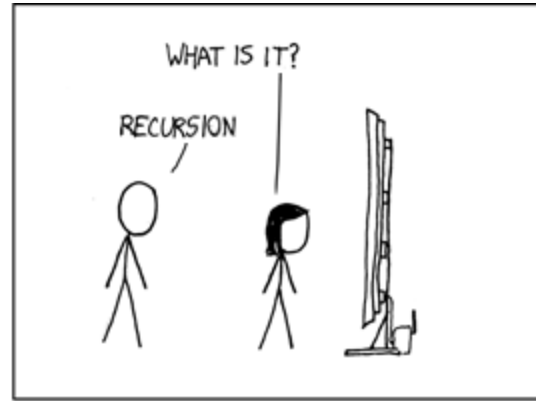
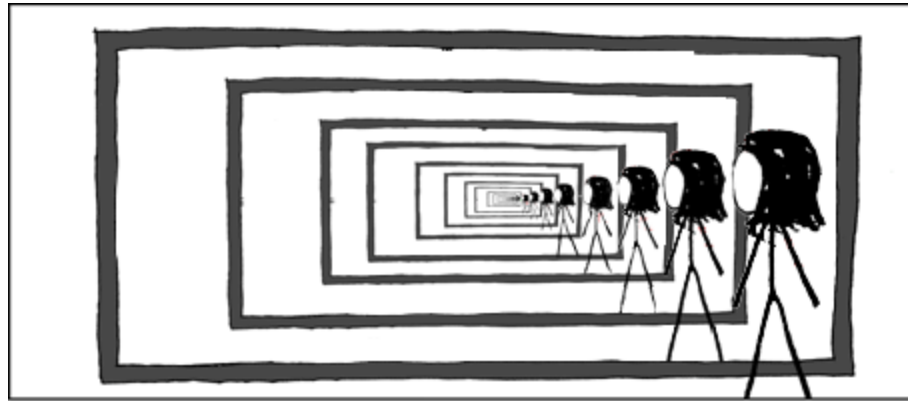


No activity today  
Happy Lunar New Year!



# Structural Induction

CSE 311 Winter 21  
Lecture 17

# Announcements

More midterm logistics details on page.

Practice materials also up.

Exam will have “fewer, longer” questions.

Would be comfortable giving as a time-constrained 2 hour exam.

I’m estimating most folks will need about 3 hours to account for typing time/collaboration time.

But you’re allowed as much time as you want, provided it’s in by the due date.

NO late days on the exam; if you run into issues, send Jamie an email.

Once the exam is out, we will only answer clarifying questions on questions == not any on course content (like you were in an exam room).

# Recursive Definition 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:  $1$

Recursive:  $\forall x \in S, \exists x \in S$

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

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.

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

$\Sigma^*$  is the set of all **words** all the strings you can build off of the letters.

# Strings

$\varepsilon$  is "the empty string"

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

$\Sigma^*$ :

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

|| " a |  
a

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:

$$\text{len}(\varepsilon) = 0;$$

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

Reversal:

$$\varepsilon^R = \varepsilon;$$

$$(wa)^R = aw^R \text{ for } w \in \Sigma^*, a \in \Sigma$$

Concatenation

$$x \cdot \varepsilon = x \text{ for all } x \in \Sigma^*;$$

$$x \cdot (wa) = (x \cdot w)a \text{ for } w \in \Sigma^*, a \in \Sigma$$

Number of  $c$ 's in a string

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

$$\#_c(wc) = \#_c(w) + 1 \text{ for } w \in \Sigma^*;$$

$$\#_c(wa) = \#_c(w) \text{ for } w \in \Sigma^*, a \in \Sigma \setminus \{c\}. \quad ]$$



# Functions on Strings

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

Length:

$$\text{len}(\varepsilon) = 0;$$

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

Reversal:

$$\varepsilon^R = \varepsilon;$$

$$(wa)^R = aw^R \text{ for } w \in \Sigma^*, a \in \Sigma$$

Concatenation

$$x \cdot \varepsilon = x \text{ for all } x \in \Sigma^*;$$

$$x \cdot (wa) = (x \cdot w)a \text{ for } w \in \Sigma^*, a \in \Sigma$$

Number of  $c$ 's in a string

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

$$\#_c(wc) = \#_c(w) + 1 \text{ for } w \in \Sigma^*;$$

$$\#_c(wa) = \#_c(w) \text{ for } w \in \Sigma^*, a \in \Sigma \setminus \{c\}.$$

# 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:  $6, 15$        $3|6$ ,  $3|15$       b/c  $3 * 2 = 6$ ,  $3 * 5 = 15$  ✓

Inductive Hypothesis: Suppose  $P$  holds for some <sup>arb.</sup>  $x, y \in S$

Inductive Step: We want to show  $P$  holds for  $x+y$

$3|x$ ,  $3|y$  by IH

$3 \cdot k = x$      $3 \cdot l = y$

$3(k+l) = x+y$

Which implies  $3|x+y$ .

