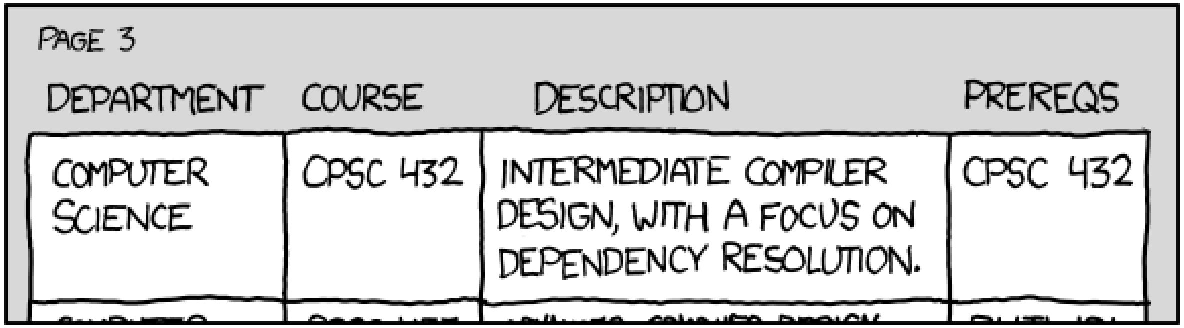
CSE 331
Spring 2025

Reasoning Extras



xkcd #754

Matt Wang

& Ali, Alice, Andrew, Anmol, Antonio, Connor, Edison, Helena, Jonathan, Katherine, Lauren, Lawrence, Mayee, Omar, Riva, Saan, and Yusong

Proof By Cases

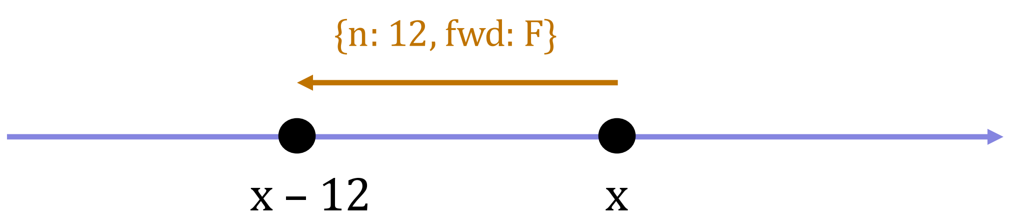
Recall: Pattern Matching

- **Define a function by an exhaustive set of patterns**

$$\textbf{type}\ \mathrm{Steps} := \{\mathrm{n} : \mathbb{N}, \mathrm{fwd} : \mathbb{B}\}$$

$$\mathrm{change}(\{\mathrm{n}: \mathrm{n}, \mathrm{fwd}: \mathrm{T}\}) := \mathrm{n}$$
$$\mathrm{change}(\{\mathrm{n}: \mathrm{n}, \mathrm{fwd}: \mathrm{F}\}) := -\mathrm{n}$$

  - Steps **describes movement on the number line**
  - $\mathrm{change}(\mathrm{s} : \mathrm{Steps})$ **says how the position changes**



  - **one of these two rules always applies**

Proof by Cases, with Records (Case T)

$\text{change}(\{n{:}\ n, \text{fwd}{:}\ T\}) \;:=\; n$

$\text{change}(\{n{:}\ n, \text{fwd}{:}\ F\}) \;:=\; -n$

- **Prove that** $|\text{change}(s)| = n$ **for any** $s = \{n{:}\ n, \text{fwd}{:}\ f\}$
  - **we need to know if** $f = T$ **or** $f = F$ **to apply the definition!**

Case $f = T$:

$\lvert\text{change}(\{n{:}\ n, \text{fwd}{:}\ f\})\rvert$	
$= \lvert\text{change}(\{n{:}\ n, \text{fwd}{:}\ T\})\rvert$	**since** $f = T$
$= \lvert n\rvert$	**def of** change
$= n$	**since** $n \geq 0$

Proof by Cases, with Records (Case F)

$\text{change}(\{\text{n: n, fwd: T}\}) \ := \ \text{n}$

$\text{change}(\{\text{n: n, fwd: F}\}) \ := \ \text{-n}$

- **Prove that** $|\text{change}(s)| = n$ **for any** $s = \{\text{n: n, fwd: f}\}$

