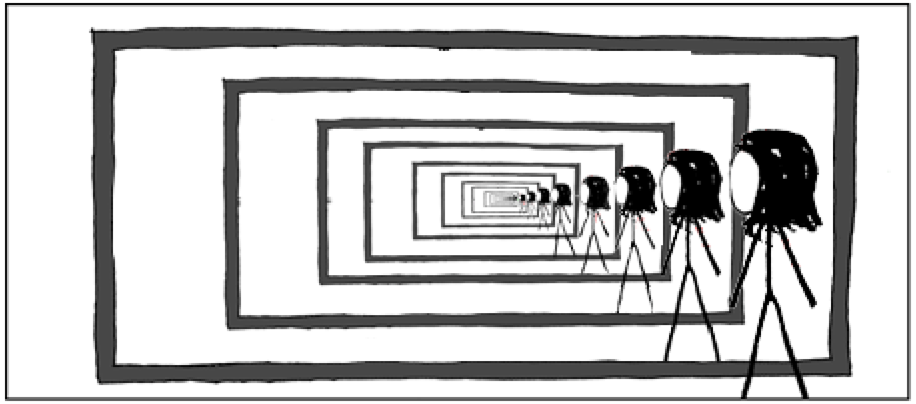
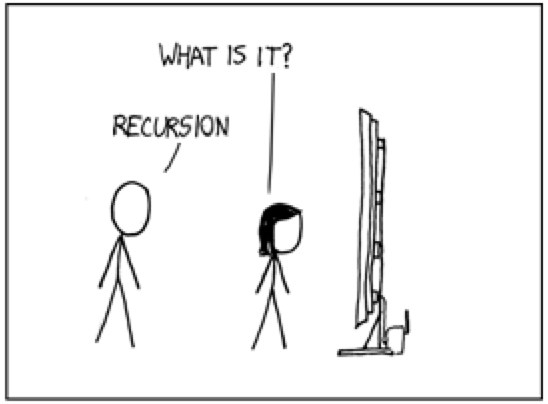
Warm up:

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 Winter 24

Lecture 16

Announcements

Don’t forget:

You can bring a handwritten 8.5x11 piece of paper (both sides) with notes to the exam.

Review session CSE2 G20 Saturday at 1.

Day of the midterm, we’ll be here to answer questions, but have nothing prepared.

Today’s CC is due Wednesday (since Monday isn’t a ‘real lecture’).

Induction Big Picture

So far: We used induction to prove a statement over the natural numbers.

"Prove that P(n) holds for all natural numbers n."

Next goal: In CS, we deal with Strings, Lists, Trees, and other recursively defined sets. Would like to prove statements over these sets.

"Prove that P(T) holds for all trees T."

"Prove that P(x) holds for all strings x."

Recursive Definition of Sets

Is $0 \in S$?

$0 \in S$!

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?

$\{0, 2, 4, 6, \ldots\}$

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

All Natural Numbers

Basis Step: $0 \in S$

Recursive Step: If $x \in S$ then $x + 1 \in S$.

All Integers

Basis Step: $0 \in S$

Recursive Step: If $x \in S$ then $x + 1 \in S$ and $x - 1 \in S$.

Integer coordinates in the line $y = x$

Basis Step: $(0,0) \in S$

Recursive Step: If $(x, y) \in S$ then $(x + 1, y + 1) \in S$ and $(x - 1, y - 1) \in S$.

Recursive Definitions of Sets

{6, 12, 15, 18, 21, 24, ... }

Q1: What is this set?

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

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

6, 15 {≥ 15 multiple of 3 and ≥ 12 or = 6}.

Q2: Write a recursive definition for the set of powers of 3 $\{1,3,9,27,\ldots\}$

Basis Step: 1 ∈ T, 9 ∈ T

Recursive Step: If x ∈ T then 3x ∈ T.

Structural Induction

$\forall s \in S(P(s))$

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 for an arbitrary element of the set, if $P()$ holds for that element then $P()$ holds for everything you can make out of it.

Structural Induction Example

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

$P(x+y)$

Show that every element of S is divisible by 3.

Structural Induction

$P(6)?$. $P(15)?$.

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

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 $x, y \in S$.

Inductive Step: By IH: $3|x$, $3|y$

$3\ell = x+y$

$3|(x+y)$

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

$\to P(x)$ for all x in S

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

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

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

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

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 $x, y \in S$.

Inductive Step: By IH $3|\mathrm{x}$ and $3|\mathrm{y}$. 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 $P(x) \forall x \in \mathrm{S}$ by the principle of induction.