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:

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.

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

Let  $P(y)$  be " $\text{len}(x \cdot y) = \text{len}(x) + \text{len}(y)$  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 inductive step!



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

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

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

Base Case:  $y = \epsilon$ .

Inductive Hypothesis:

Inductive Step:

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

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

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

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

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

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

$P(\epsilon)$  is " $\text{len}(x \cdot \epsilon) = \text{len}(x) + \text{len}(\epsilon)$ " for all  $x \in \Sigma^*$

Inductive Hypothesis: Suppose  $P(w)$

Inductive Step: Let  $x$  be an arbitrary string.

Therefore,  $\text{len}(xwa) = \text{len}(x) + \text{len}(wa)$   $\Sigma^*$ : Basis:  $\epsilon \in \Sigma^*$ .

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

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

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

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

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

Inductive Hypothesis: Suppose  $P(w)$   $\rightarrow$

Inductive Step: Let  $x$  be an arbitrary string.

$\text{len}(xwa) = \text{len}(xw) + 1$  (by definition of  $\text{len}$ )

$= \text{len}(x) + \text{len}(w) + 1$  (by IH)

$= \text{len}(x) + \text{len}(wa)$  (by definition of  $\text{len}$ )

Therefore,  $\text{len}(xwa) = \text{len}(x) + \text{len}(wa)$

for an arbitrary  $w \in \Sigma^*$   
"previous"  $y = w$   
 $y = wa$  for arbitrary  $a \in \Sigma$

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

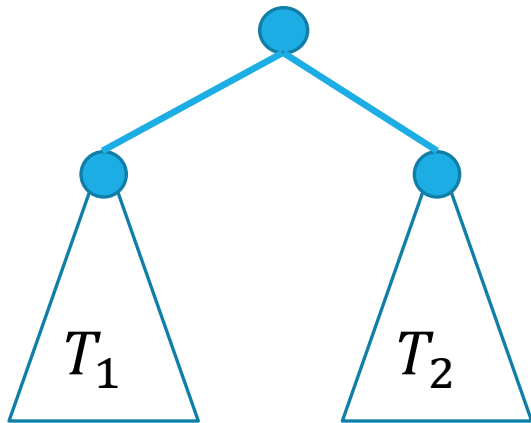
Recursive: If  $w \in \Sigma^*$  and  $a \in \Sigma$  then  $wa \in \Sigma^*$

# More Structural Sets

Binary Trees are another common source of structural induction.

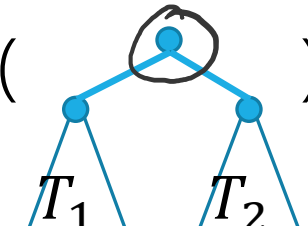
Basis: A single node is a rooted binary tree. 

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.

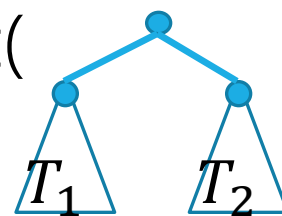


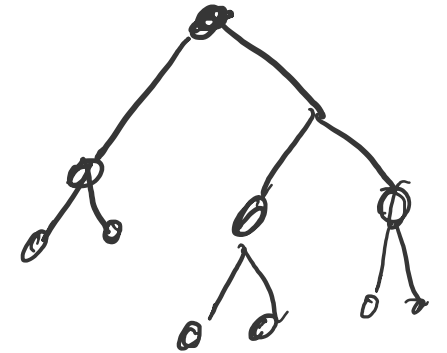
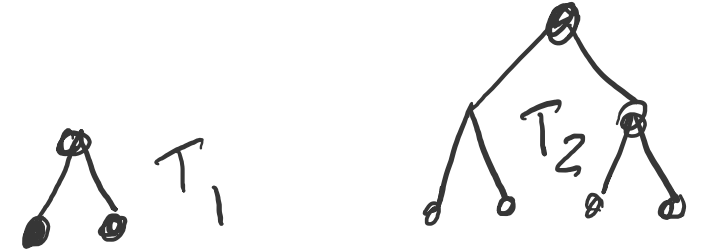
# Functions on Binary Trees

$$\text{size}(\bullet) = 1$$

$$\text{size}\left(\begin{array}{c} \bullet \\ / \quad \backslash \\ T_1 \quad T_2 \end{array}\right) = \text{size}(T_1) + \text{size}(T_2) + 1$$
A diagram of a binary tree with a root node and two children, labeled  $T_1$  and  $T_2$ . The root node is circled in blue.

$$\text{height}(\bullet) = 0$$

$$\text{height}\left(\begin{array}{c} \bullet \\ / \quad \backslash \\ T_1 \quad T_2 \end{array}\right) = 1 + \max(\text{height}(T_1), \text{height}(T_2))$$
A diagram of a binary tree with a root node and two children, labeled  $T_1$  and  $T_2$ .



# Structural Induction on Binary Trees

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

We'll show this next time.