Case $f = T$: $|\text{change}(\{\text{n: n, fwd: f}\})| = \ldots = n$

Case $f = F$:

$$
\begin{aligned}
&|\text{change}(\{\text{n: n, fwd: f}\})| \\
&= |\text{change}(\{\text{n: n, fwd: F}\})| && \textbf{since } f = F \\
&= |\text{-n}| && \textbf{def of } \text{change} \\
&= n && \textbf{since } n \geq 0
\end{aligned}
$$

Since these two cases are exhaustive, the claim holds in general.

Structural Induction

Definition of List Reversal

- Reversal of a List: “same values but in reverse order”

- Look at some examples…

L	rev(L)	
nil	nil	
[3]	[3]	3 :: nil
[2, 3]	[3, 2]	3 :: 2 :: nil
[1, 2, 3]	[3, 2, 1]	3 :: 2 :: 1 :: nil
…	…	

Structural Recursion in List Reversal

- Look at some examples…

L	$\mathrm{rev}(L)$
nil	nil
$3 :: \mathrm{nil}$	$3 :: \mathrm{nil}$
$2 :: 3 :: \mathrm{nil}$	$3 :: 2 :: \mathrm{nil}$
$1 :: 2 :: 3 :: \mathrm{nil}$	$3 :: 2 :: 1 :: \mathrm{nil}$

- **Where does** $\mathrm{rev}([2, 3])$ **show up in** $\mathrm{rev}([1, 2, 3])$**?**
  - at the beginning, with $1 :: \mathrm{nil}$ *after* it

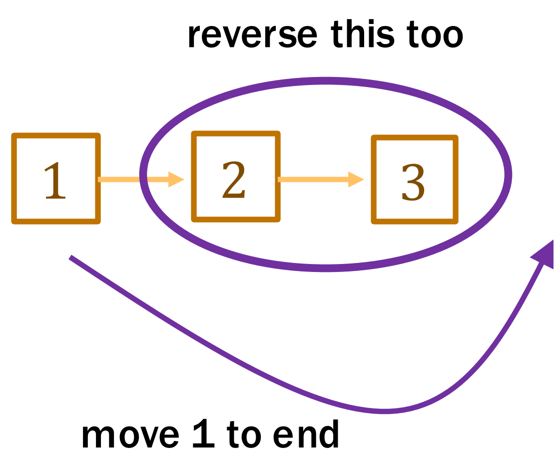
- **Where does** $\mathrm{rev}([3])$ **show up in** $\mathrm{rev}([2, 3])$**?**
  - at the beginning, with $2 :: \mathrm{nil}$ *after* it

Recall: Reversing a List

- **Mathematical definition of** $\mathrm{rev}(S)$

$$\begin{aligned} \mathrm{rev}(\mathsf{nil}) &:= \mathrm{nil} \\ \mathrm{rev}(x :: L) &:= \mathrm{rev}(L) \mathbin{+\!\!\!+} [x] \end{aligned}$$

  – **note that** rev **uses** concat ($\mathbin{+\!\!\!+}$) **as a helper function**



Definition of List Reversal: Checking Examples

$1 :: 2 :: 3 :: \text{nil}$ $\qquad\qquad$ $3 :: 2 :: 1 :: \text{nil}$

- **Mathematical definition of** $\text{rev} : \text{List} \to \text{List}$

$$
\begin{aligned}
\text{rev}(\text{nil}) &:= \text{nil} \\
\text{rev}(x :: L) &:= \text{rev}(L) +\!\!\!+ [x]
\end{aligned}
$$

- **Check that this matches examples…**

$$
\begin{aligned}
&\text{rev}(1 :: 2 :: 3 :: \text{nil}) \\
&\quad = \text{rev}(2 :: 3 :: \text{nil}) +\!\!\!+ [1] && \text{def of rev} \\
&\quad = \text{rev}(3 :: \text{nil}) +\!\!\!+ [2] +\!\!\!+ [1] && \text{def of rev} \\
&\quad = \text{rev}(\text{nil}) +\!\!\!+ [3] +\!\!\!+ [2] +\!\!\!+ [1] && \text{def of rev} \\
&\quad = [\,] +\!\!\!+ [3] +\!\!\!+ [2] +\!\!\!+ [1] && \text{def of rev} \\
&\quad = \ldots = [3, 2, 1] && \text{def of concat (many times)}
\end{aligned}
$$

