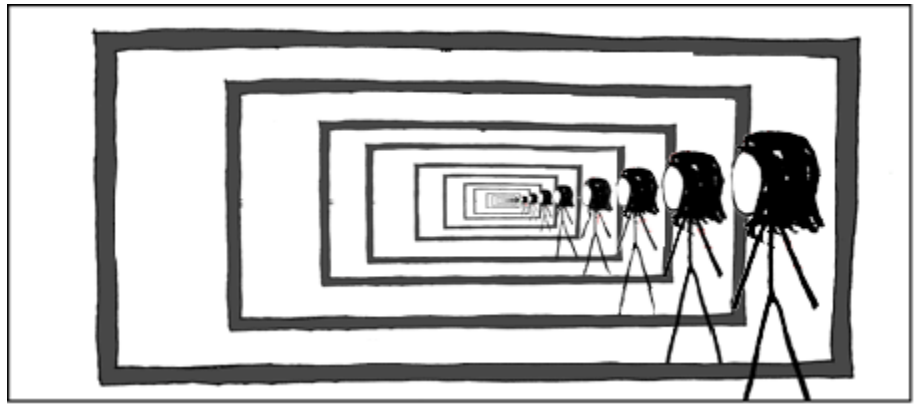
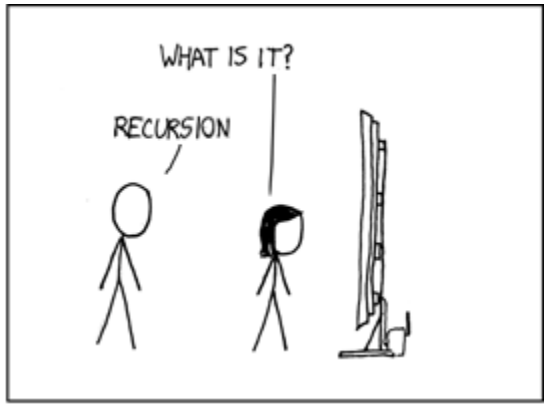
Warm up:

What's the first sentence of the base case and IS for a proof of:

$P(n)$: "Every set of integers of size n has a largest element"





Structural Induction

CSE 311 Spring 22
Lecture 19

Announcements

Homework 6 comes out tonight, due May 18th.

Normal number of problems, but a lot of them are induction, so those extra days may be necessary.

We're wrapping up some logistics related to the midterm today.

Once that's all done, we'll post solutions.

Late this week and next week, I'll have slots for one-on-one meetings to discuss grade concerns.

Recursive Definitions of Sets

Define a set S as follows:

Basis Step: $0 \in S$

Recursive Step: If $x \in S$ then $x + 2 \in S$.

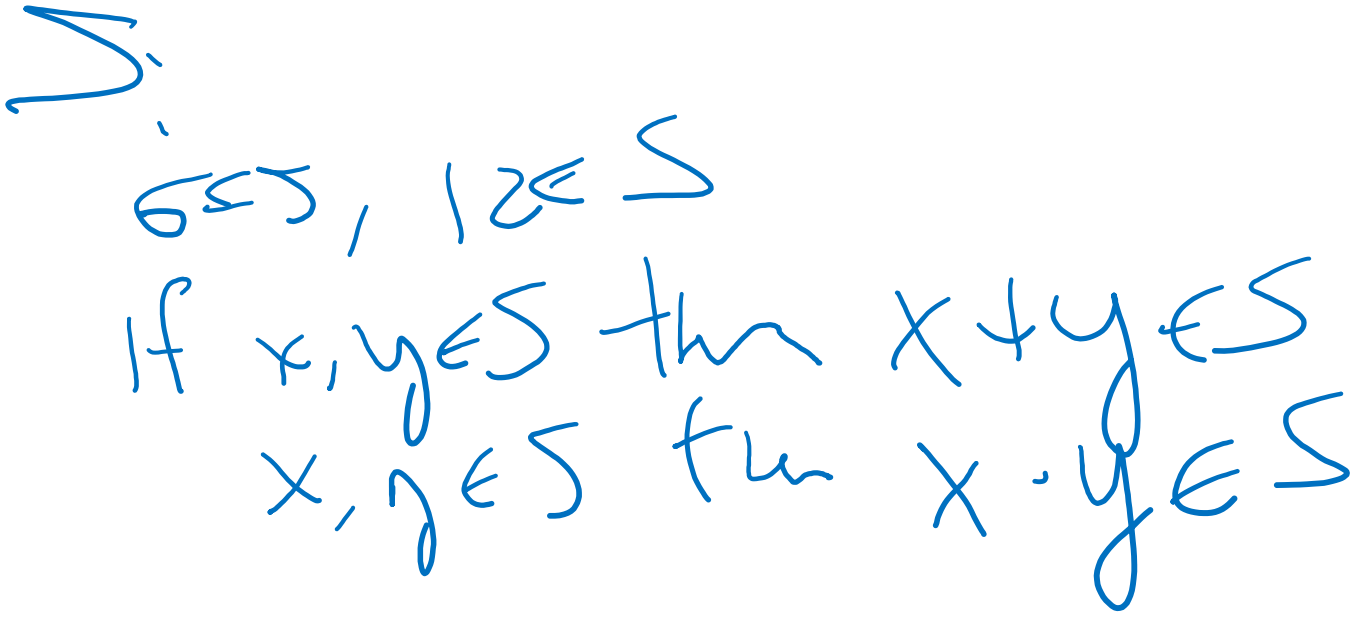
Exclusion Rule: Every element of S is in S from the basis step (alone) or a finite number of recursive steps starting from a basis step.

