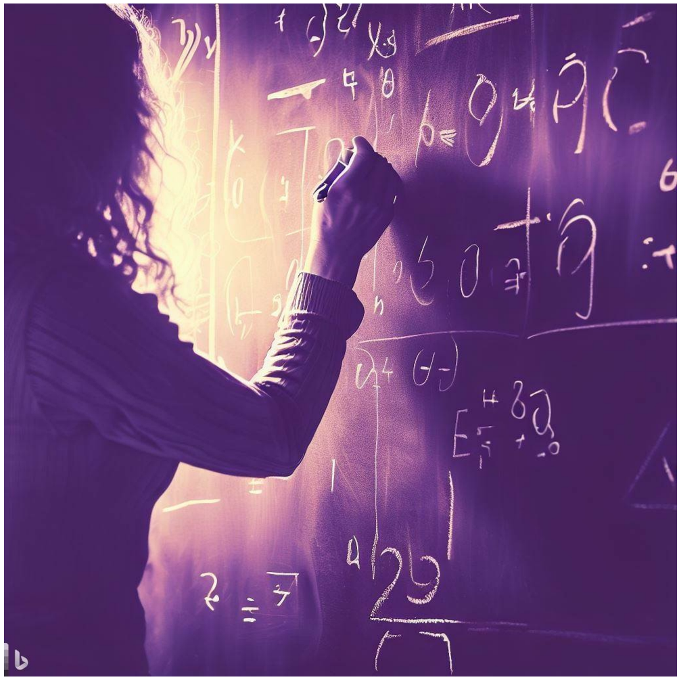


CSE 331

Structural Induction

Kevin Zatloukal

Proof by Calculation

- Our proofs so far have used fixed-length lists
  - e.g., $\text{len}(\text{twice}(\text{cons}(a, \text{cons}(b, \text{nil})))) = \text{len}(\text{cons}(a, \text{cons}(b, \text{nil})))$
  - problems in HW3 restrict to this case

- Would like to prove correctness on <u>any</u> list L

- Need more tools for this...
  - structural recursion *calculates* on inductive types
  - structural induction *reasons* about structural recursion

    or more generally, to prove facts containing variables of an inductive type
  - both tools are specific to inductive types

Structural Induction

Let $\mathrm{P}(\mathrm{S})$ **be the claim** "$\mathrm{len}(\mathrm{twice}(\mathrm{S})) = \mathrm{len}(\mathrm{S})$"

To prove $\mathrm{P}(\mathrm{S})$ **holds for <u>any</u> list** S**, prove two implications**

Base Case: prove $\mathrm{P}(\mathrm{nil})$

- **use any known facts and definitions**

Inductive Step: prove $\mathrm{P}(\mathrm{cons}(\mathrm{x}, \mathrm{L}))$ **for any** $\mathrm{x} : \mathbb{Z}$**,** $\mathrm{L} : \mathrm{List}$

- x **and** L **are variables ("direct proof")**
- **use any known facts and definitions plus <u>one more fact...</u>**
- **make use of the fact that** L **is also a** List

Structural Induction

To prove $\mathrm{P}(S)$ holds for any list S, prove two implications

Base Case: prove $\mathrm{P}(\mathrm{nil})$

- use any known facts and definitions

Inductive Hypothesis: assume $\mathrm{P}(\mathrm{L})$ is true

- use this in the inductive step, but not anywhere else

Inductive Step: prove $\mathrm{P}(\mathrm{cons}(\mathrm{x}, \mathrm{L}))$ for any $\mathrm{x} : \mathbb{Z}$, $\mathrm{L} : \mathrm{List}$

- direct proof
- use known facts and definitions and Inductive Hypothesis

Why This Works

With Structural Induction, we prove two facts

$\mathrm{P}(\mathrm{nil})$ $\qquad$ $\mathrm{len}(\mathrm{twice}(\mathrm{nil})) = \mathrm{len}(\mathrm{nil})$