Example 5: Length of Reversed List: Setup

$$\begin{aligned} \text{rev}(\text{nil}) &:= \text{nil} \\ \text{rev}(x :: L) &:= \text{rev}(L) \mathbin{+\!\!+} [x] \end{aligned}$$

- **Suppose we have the following code:**

```
const m = len(S);        // S is some List
const R = rev(S);
…
return m;  // = len(rev(S))
```

  – **spec returns** $\text{len}(\text{rev}(S))$ **but code returns** $\text{len}(S)$

- **Need to prove that** $\text{len}(\text{rev}(S)) = \text{len}(S)$ **for any** $S : \text{List}$

Example 5: Length of Reversed List (1/3)

$$\begin{aligned} \text{rev}(\text{nil}) \quad &:= \text{nil} \\ \text{rev}(x :: L) \quad &:= \text{rev}(L) \mathbin{+\!\!+} [x] \end{aligned}$$

- **Prove that** $\text{len}(\text{rev}(S)) = \text{len}(S)$ **for any** $S : \text{List}$

Base Case (nil):

$\text{len}(\text{rev}(\text{nil})) = \text{len}(\text{nil})$ **def of** rev

Inductive Step $(\text{cons}(x, L))$**:**

Need to prove that $\text{len}(\text{rev}(x :: L)) = \text{len}(x :: L)$

Get to assume that $\text{len}(\text{rev}(L)) = \text{len}(L)$

Example 5: Length of Reversed List (2/3)

$$\begin{aligned} \mathrm{rev}(\mathrm{nil}) &:= \mathrm{nil} \\ \mathrm{rev}(x :: L) &:= \mathrm{rev}(L) \mathbin{+\!\!+} [x] \end{aligned}$$

- **Prove that** $\mathrm{len}(\mathrm{rev}(S)) = \mathrm{len}(S)$ **for any** $S : \mathrm{List}$

Inductive Hypothesis: assume that $\mathrm{len}(\mathrm{rev}(L)) = \mathrm{len}(L)$

Inductive Step $(x :: L)$**:**

$$\mathrm{len}(\mathrm{rev}(x :: L))$$
$$=$$

$$= \mathrm{len}(x :: L)$$

Example 5: Length of Reversed List (3/3)

$$\begin{aligned} \mathrm{rev}(\mathrm{nil}) &:= \mathrm{nil} \\ \mathrm{rev}(\mathrm{x} :: \mathrm{L}) &:= \mathrm{rev}(\mathrm{L}) + [\mathrm{x}] \end{aligned}$$

- **Prove that** $\mathrm{len}(\mathrm{rev}(\mathrm{S})) = \mathrm{len}(\mathrm{S})$ **for any** $\mathrm{S} : \mathrm{List}$

Inductive Hypothesis: assume that $\mathrm{len}(\mathrm{rev}(\mathrm{L})) = \mathrm{len}(\mathrm{L})$

Inductive Step $(\mathrm{x} :: \mathrm{L})$**:**

$\mathrm{len}(\mathrm{rev}(\mathrm{x} :: \mathrm{L})$

$= \mathrm{len}(\mathrm{rev}(\mathrm{L}) + [\mathrm{x}])$	**def of** rev
$= \mathrm{len}(\mathrm{rev}(\mathrm{L})) + \mathrm{len}([\mathrm{x}])$	**by Example 3**
$= \mathrm{len}(\mathrm{L}) + \mathrm{len}([\mathrm{x}])$	**Ind. Hyp.**
$= \mathrm{len}(\mathrm{L}) + 1 + \mathrm{len}(\mathrm{nil})$	**def of** len
$= \mathrm{len}(\mathrm{L}) + 1$	**def of** len
$= \mathrm{len}(\mathrm{x} :: \mathrm{L})$	**def of** len

Finer Points of Structural Induction

- Structural Induction is how we reason about recursion

- Reasoning also follows structure of code
  - code uses structural recursion, so reasoning uses structural induction