What is S?

Recursive Definitions of Sets

We'll always list the Basis and Recursive parts of the definition.

Starting…now…we're going to be lazy and skip writing the "exclusion" rule. It's still part of the definition.



Recursive Definitions of Sets

Basis: $6 \in S, 15 \in S$

Recursive: If $x, y \in S$ then $x + y \in S$

Basis: $\begin{bmatrix}1\\1\\0\end{bmatrix} \in S \begin{bmatrix}1\\0\\1\end{bmatrix} \in S$

Recursive: if $x \in S$, then $\alpha x \in S$ for all $\alpha \in \mathbb{R}$.

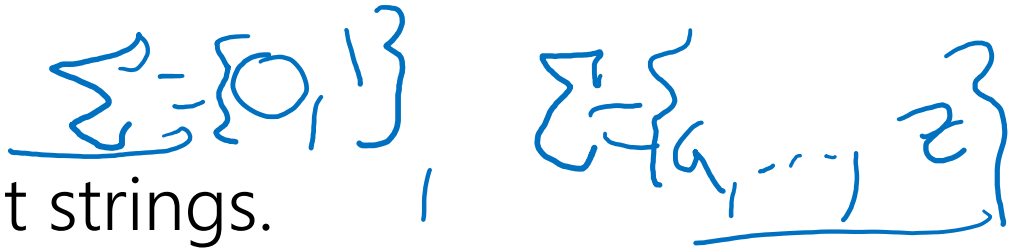
If $x, y \in S$ then $x + y \in S$.

Write a recursive definition of $\{x : x = 3^i \text{ for some } i \in \mathbb{N}\}$.

Strings

Why these recursive definitions?

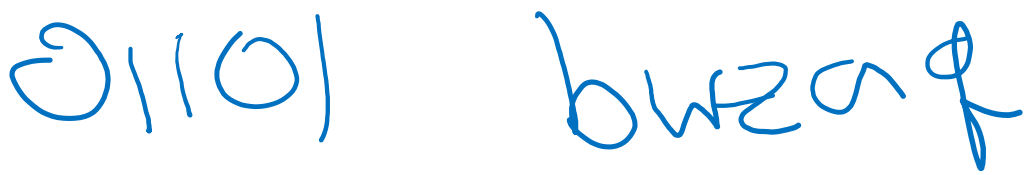
They're the basis for regular expressions, which we'll introduce next week. Answer questions like "how do you search for anything that looks like an email address"



First, we need to talk about strings.

Σ will be an **alphabet** the set of all the letters you can use in words.

Σ^* is the set of all **words** all the strings you can build off of the letters.



Strings

ε is "the empty string"

The string with **0** characters – `""` `in Java` (not `null!`)

Σ^*:

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

Recursive: If $w \in \Sigma^*$ and $a \in \Sigma$ then $wa \in \Sigma^*$

wa means the string of w with the character a appended.

You'll also see $w \cdot a$ (a $\cdot$ to mean "concatenate" i.e. + in Java)

Functions on Strings

Since strings are defined recursively, most functions on strings are as well.

Length:

len(ε)=0;

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

Reversal:

$\varepsilon^R = \varepsilon$;
$(wa)^R = aw^R$ for $w \in \Sigma^*$, $a \in \Sigma$

Concatenation

$x \cdot \varepsilon = x$ for all $x \in \Sigma^*$;
$x \cdot (wa) = (x \cdot w)a$ 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 $w \in \Sigma^*, a \in \Sigma \setminus \{c\}$.