$\mathrm{P}(\mathrm{cons}(\mathrm{x}, \mathrm{L}))$ $\qquad$ $\mathrm{len}(\mathrm{twice}(\mathrm{cons}(\mathrm{x}, \mathrm{L}))) = \mathrm{len}(\mathrm{cons}(\mathrm{x}, \mathrm{L}))$

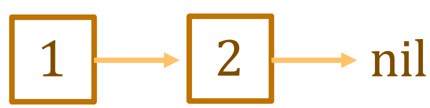
(**second assuming** $\mathrm{len}(\mathrm{twice}(\mathrm{L})) = \mathrm{len}(\mathrm{L})$)

Why is this enough to prove $\mathrm{P}(S)$ **for any** $S : \mathrm{List}$**?**

Why This Works

Build up an object using constructors:

nil	**first constructor**
cons(2, nil)	**second constructor**
cons(1, cons(2, nil))	**second constructor**



nil **already exists when building** cons(2, nil)

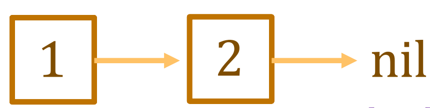
cons(2, nil) **already exists when building** cons(1, cons(2, nil))

Why This Works

Build up a proof the same way we built up the object

$\mathrm{P(nil)}$	$\mathrm{len(twice(nil)) = len(nil)}$
$\mathrm{P(cons(x, L))}$	$\mathrm{len(twice(cons(x, L))) = len(cons(x, L))}$

(**second assuming** $\mathrm{len(twice(L)) = len(L)}$)