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

Structural Induction Template

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

2. Base Case: Show $P(b)$

   [Do that for every b in the basis step of defining S]

3. Inductive Hypothesis: Suppose $P(x)$

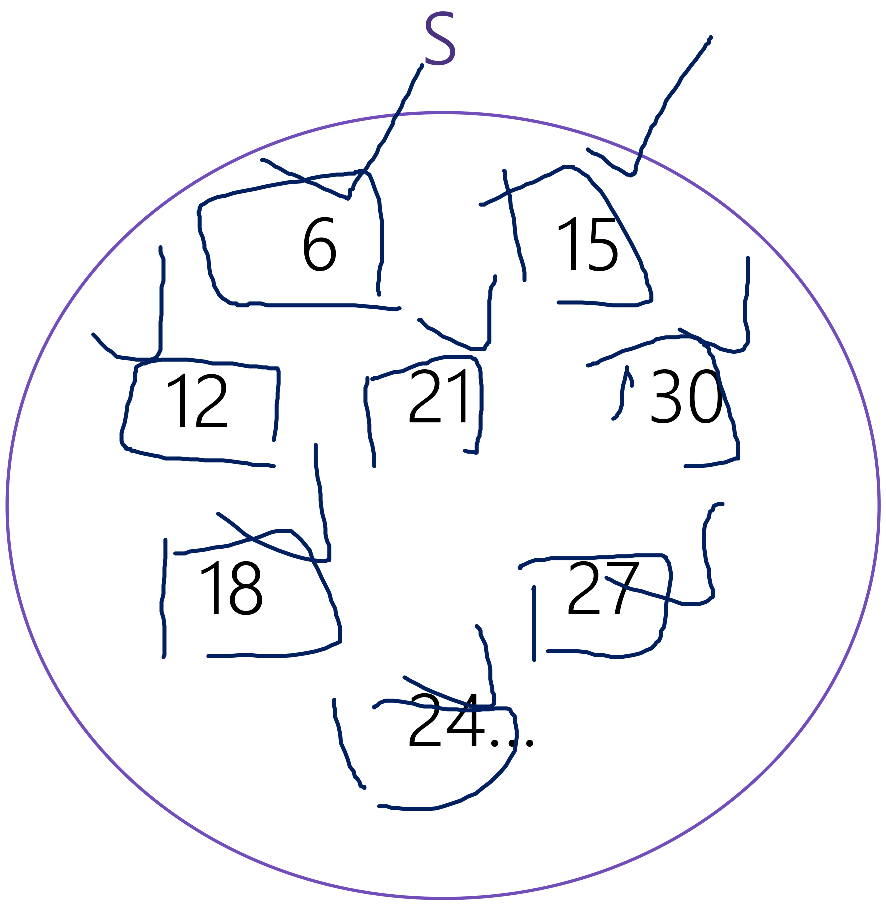
   [Do that for every x listed as already in S in the recursive rules].

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

   [You will need a separate step for every element created by the recursive rules].

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

Wait a minute! Why can we do this?



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

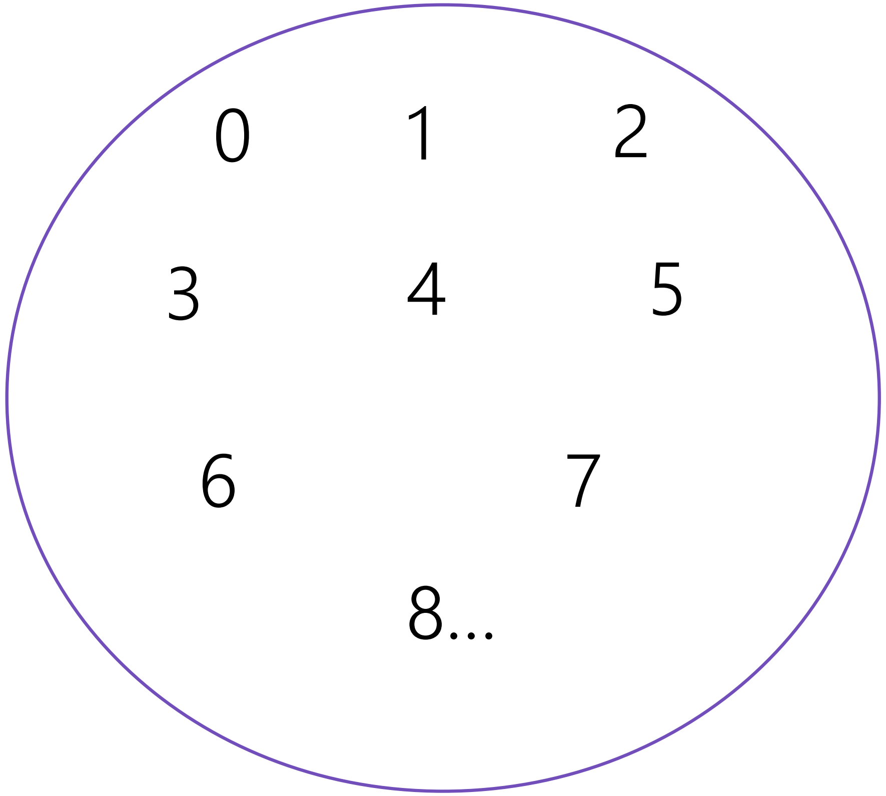
We proved:

