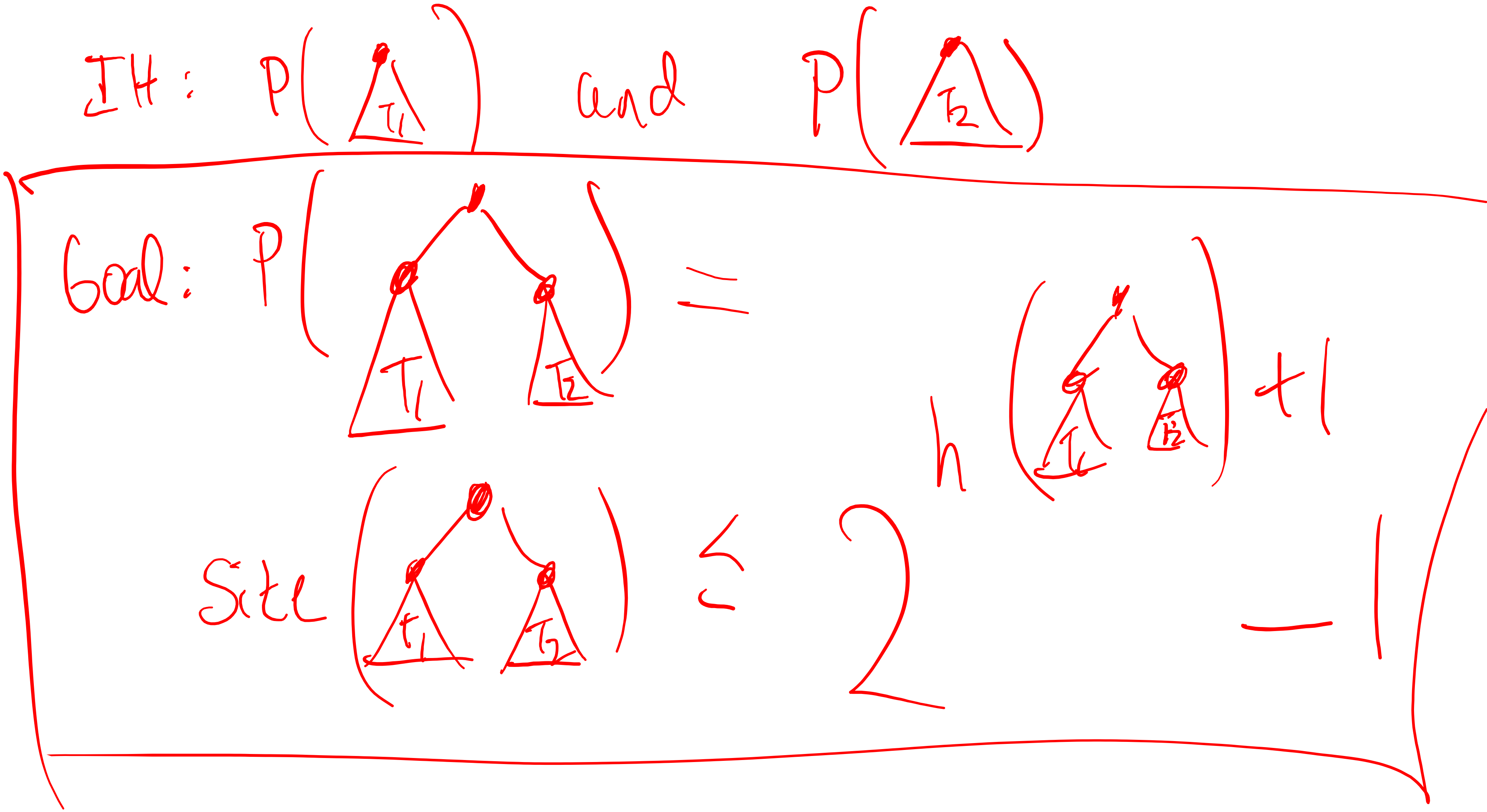
size () ::= 1 + size($\mathbf{T}_1$) + size($\mathbf{T}_2$)

height(•) ::= 0

height () ::= 1 + max{height($\mathbf{T}_1$), height($\mathbf{T}_2$)}

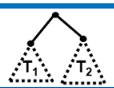
Claim: For every rooted binary tree T, size(T) $\leq 2^{\text{height}(\mathbf{T})+1} - 1$

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.**



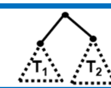
Claim: **For every rooted binary tree T,** size(**T**) $\leq 2^{\text{height}(\mathbf{T}) + 1} - 1$

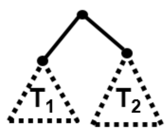
1. **Let** P(**T**) **be** "size(T) $\leq 2^{\text{height}(T)+1} - 1$". **We prove** P(**T**) **for all rooted binary trees T by structural induction.**
2. **Base Case:** size(•)=1, height(•)=0, **and** $2^{0+1}-1=2^1-1=1$ **so** P(•) **is true.**
3. **Inductive Hypothesis: Suppose that** P($\mathbf{T_1}$)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

Goal: Prove P().

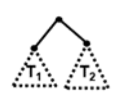
Claim: For every rooted binary tree T, size(T) ≤ $2^{\text{height}(T)+1} - 1$

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

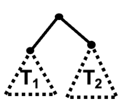
Goal: Prove P().

size() = 1 + size(T_1) + size(T_2)   by def of size

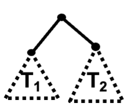
≤ $1 + 2^{h(T_1)+1} - 1 + 2^{h(T_2)+1} - 1$ by IH x2

≤ $2^{\text{height}(\text{tree})+1} - 1$ (tree: )

size(•) ::= 1

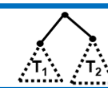
size () ::= 1 + size(T_1) + size(T_2)

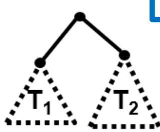
height(•) ::= 0

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

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

1. Let P(T) be "$\text{size}(T) \le 2^{\text{height}(T)+1} - 1$". 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., $\text{size}(T_k) \le 2^{\text{height}(T_k)+1} - 1$ for k=1,2
4. **Inductive Step:**

Goal: Prove P().

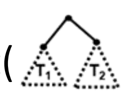
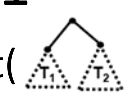
By def, size() $=1+\text{size}(T_1)+\text{size}(T_2)$

$\le 1+2^{\text{height}(T_1)+1}-1+2^{\text{height}(T_2)+1}-1$ by IH for T_1 and T_2

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

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

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

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

which is what we wanted to show.

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