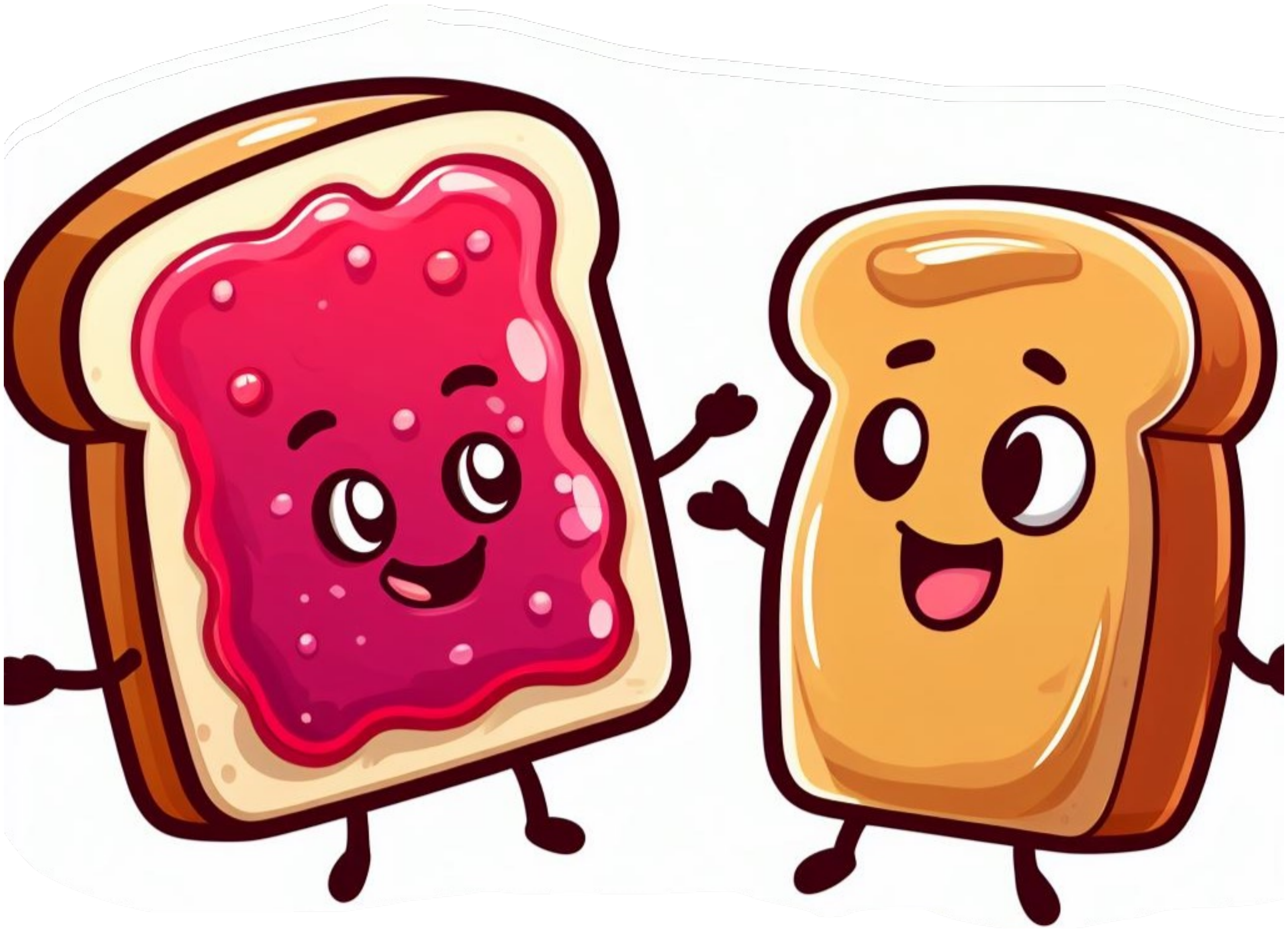


$\mathrm{P(nil)}$

$\mathrm{P(nil)}$ **already proven when proving** $\mathrm{P(cons(2, nil))}$

$\mathrm{P(cons(2, nil))}$ **already proven when proving** $\mathrm{P(cons(1, cons(2, nil)))}$

“We go together”



structural induction

inductive types

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**

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

- Each inductive type has its own form of induction
  - special way to reason about that type

Example: Repeating List Elements

- **Consider the following function:**

$$\begin{aligned} &\textbf{func}\ \text{echo}(\text{nil}) &&:= \text{nil} \\ &\quad\ \text{echo}(\text{cons}(x, L)) &&:= \text{cons}(x, \text{cons}(x, \text{echo}(L))) \quad \text{for any } x : \mathbb{Z}, L : \text{List} \end{aligned}$$

- **Produces a list where every element is repeated twice**

$$\begin{aligned} &\text{echo}(\text{cons}(1, \text{cons}(2, \text{nil}))) \\ &\quad = \text{cons}(1, \text{cons}(1, \text{echo}(\text{cons}(2, \text{nil})))) && \textbf{def of } \text{echo} \\ &\quad = \text{cons}(1, \text{cons}(1, \text{cons}(2, \text{cons}(2, \text{echo}(\text{nil}))))) && \textbf{def of } \text{echo} \\ &\quad = \text{cons}(1, \text{cons}(1, \text{cons}(2, \text{cons}(2, \text{nil})))) && \textbf{def of } \text{echo} \end{aligned}$$

Example: Repeating List Elements

$$
\begin{aligned}
&\textbf{func } \mathrm{echo}(\mathrm{nil}) && := \mathrm{nil} \\
&\quad\ \mathrm{echo}(\mathrm{cons}(x, L)) && := \mathrm{cons}(x, \mathrm{cons}(x, \mathrm{echo}(L))) \quad \text{for any } x : \mathbb{Z},\ L : \mathrm{List}
\end{aligned}
$$

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

```
const m: number = len(S);      // S is some List
const R: List = echo(S);
…
return 2*m;  // = len(echo(S))
```

Level 1

  - spec says to return $\mathrm{len}(\mathrm{echo}(S))$ but code returns $2\ \mathrm{len}(S)$

- **Need to prove that** $\mathrm{len}(\mathrm{echo}(S)) = 2\ \mathrm{len}(S)$

Example: Repeating List Elements

func echo(nil) := nil
echo(cons(x, L)) := cons(x, cons(x, echo(L))) for any x : $\mathbb{Z}$, L : List

- **Prove that** $\text{len}(\text{echo}(S)) = 2\ \text{len}(S)$ **for any** S : List

Base Case (nil):