- Note that rev is defined in terms of concat
  - reasoning about $\mathrm{len}(\mathrm{rev}(\ldots))$ used fact about $\mathrm{len}(\mathrm{concat}(\ldots))$
  - this is common

Example 6: Reversing a List Performance

$$\begin{aligned} &\mathrm{rev}(\mathrm{nil}) && := \mathrm{nil} \\ &\mathrm{rev}(x :: L) && := \mathrm{rev}(L) \mathbin{+\!\!+} [x] \end{aligned}$$

- This correctly reverses a list but is slow
  - concat takes $\Theta(n)$ time, where n is length of L
  - n calls to concat takes $\Theta(n^2)$ time
- Can we do this faster?
  - yes, but we need a helper function

Example 6: Reversing a List, Linear Time (1/3)

- **Helper function** $\text{rev-acc}(S, R)$ **for any** $S, R : \text{List}$

$$\begin{aligned} \text{rev-acc}(\text{nil}, R) &:= R \\ \text{rev-acc}(x :: L, R) &:= \text{rev-acc}(L, x :: R) \end{aligned}$$

$$\text{rev-acc}\left(\boxed{3} \to \boxed{4} \to \text{nil}, \; \boxed{2} \to \boxed{1} \to \text{nil}\right)$$

Example 6: Reversing a List, Linear Time (2/3)

- **Helper function** $\text{rev-acc}(S, R)$ **for any** $S, R : \text{List}$

$$\begin{aligned}
\text{rev-acc}(\text{nil}, R) &:= R \\
\text{rev-acc}(x :: L, R) &:= \text{rev-acc}(L, x :: R)
\end{aligned}$$

$$\text{rev-acc}\left(\boxed{3} \to \boxed{4} \to \text{nil}\,,\ \boxed{2} \to \boxed{1} \to \text{nil}\right)$$

$$= \text{rev-acc}\left(\boxed{4} \to \text{nil}\,,\ \boxed{3} \to \boxed{2} \to \boxed{1} \to \text{nil}\right)$$

Example 6: Reversing a List

- **Helper function** rev-acc(S, R) **for any** S, R : List

$$\begin{aligned}\text{rev-acc}(\text{nil}, R) &:= R \\ \text{rev-acc}(x :: L, R) &:= \text{rev-acc}(L, x :: R)\end{aligned}$$

$$\text{rev-acc}\left(\boxed{3} \to \boxed{4} \to \text{nil}\ ,\ \boxed{2} \to \boxed{1} \to \text{nil}\right)$$

$$= \text{rev-acc}\left(\boxed{4} \to \text{nil}\ ,\ \boxed{3} \to \boxed{2} \to \boxed{1} \to \text{nil}\right)$$

$$= \text{rev-acc}\left(\text{nil}\ ,\ \boxed{4} \to \boxed{3} \to \boxed{2} \to \boxed{1} \to \text{nil}\right)$$

Proving that rev-acc works, in pieces

$$\begin{aligned} \text{rev-acc}(\text{nil}, \text{R}) \quad &:= \ \text{R} \\ \text{rev-acc}(\text{x} :: \text{L}, \text{R}) \quad &:= \ \text{rev-acc}(\text{L}, \text{x} :: \text{R}) \end{aligned}$$

- **Can prove that** $\text{rev-acc}(\text{S}, \text{R}) = \text{concat}(\text{rev}(\text{S}), \text{R})$ **(Lemma 1)**
- **Can prove that** $\text{concat}(\text{L}, \text{nil}) = \text{L}$ **(Lemma 2)**
  - **structural induction like prior examples**
- **Prove that** $\text{rev}(\text{S}) = \text{rev-acc}(\text{S}, \text{nil})$

$$\begin{aligned} \text{rev-acc}(\text{S}, \text{nil}) \quad &= \text{concat}(\text{rev}(\text{S}), \text{nil}) && \textbf{Lemma 1} \\ &= \text{rev}(\text{S}) && \textbf{Lemma 2} \end{aligned}$$

Proving Lemma 2: Setup

$$\text{rev-acc}(\text{nil}, R) \quad := \quad R$$
$$\text{rev-acc}(x :: L, R) \quad := \quad \text{rev-acc}(L, x :: R)$$