Functions on Strings

Since strings are defined recursively, most functions on strings are as well.

Length:

len(ε)=0;

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

Reversal:

$\varepsilon^R = \varepsilon$;
$(wa)^R = aw^R$ for $w \in \Sigma^*$, $a \in \Sigma$

Concatenation

$x \cdot \varepsilon = x$ for all $x \in \Sigma^*$;
$x \cdot (wa) = (x \cdot w)a$ 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 $w \in \Sigma^*, a \in \Sigma \setminus \{c\}$.

Structural Induction

Every element is built up recursively…

So to show $P(s)$ for all $s \in S$…

Show $P(b)$ for all base case elements b.

Show if $P()$ holds for every named element in the recursive rule, then $P()$ holds for the new element (repeat for each rule).

Structural Induction Example

Let S be:
Basis: $6 \in S, 15 \in S$
Recursive: if $x, y \in S$ then $x + y \in S$.

Show that every element of S is divisible by 3.

Structural Induction

Let $P(x)$ be "x is divisible by 3."

We show $P(x)$ holds for all $x \in S$ by structural induction.

Base Cases:

Inductive Hypothesis:

Inductive Step:

We conclude $P(x) \forall x \in S$ by the principle of induction.

Basis: $6 \in S, 15 \in S$

Recursive: if $x, y \in S$ then $x + y \in S$.

Structural Induction

If $x, y \in S$ then $x+y \in S$

Let $P(x)$ be "x is divisible by 3."

We show $P(x)$ holds for all $x \in S$ by structural induction.

Base Cases:
$6 = 2 \cdot 3$ so $3|6$, and $P(6)$ holds. $15 = 5 \cdot 3$, so$3|15$ and $P(15)$ holds.

Inductive Hypothesis: Suppose $P(x)$ and $P(y)$ for arbitrary x, y.

Inductive Step:

By IH $3|x$, $3|y$

$P(x+y) \rightarrow 3|x+y$

We conclude $P(x) \forall x \in S$ by the principle of induction.

Basis: $6 \in S, 15 \in S$
Recursive: if $x, y \in S$ then $x + y \in S$.

Structural Induction

Let $P(x)$ be "x is divisible by 3."

We show $P(x)$ holds for all $x \in S$ by structural induction.

Base Cases:
$6 = 2 \cdot 3$ so $3|6$, and $P(6)$ holds. $15 = 5 \cdot 3$, so$3|15$ and $P(15)$ holds.

Inductive Hypothesis: Suppose $P(x)$ and $P(y)$ for arbitrary x, y.

Inductive Step: By IH $3|x$ and $3|y$. So $x = 3n$ and $y = 3m$ for integers m, n.

Adding the equations, $x + y = 3(n + m)$. Since n, m are integers, we have $3|(x + y)$ by definition of divides. This gives $P(x + y)$.

We conclude $P(x) \forall x \in S$ by the principle of induction.

Basis: $6 \in S, 15 \in S$
Recursive: if $x, y \in S$ then $x + y \in S$.

Structural Induction Template

1. Define $P()$ Show that $P(x)$ holds for all $x \in S$. State your proof is by structural induction.

2. Base Case: Show $P(x)$ for all base cases x in S.

3. Inductive Hypothesis: Suppose $P(x)$ for all x listed as in S in the recursive rules.

4. Inductive Step: Show $P()$ holds for the "new element" given.

   You will need a separate step for every rule.

5. Therefore $P(x)$ holds for all $x \in S$ by the principle of induction.

Wait a minute! Why can we do this?

Think of each element of S as requiring k "applications of a rule" to get in

$P(base\ cases)$ is true

$P(base\ cases) \rightarrow P(one\ application)$ so $P(one\ application)$

$P(one\ application) \rightarrow P(two\ applications)$ so $P(two\ applications)$

…

It's the same principle as regular induction. You're just inducting on "how many steps did we need to get this element?"

You're still only assuming the IH about a domino you've knocked over.

Wait a minute! Why can we do this?

Imagine building S "step-by-step"

$S_0 = \{6,15\}$

$S_1 = \{12,21,30\}$

$S_2 = \{18,24,27,33,36,42,45,51,60\}$

IS can always of the form "suppose $P(x) \forall x \in (S_0 \cup \cdots \cup S_k)$" and show $P(y)$ for some $y \in S_{k+1}$

We use the structural induction phrasing assuming our reader knows how induction works and so don't phrase it explicitly in this form.