Need to prove that $\text{len}(\text{echo}(\text{nil})) = 2\ \text{len}(\text{nil})$

$\text{len}(\text{echo}(\text{nil})) \quad =$

Example: Repeating List Elements

$$
\begin{aligned}
&\textbf{func}\ \mathrm{echo}(\mathrm{nil}) && := \mathrm{nil} \\
&\qquad \mathrm{echo}(\mathrm{cons}(x, L)) && := \mathrm{cons}(x, \mathrm{cons}(x, \mathrm{echo}(L))) \quad \text{for any } x : \mathbb{Z},\ L : \mathrm{List}
\end{aligned}
$$

- **Prove that** $\mathrm{len}(\mathrm{echo}(S)) = 2\ \mathrm{len}(S)$ **for any** $S : \mathrm{List}$

Base Case (nil):

$$
\begin{aligned}
\mathrm{len}(\mathrm{echo}(\mathrm{nil})) &= \mathrm{len}(\mathrm{nil}) && \textbf{def of}\ \mathrm{echo} \\
&= 0 && \textbf{def of}\ \mathrm{len} \\
&= 2 \cdot 0 && \\
&= 2\ \mathrm{len}(\mathrm{nil}) && \textbf{def of}\ \mathrm{len}
\end{aligned}
$$

Example: Repeating List Elements

$$\begin{aligned} &\textbf{func}\ \mathrm{echo}(\mathrm{nil}) && := \ \mathrm{nil} \\ &\qquad \mathrm{echo}(\mathrm{cons}(x, L)) && := \mathrm{cons}(x, \mathrm{cons}(x, \mathrm{echo}(L))) \quad \text{for any } x : \mathbb{Z},\ L : \mathrm{List} \end{aligned}$$

- **Prove that** $\mathrm{len}(\mathrm{echo}(S)) = 2\ \mathrm{len}(S)$ **for any** $S : \mathrm{List}$

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

Need to prove that $\mathrm{len}(\mathrm{echo}(\mathrm{cons}(x, L))) = 2\ \mathrm{len}(\mathrm{cons}(x, L))$

Get to assume claim holds for L, **i.e., that** $\mathrm{len}(\mathrm{echo}(L)) = 2\ \mathrm{len}(L)$

Example: Repeating List Elements

$$\begin{aligned} &\textbf{func}\ \mathrm{echo}(\mathrm{nil}) && := \ \mathrm{nil} \\ &\qquad \mathrm{echo}(\mathrm{cons}(x, L)) && := \mathrm{cons}(x, \mathrm{cons}(x, \mathrm{echo}(L))) \quad \text{for any } x : \mathbb{Z},\ L : \mathrm{List} \end{aligned}$$

- **Prove that** $\mathrm{len}(\mathrm{echo}(S)) = 2\ \mathrm{len}(S)$ **for any** $S : \mathrm{List}$

Inductive Hypothesis: assume that $\mathrm{len}(\mathrm{echo}(L)) = 2\ \mathrm{len}(L)$

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

$$\mathrm{len}(\mathrm{echo}(\mathrm{cons}(x, L)))$$

$$= 2\ \mathrm{len}(\mathrm{cons}(x, L))$$

Example: Repeating List Elements

$$
\begin{aligned}
&\textbf{func } \mathrm{echo}(\mathrm{nil}) && := \mathrm{nil} \\
&\quad \mathrm{echo}(\mathrm{cons}(x, L)) && := \mathrm{cons}(x, \mathrm{cons}(x, \mathrm{echo}(L))) \quad \text{for any } x : \mathbb{Z}, L : \mathrm{List}
\end{aligned}
$$

- **Prove that** $\mathrm{len}(\mathrm{echo}(S)) = 2\ \mathrm{len}(S)$ **for any** $S : \mathrm{List}$

Inductive Hypothesis: assume that $\mathrm{len}(\mathrm{echo}(L)) = 2\ \mathrm{len}(L)$