- **Prove that** $\text{concat}(S, \text{nil}) = S$

Base Case (nil):

$\text{concat}(\text{nil}, \text{nil}) \quad = \text{nil}$ **def of** concat

Inductive Hypothesis: **assume that** $\text{concat}(L, \text{nil}) = \text{nil}$

Inductive Step $(\text{cons}(x, L))$: **prove that** $\text{concat}(\text{cons}(x, L), \text{nil}) = \text{cons}(x, L)$

Proving Lemma 2: Inductive Step (1/2)

$$\text{rev-acc}(\text{nil}, R) \quad := \quad R$$
$$\text{rev-acc}(x :: L, R) \quad := \quad \text{rev-acc}(L, x :: R)$$

- **Prove that** $\text{concat}(S, \text{nil}) = S$

Inductive Hypothesis: **assume that** $\text{concat}(L, \text{nil}) = L$

Inductive Step $(x :: L)$:

$$\text{concat}(x :: L, \text{nil}) \quad =$$

$$= x :: L$$

Proving Lemma 2: Inductive Step (2/2)

$$\begin{aligned} \text{rev-acc}(\text{nil}, R) &:= R \\ \text{rev-acc}(x :: L, R) &:= \text{rev-acc}(L, x :: R) \end{aligned}$$

- **Prove that** $\text{concat}(S, \text{nil}) = S$

Inductive Hypothesis: **assume that** $\text{concat}(L, \text{nil}) = L$

Inductive Step $(x :: L)$:

$\text{concat}(x :: L, \text{nil})$	$= x :: \text{concat}(L, \text{nil})$	**def of** concat
	$= x :: L$	**Ind. Hyp.**

Proving Lemma 1: Setup

$$\begin{aligned} &\text{rev-acc}(\text{nil}, \text{R}) &&:= \text{R} \\ &\text{rev-acc}(\text{x} :: \text{L}, \text{R}) &&:= \text{rev-acc}(\text{L}, \text{x} :: \text{R}) \end{aligned}$$

- **Prove that** $\text{rev-acc}(\text{S}, \text{R}) = \text{concat}(\text{rev}(\text{S}), \text{R})$
  - **prove by structural induction**

- **Need the following property of** $\text{concat}\ (⧺)$

$$\text{A} ⧺ (\text{B} ⧺ \text{C}) = (\text{A} ⧺ \text{B}) ⧺ \text{C}$$

  - **with strings, we know that "A + (B + C) = (A + B) + C"**
  - **this says the same thing for lists with "⧺"**

Proving Lemma 1: Base Case (1/2)

$$\begin{aligned} \text{rev-acc}(\text{nil}, \text{R}) &:= \text{R} \\ \text{rev-acc}(\text{x} :: \text{L}, \text{R}) &:= \text{rev-acc}(\text{L}, \text{x} :: \text{R}) \end{aligned}$$

- **Prove that** $\text{rev-acc}(\text{S}, \text{R}) = \text{concat}(\text{rev}(\text{S}), \text{R})$
  - **prove by induction on** S **(so** R **is a variable)**

Base Case (nil):

$$\begin{aligned} \text{rev-acc}(\text{nil}, \text{R}) &= \\ \\ &= \text{concat}(\text{rev}(\text{nil}), \text{R}) \end{aligned}$$

$$\begin{aligned} \text{concat}(\text{nil}, \text{R}) &:= \text{R} \\ \text{concat}(\text{x} :: \text{L}, \text{R}) &:= \text{x} :: \text{concat}(\text{L}, \text{R}) \end{aligned}$$

$$\begin{aligned} \text{rev}(\text{nil}) &:= \text{nil} \\ \text{rev}(\text{x} :: \text{L}) &:= \text{rev}(\text{L}) \mathbin{+\!\!+} [\text{x}] \end{aligned}$$

Proving Lemma 1: Base Case (2/2)

$$\begin{aligned} \text{rev-acc}(\text{nil}, R) &:= R \\ \text{rev-acc}(x :: L, R) &:= \text{rev-acc}(L, x :: R) \end{aligned}$$