Base Case: P(6) and P(15)

IH → IS: If P(x) and P(y), then P(x+y)

Weak Induction is a special case of Structural

$\mathbb{N}$



Basis: $\mathbf{0} \in \mathbb{N}$
Recursive: if $k \in \mathbb{N}$ then $k + 1 \in \mathbb{N}$.

We proved:

Base Case: P(0)

IH → IS: If P(k), then P(k+1)

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"

$S_0 = \{6,15\}$

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

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

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.

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~~ Σ 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

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 $w \in \Sigma^*, a \in \Sigma \setminus \{c\}$.

Σ^*

set of all strings

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 $w \in \Sigma^*, a \in \Sigma \setminus \{c\}$.

A string proof

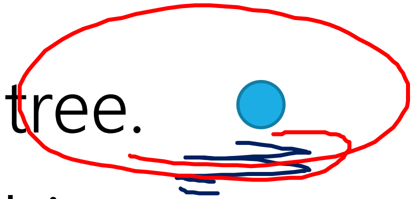
You'll do a string-based induction proof on the concept check. We'll do another one next week in lecture. It's got a lot of details that are worth going through slowly.

Let's do a different set—trees!

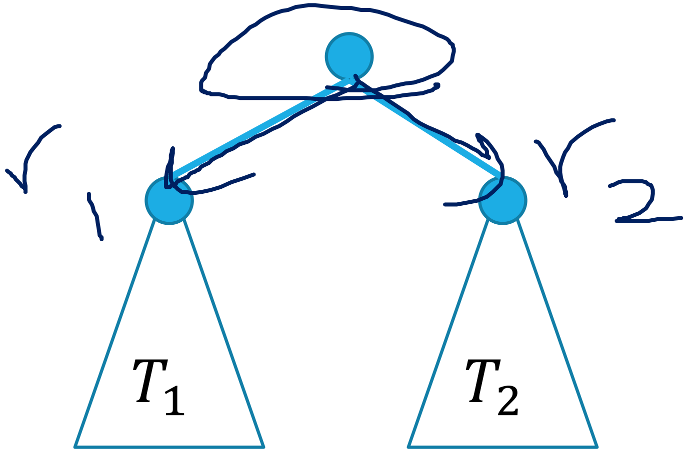
Trees!

More Structural Sets

Binary Trees are another common source of structural induction.

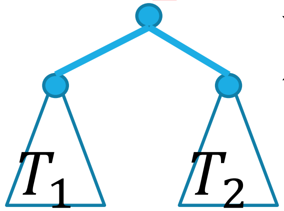
Basis: A single node is a rooted binary tree. 

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

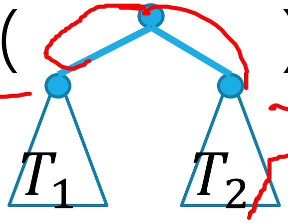
size(●)=1

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

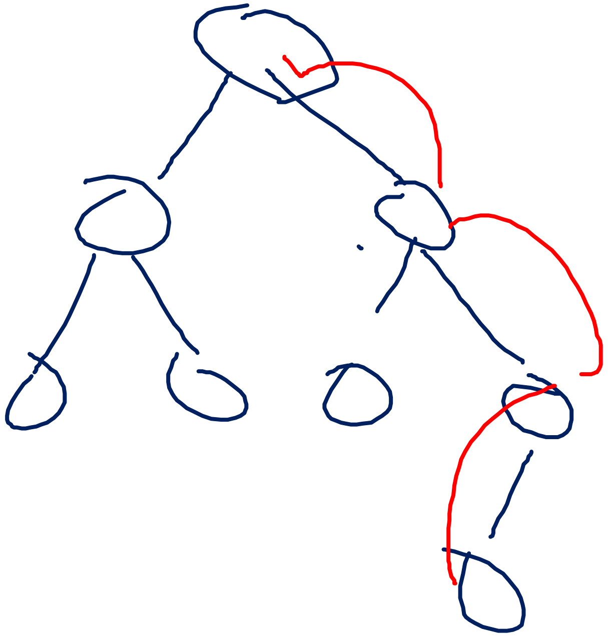
\# nodes

2

height(●) = 0

height() = 1+max(height(T_1),height(T_2))

$h(T_1)$? $h(T_2)$



Claim

We want to show that trees of a certain height can't have too many nodes. Specifically our claim is this:

For all trees T, size$(T) \leq 2^{height(T)+1} - 1$

Take a moment to absorb this formula, then we'll do induction!