Claim len(x·y)=len(x) + len(y) for all $x, y \in \Sigma^*$.

Let $P(y)$ be "len(x·y)=len(x) + len(y) for all $x \in \Sigma^*$."

Notice the strangeness of this $P()$ there is a "for all x" inside the definition of $P(y)$.

That means we'll have to introduce an arbitrary x as part of the base case and the inductive step!

Claim len(x·y)=len(x) + len(y) for all $x, y \in \Sigma^*$.

Define Let $P(y)$ be "len(x·y)=len(x) + len(y) for all $x \in \Sigma^*$."

We prove $P(y)$ for all $y \in \Sigma^*$ by structural induction.

Base Case: $P(\varepsilon)$. Goal: $\forall x \in \Sigma^*, \text{len}(x \cdot \varepsilon) = \text{len}(x) + \text{len}(\varepsilon)$.

Inductive Hypothesis:

Inductive Step:

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

Recursive: If $w \in \Sigma^*$ and $a \in \Sigma$ then $wa \in \Sigma^*$

Claim len(x·y)=len(x) + len(y) for all $x, y \in \Sigma^*$.

Define Let $P(y)$ be "len(x·y)=len(x) + len(y) for all $x \in \Sigma^*$."

We prove $P(y)$ for all $y \in \Sigma^*$ by structural induction.

Base Case: Let x be an arbitrary string, len($x \cdot \epsilon$)=len(x)
=len(x)+0=len(x)+len(ε)

Inductive Hypothesis: Suppose $P(w)$

Inductive Step: Let x be an arbitrary string.

Therefore, len(xwa)=len(x) + len(wa)

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

Recursive: If $w \in \Sigma^*$ and $a \in \Sigma$ then $wa \in \Sigma^*$

Claim len(x·y)=len(x) + len(y) for all $x, y \in \Sigma^*$.

Define Let $P(y)$ be "len(x·y)=len(x) + len(y) for all $x \in \Sigma^*$."

We prove $P(y)$ for all $y \in \Sigma^*$ by structural induction.

Base Case: Let x be an arbitrary string, len($x \cdot \epsilon$)=len(x)
=len(x)+0=len(x)+len(ε)

Inductive Hypothesis: Suppose $P(w)$

Inductive Step: Let x be an arbitrary string.

len(xwa) =len(xw)+1 (by definition of len)

=len(x) + len(w) + 1 (by IH)

=len(x) + len(wa) (by definition of len)

Therefore, len(xwa)=len(x) + len(wa)

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

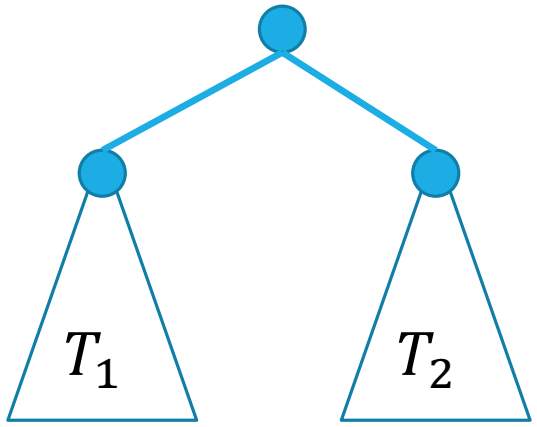
Recursive: If $w \in \Sigma^*$ and $a \in \Sigma$ then $wa \in \Sigma^*$

More Structural Sets

Binary Trees are another common source of structural induction.

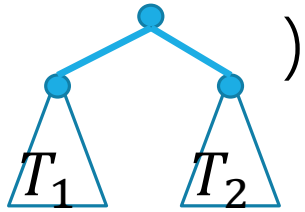
Basis: A single node is a rooted binary tree.

Recursive Step: If T_1 and T_2 are rooted binary trees with roots r_1 and r_2, then a tree rooted at a new node, with children r_1, r_2 is a binary tree.

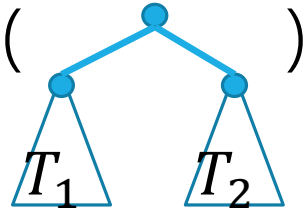


Functions on Binary Trees

size(●)=1

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

height(●) = 0

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

Structural Induction on Binary Trees

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

We'll show this next time.