- **Prove that** $\text{rev-acc}(S, R) = \text{concat}(\text{rev}(S), R)$
  - **prove by induction on** S **(so** R **is a variable)**

Base Case (nil):

$\text{rev-acc}(\text{nil}, R)$	$= R$	**def of** rev-acc
	$= \text{concat}(\text{nil}, R)$	**def of** concat
	$= \text{concat}(\text{rev}(\text{nil}), R)$	**def of** rev

$$\begin{aligned} \text{concat}(\text{nil}, R) &:= R \\ \text{concat}(x :: L, R) &:= x :: \text{concat}(L, R) \end{aligned}$$

$$\begin{aligned} \text{rev}(\text{nil}) &:= \text{nil} \\ \text{rev}(x :: L) &:= \text{rev}(L) \mathbin{+\!\!+} [x] \end{aligned}$$

Proving Lemma 1: Inductive Step (1/4)

$$\begin{aligned} \text{rev-acc}(\text{nil}, R) &:= R \\ \text{rev-acc}(x :: L, R) &:= \text{rev-acc}(L, x :: R) \end{aligned}$$

- **Prove that** $\text{rev-acc}(S, R) = \text{concat}(\text{rev}(S), R)$

Inductive Hypothesis: assume that $\text{rev-acc}(L, R) = \text{concat}(\text{rev}(L), R)$ **for any** R

Inductive Step $(x :: L)$**:**

$\text{rev-acc}(x :: L, R) \quad =$

$= \text{concat}(\text{rev}(x :: L), R)$

func $\text{concat}(\text{nil}, R) := R$ $\text{concat}(\text{cons}(x, L), R) := \text{cons}(x, \text{concat}(L, R))$	**func** $\text{rev}(\text{nil}) := \text{nil}$ $\text{rev}(\text{cons}(x, L)) := \text{concat}(\text{rev}(L), \text{cons}(x, \text{nil}))$

Proving Lemma 1: Inductive Step (2/4)

$$\begin{aligned} \text{rev-acc}(\text{nil}, R) &:= R \\ \text{rev-acc}(x :: L, R) &:= \text{rev-acc}(L, x :: R) \end{aligned}$$

- **Prove that** $\text{rev-acc}(S, R) = \text{concat}(\text{rev}(S), R)$

  Inductive Hypothesis: **assume that** $\text{rev-acc}(L, R) = \text{concat}(\text{rev}(L), R)$ **for any** R

  Inductive Step $(x :: L)$:

$\text{rev-acc}(x :: L, R)$	$= \text{rev-acc}(L, x :: R)$	**def of** rev-acc
	$= \text{concat}(\text{rev}(L), x :: R)$	**Ind. Hyp.**
	$= (\text{rev}(L) \mathbin{+\!\!+} [x]) \mathbin{+\!\!+} R$	??
	$= \text{concat}(\text{rev}(L) \mathbin{+\!\!+} [x], R)$	
	$= \text{concat}(\text{rev}(x :: L), R)$	**def of** rev

$$\begin{aligned} \text{concat}(\text{nil}, R) &:= R \\ \text{concat}(x :: L, R) &:= x :: \text{concat}(L, R) \end{aligned}$$

$$\begin{aligned} \text{rev}(\text{nil}) &:= \text{nil} \\ \text{rev}(x :: L) &:= \text{rev}(L) \mathbin{+\!\!+} [x] \end{aligned}$$

Proving Lemma 1: Inductive Step (3/4)

$$\begin{aligned} \text{rev-acc}(\text{nil}, R) &:= R \\ \text{rev-acc}(x :: L, R) &:= \text{rev-acc}(L, x :: R) \end{aligned}$$

- **Prove that** $\text{rev-acc}(S, R) = \text{concat}(\text{rev}(S), R)$

Inductive Hypothesis: assume that $\text{rev-acc}(L, R) = \text{concat}(\text{rev}(L), R)$ **for any** R