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

$$
\begin{aligned}
\mathrm{len}(\mathrm{echo}(\mathrm{cons}(x, L))) &= \mathrm{len}(\mathrm{cons}(x, \mathrm{cons}(x, \mathrm{echo}(L)))) && \textbf{def of } \mathrm{echo} \\
&= 1 + \mathrm{len}(\mathrm{cons}(x, \mathrm{echo}(L))) && \textbf{def of } \mathrm{len} \\
&= 2 + \mathrm{len}(\mathrm{echo}(L)) && \textbf{def of } \mathrm{len} \\
&= 2 + 2\ \mathrm{len}(L) && \textbf{Ind. Hyp.} \\
&= 2(1 + \mathrm{len}(L)) && \\
&= 2\ \mathrm{len}(\mathrm{cons}(x, L)) && \textbf{def of } \mathrm{len}
\end{aligned}
$$

Example 2: Repeating List Elements

$$\begin{aligned} \textbf{func } \mathrm{echo}(\mathrm{nil}) &:= \mathrm{nil} \\ \mathrm{echo}(\mathrm{cons}(x, L)) &:= \mathrm{cons}(x, \mathrm{cons}(x, \mathrm{echo}(L))) \quad \text{for any } x : \mathbb{Z}, L : \mathrm{List} \end{aligned}$$

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

```
const y: number = sum(S);      // S is some List
const R: List = echo(S);
…
return 2*y;  // = sum(echo(S))
```

Level 1

  - **spec says to return** $\mathrm{sum}(\mathrm{echo}(S))$ **but code returns** $2\ \mathrm{sum}(S)$

- **Need to prove that** $\mathrm{sum}(\mathrm{echo}(S)) = 2\ \mathrm{sum}(S)$

Example 2: Repeating List Elements

$$\textbf{func } \mathrm{echo}(\mathrm{nil}) := \mathrm{nil}$$
$$\mathrm{echo}(\mathrm{cons}(x, L)) := \mathrm{cons}(x, \mathrm{cons}(x, \mathrm{echo}(L))) \quad \text{for any } x : \mathbb{Z},\ L : \mathrm{List}$$

- **Prove that** $\mathrm{sum}(\mathrm{echo}(S)) = 2\ \mathrm{sum}(S)$ **for any** $S : \mathrm{List}$

Base Case (nil):

$$\mathrm{sum}(\mathrm{echo}(\mathrm{nil})) =$$

$$= 2\ \mathrm{sum}(\mathrm{nil})$$

$$\textbf{func } \mathrm{sum}(\mathrm{nil}) := 0$$
$$\mathrm{sum}(\mathrm{cons}(x, L)) := x + \mathrm{sum}(L) \qquad \text{for any } x \in \mathbb{Z} \text{ and any } L \in \mathrm{List}$$

Example 2: Repeating List Elements

$$\begin{aligned} &\textbf{func } \mathrm{echo}(\mathrm{nil}) &&:= \mathrm{nil} \\ &\quad \mathrm{echo}(\mathrm{cons}(x, L)) &&:= \mathrm{cons}(x, \mathrm{cons}(x, \mathrm{echo}(L))) \quad \text{for any } x : \mathbb{Z},\ L : \mathrm{List} \end{aligned}$$

- **Prove that** $\mathrm{sum}(\mathrm{echo}(S)) = 2\ \mathrm{sum}(S)$ **for any** $S : \mathrm{List}$

Base Case (nil):

$$\begin{aligned} \mathrm{sum}(\mathrm{echo}(\mathrm{nil})) &= \mathrm{sum}(\mathrm{nil}) && \textbf{def of } \mathrm{echo} \\ &= 0 && \textbf{def of } \mathrm{sum} \\ &= 2 \cdot 0 && \\ &= 2\ \mathrm{sum}(\mathrm{nil}) && \textbf{def of } \mathrm{sum} \end{aligned}$$

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

Need to prove that $\mathrm{sum}(\mathrm{echo}(\mathrm{cons}(x, L))) = 2\ \mathrm{sum}(\mathrm{cons}(x, L))$

Get to assume claim holds for L**, i.e., that** $\mathrm{sum}(\mathrm{echo}(L)) = 2\ \mathrm{sum}(L)$