1

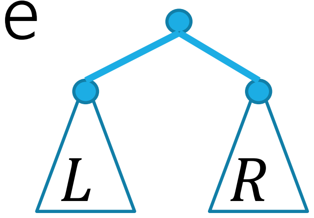
$2^{0+1} - 1 = 2^1 - 1 = 1$

Structural Induction on Binary Trees

Let $P(T)$ be “size(T) $\leq 2^{height(T)+1} - 1$“. We show $P(T)$ for all binary trees T by structural induction.

Base Case: Let $T = \bullet$. size(T)=1 and height(T) = 0, so size(T)=1$\leq 2 - 1 = 2^{0+1} - 1 = 2^{height(T)+1} - 1$.

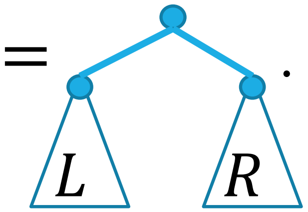
Inductive Hypothesis: Suppose $\mathrm{P}(L)$ and $\mathrm{P}(R)$ hold for arbitrary trees L, R. Let T be the tree



Inductive step: Figure out, (1) what we must show (2) a formula for height and a formula for size of T.

Structural Induction on Binary Trees (cont.)

Let $P(T)$ be “size(T) $\leq 2^{height(T)+1} - 1$“. We show $P(T)$ for all binary trees T by structural induction.

$T =$ 

.

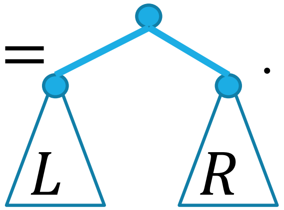
height(T)=$1 + \max\{height(L), height(R)\}$

size(T)= 1 +size(L)+size(R)

So $P(T)$ holds, and we have $P(T)$ for all binary trees T by the principle of induction.

Structural Induction on Binary Trees (cont.)

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

$T =$  .

height(T)=1 $+ \max\{height(L), height(R)\}$

size(T)= $1 +$size(L)+size(R)

size(T)=1+size(L)+size(R) $\leq 1 + 2^{height(L)+1} - 1 + 2^{height(R)+1} - 1$ (by IH)

$\leq 2^{height(L)+1} + 2^{height(R)+1} - 1$ (cancel 1's)

$\leq 2^{height(T)} + 2^{height(T)} - 1 = 2^{height(T)+1} - 1$ (T taller than subtrees)

So $P(T)$ holds, and we have $P(T)$ for all binary trees T by the principle of induction.

Structural Induction on Strings

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

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

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

Base Case:

Inductive Hypothesis

Inductive Step:

We conclude that $P(y)$ holds for all string y by the principle of induction. Unwrapping the definition of P, we get $\forall x \forall y \in \Sigma^*$ len(xy)=len(x)+len(y), as required.

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

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

We prove $P(y)$ for all $x \in \Sigma^*$ 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 $P(w)$ for an arbitrary string w.

Inductive Step:

We conclude that $P(y)$ holds for all string y by the principle of induction. Unwrapping the definition of P, we get $\forall x \forall y \in \Sigma^*$ len(xy)=len(x)+len(y), as required.

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

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

We prove $P(y)$ for all $x \in \Sigma^*$ 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 $P(w)$ for an arbitrary string w.

Inductive Step: Let $y = wa$ for an arbitrary $a \in \Sigma$. We show $P(y)$. Let x be an arbitrary string.

…

Therefore, len(xy)=len(x) + len(y), as required.

We conclude that $P(y)$ holds for all string y by the principle of induction. Unwrapping the definition of P, we get $\forall x \forall y \in \Sigma^*$ len(xy)=len(x)+len(y), as required.

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

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

We prove $P(y)$ for all $x \in \Sigma^*$ 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 $P(w)$ for an arbitrary string w.

Inductive Step: Let $y = wa$ for an arbitrary $a \in \Sigma$. We show $P(y)$. Let x be an arbitrary string.

len(xy)=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(xy)=len(x) + len(y), as required.

We conclude that $P(y)$ holds for all string y by the principle of induction. Unwrapping the definition of P, we get $\forall x \forall y \in \Sigma^*$ len(xy)=len(x)+len(y), as required.