Inductive Step $(x :: L)$**:**

$\text{rev-acc}(x :: L, R)$	$= \text{rev-acc}(L, x :: R)$	**def of** rev-acc
	$= \text{concat}(\text{rev}(L), x :: R)$	**Ind. Hyp.**
	$= \text{rev}(L) \mathbin{+\!\!+} ([x] \mathbin{+\!\!+} R)$	
	$= (\text{rev}(L) \mathbin{+\!\!+} [x]) \mathbin{+\!\!+} R$	**assoc. of** $\mathbin{+\!\!+}$
	$= \text{concat}(\text{rev}(L) \mathbin{+\!\!+} [x], R)$	
	$= \text{concat}(\text{rev}(x :: L), R)$	**def of** rev

$$\begin{aligned} \text{concat}(\text{nil}, R) &:= R \\ \text{concat}(x :: L, R) &:= x :: \text{concat}(L, R) \end{aligned}$$

$$\begin{aligned} \text{rev}(\text{nil}) &:= \text{nil} \\ \text{rev}(x :: L) &:= \text{rev}(L) \mathbin{+\!\!+} [x] \end{aligned}$$

Proving Lemma 1: Inductive Step (4/4)

$$\begin{aligned} \text{rev-acc}(\text{nil}, R) &:= R \\ \text{rev-acc}(x :: L, R) &:= \text{rev-acc}(L, x :: R) \end{aligned}$$

- **Prove that** $\text{rev-acc}(S, R) = \text{concat}(\text{rev}(S), R)$

  Inductive Hypothesis: assume that $\text{rev-acc}(L, R) = \text{concat}(\text{rev}(L), R)$ **for any** R

  Inductive Step $(x :: L)$**:**

$\text{rev-acc}(x :: L, R)$	$= \text{rev-acc}(L, x :: R)$	**def of** rev-acc
	$= \text{concat}(\text{rev}(L), x :: R)$	**Ind. Hyp.**
	$= \text{rev}(L) \mathbin{+\!\!+} (x :: R)$	
	$= \text{rev}(L) \mathbin{+\!\!+} ([x] \mathbin{+\!\!+} R)$	**def of** concat
	$= (\text{rev}(L) \mathbin{+\!\!+} [x]) \mathbin{+\!\!+} R$	**assoc. of** $\mathbin{+\!\!+}$
	$= \text{concat}(\text{rev}(L) \mathbin{+\!\!+} [x], R)$	
	$= \text{concat}(\text{rev}(x :: L), R)$	**def of** rev

$$\begin{aligned} \text{concat}(\text{nil}, R) &:= R \\ \text{concat}(x :: L, R) &:= x :: \text{concat}(L, R) \end{aligned}$$

$$\begin{aligned} \text{rev}(\text{nil}) &:= \text{nil} \\ \text{rev}(x :: L) &:= \text{rev}(L) \mathbin{+\!\!+} [x] \end{aligned}$$

Structural Induction in General

- General case: assume P holds for constructor *arguments*

$$\textbf{type}\ \mathrm{T} := \mathrm{A} \mid \mathrm{B}(\mathrm{x} : \mathbb{Z}) \mid \mathrm{C}(\mathrm{y} : \mathbb{Z}, \mathrm{t} : \mathrm{T}) \mid \mathrm{D}(\mathrm{z} : \mathbb{Z}, \mathrm{u} : \mathrm{T}, \mathrm{v} : \mathrm{T})$$

- To prove $\mathrm{P}(\mathrm{t})$ for any t, we need to prove:
  - $\mathrm{P}(\mathrm{A})$
  - $\mathrm{P}(\mathrm{B}(\mathrm{x}))$ for any $\mathrm{x} : \mathbb{Z}$
  - $\mathrm{P}(\mathrm{C}(\mathrm{y}, \mathrm{t}))$ for any $\mathrm{y} : \mathbb{Z}$ and $\mathrm{t} : \mathrm{T}$ assuming $\mathrm{P}(\mathrm{t})$ is true
  - $\mathrm{P}(\mathrm{D}(\mathrm{z}, \mathrm{u}, \mathrm{v}))$ for any $\mathrm{z} : \mathbb{Z}$ and $\mathrm{u}, \mathrm{v} : \mathrm{T}$ assuming $\mathrm{P}(\mathrm{u})$ and $\mathrm{P}(\mathrm{v})$

- These four facts are enough to prove $\mathrm{P}(\mathrm{t})$ for any t
  - for each constructor, have proof that it produces an object satisfying P
  - more generally, each inductive type has its own form of induction