Example 2: Repeating List Elements

$$\begin{aligned} &\textbf{func}\ \mathrm{echo}(\mathrm{nil}) && := \mathrm{nil} \\ &\quad \mathrm{echo}(\mathrm{cons}(x, L)) && := \mathrm{cons}(x, \mathrm{cons}(x, \mathrm{echo}(L))) \quad \text{for any } x : \mathbb{Z},\ L : \mathrm{List} \end{aligned}$$

- **Prove that** $\mathrm{sum}(\mathrm{echo}(S)) = 2\ \mathrm{sum}(S)$ **for any** $S : \mathrm{List}$

Inductive Hypothesis: assume that $\mathrm{sum}(\mathrm{echo}(L)) = 2\ \mathrm{sum}(L)$

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

$\mathrm{sum}(\mathrm{echo}(\mathrm{cons}(x, L))) =$

$= 2\ \mathrm{sum}(\mathrm{cons}(x, L))$

$$\begin{aligned} &\textbf{func}\ \mathrm{sum}(\mathrm{nil}) && := 0 \\ &\quad \mathrm{sum}(\mathrm{cons}(x, L)) && := x + \mathrm{sum}(L) \quad \text{for any } x \in \mathbb{Z} \text{ and any } L \in \mathrm{List} \end{aligned}$$

Example 2: Repeating List Elements

$$
\begin{aligned}
&\textbf{func}\ \mathrm{echo}(\mathrm{nil}) && := \mathrm{nil} \\
&\qquad \mathrm{echo}(\mathrm{cons}(x, L)) && := \mathrm{cons}(x, \mathrm{cons}(x, \mathrm{echo}(L))) \quad \text{for any } x : \mathbb{Z},\ L : \mathrm{List}
\end{aligned}
$$

- **Prove that** $\mathrm{sum}(\mathrm{echo}(S)) = 2\ \mathrm{sum}(S)$ **for any** $S : \mathrm{List}$

Inductive Hypothesis: **assume that** $\mathrm{sum}(\mathrm{echo}(L)) = 2\ \mathrm{sum}(L)$

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

$$
\begin{aligned}
\mathrm{sum}(\mathrm{echo}(\mathrm{cons}(x, L))) &= \mathrm{sum}(\mathrm{cons}(x, \mathrm{cons}(x, \mathrm{echo}(L)))) && \textbf{def of } \mathrm{echo} \\
&= x + \mathrm{sum}(\mathrm{cons}(x, \mathrm{echo}(L))) && \textbf{def of } \mathrm{sum} \\
&= 2x + \mathrm{sum}(\mathrm{echo}(L)) && \textbf{def of } \mathrm{sum} \\
&= 2x + 2\ \mathrm{sum}(L) && \textbf{Ind. Hyp.} \\
&= 2(x + \mathrm{sum}(L)) && \\
&= 2\ \mathrm{sum}(\mathrm{cons}(x, L)) && \textbf{def of } \mathrm{sum}
\end{aligned}
$$

Proof By Cases

Defining Functions by Cases

- **Usually combine pattern matching with recursion**

- **Can use pattern matching on its own**

$$
\begin{aligned}
&\textbf{func}\ \ \mathrm{empty}(\mathrm{nil}) && := \mathrm{T} \\
&\phantom{\textbf{func}}\ \ \mathrm{empty}(\mathrm{cons}(\mathrm{x}, \mathrm{L})) && := \mathrm{F} \qquad \textbf{for any } \mathrm{x} : \mathbb{Z}, \mathrm{L} : \mathrm{List}
\end{aligned}
$$

  - **every list is either** nil **or** $\mathrm{cons}(\mathrm{x}, \mathrm{L})$ **for some** x **and** L
  - **rule can be applied to any list**

- **Pattern matching is one way to define <u>by cases</u>**
  - **we've seen another way to do this…**

Defining Functions by Cases

- Pattern matching is one way to define <u>by cases</u>
- Side conditions also define by cases
  - e.g., define $\mathrm{f}(\mathrm{m})$ where $\mathrm{m} : \mathbb{Z}$

$$\begin{array}{lll} \textbf{func} & \mathrm{f}(\mathrm{m}) := 2\mathrm{m} + 1 & \textbf{if } \mathrm{m} \geq 0 \\ & \mathrm{f}(\mathrm{m}) := 0 & \textbf{if } \mathrm{m} < 0 \end{array}$$

  - to use the definition on $\mathrm{f}(\mathrm{x})$, need to know if $\mathrm{x} < 0$ or not
- Need ways to reason about these functions as well

Proof By Cases

- New code structure means new proof structure
- Can split a proof into cases
  - e.g., $x \geq 0$ and $x < 0$
  - need to be sure the cases are exhaustive

    (don't need to be exclusive in this case)
- If we can prove both cases, it is true in general

Proof By Cases

func $f(m) := 2m + 1$ **if** $m \geq 0$
$f(m) := 0$ **if** $m < 0$

- **Prove that** $f(m) > m$ **for any** $m : \mathbb{Z}$

Case $m \geq 0$:

$f(m) =$

$> m$

Proof By Cases

func $f(m) := 2m + 1$ **if** $m \geq 0$
$f(m) := 0$ **if** $m < 0$

- **Prove that** $f(m) > m$ **for any** $m : \mathbb{Z}$

Case $m \geq 0$:

$f(m) = 2m + 1$ **def of** f **(since** $m \geq 0$**)**
$\geq m + 1$ **since** $m \geq 0$
$> m$ **since** $1 > 0$

Proof By Cases

func $f(m) := 2m + 1$	**if** $m \geq 0$
$f(m) := 0$	**if** $m < 0$

- **Prove that** $f(m) > m$ **for any** $m : \mathbb{Z}$

Case $m \geq 0$:

$$f(m) = \ldots > m$$

Case $m < 0$:

$f(m) = 0$	**def of** f **(since** $m < 0$**)**
$> m$	**since** $m < 0$

Since these two cases are exhaustive, $f(m) > m$ **holds in general.**

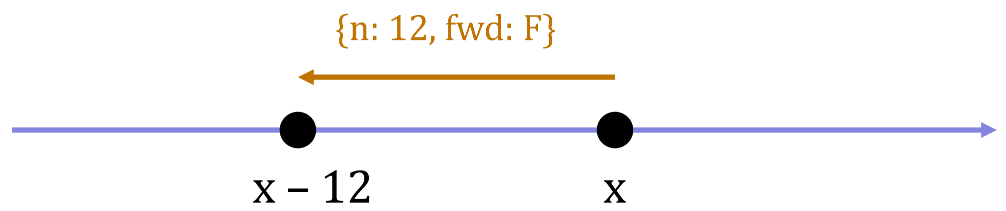
Recall: Pattern Matching

- Define a function by an exhaustive set of patterns

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

$$\begin{aligned} &\textbf{func}\ \text{change}(\{n\text{: } n,\ \text{fwd: T}\}) := n && \text{for any } n : \mathbb{N} \\ &\quad\ \text{change}(\{n\text{: } n,\ \text{fwd: F}\}) := -n && \text{for any } n : \mathbb{N} \end{aligned}$$

  – Steps **describes movement on the number line**
  – change(s : Steps) **says how the position changes**



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

More Proof By Cases

func change({n: n, fwd: T}) := n for any n : $\mathbb{N}$
change({n: n, fwd: F}) := -n for any n : $\mathbb{N}$

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

Case $\text{f} = \text{T}$:

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

More Proof By Cases

func change({n: n, fwd: T}) := n   for any n : $\mathbb{N}$
change({n: n, fwd: F}) := -n   for any n : $\mathbb{N}$

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

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

Case $\text{f} = \text{F}$:

$|\text{change}(\{\text{n: n, fwd: f}\})|$
$= |\text{change}(\{\text{n: n, fwd: F}\})|$ **since** $\text{f} = \text{F}$
$= |\text{-n}|$ **def of** change
$= \text{n}$ **since** $\text{n} \geq 0$

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