CSE 331
Software Design & Implementation

Autumn 2023

Section 4 – Functional Programming II

Administrivia

- HW4 released later today
  - Due Wednesday (10/25) @ 11:00pm

- Remember to check the autograder to make sure you pass your own tests! You won't lose *autograder* points, but they should pass!

- Remember to look at your feedback, you may not have lost points on a question but still have helpful comments

Structural Induction – Review

- Let **P(S)** be the claim
- To Prove P(S) holds for any list S, we need to prove two implications: base case and inductive case
  - **Base Case:** prove P(nil)
    - Use any know facts and definitions
  - **Inductive Hypothesis:** assume **P(L)** is true for a L: List
    - Use this in the inductive step ONLY ⤵️
  - **Inductive Step:** prove **P(cons(x, L))** for any x : *Z*, L : List
    - Direct proof
    - Use know facts and definitions and Inductive Hypothesis
- Assuming we know P(S), if we prove P(cons(x, L)), we then prove recursively that P(S) holds for any List

Defining Function By Cases – Review

- Sometimes we want to define functions by cases
  - **Ex:** define $f(\mathrm{n})$ where $\mathrm{n} : \mathbb{Z}$

$$\begin{aligned} &\textbf{func}\ \ \mathrm{f(n)} &&:= 2\mathrm{n}+1 && \textbf{if } \mathrm{n} \geq 0 \\ &\phantom{\textbf{func}}\ \ \mathrm{f(n)} &&:= 0 && \textbf{if } \mathrm{n} < 0 \end{aligned}$$

  - To use the definition $f(\mathrm{m})$, we need to know if $\mathrm{m} > 0$ or not
  - This new code structure requires a new proof structure

Proof By Cases – Review

- Split a proof into cases:
  - **Ex:** $a = \text{True}$ and $a = \text{False}$ or $n >= 0$ and $n < 0$
  - These cases needs to be *exhaustive*

- **Ex:** $\textbf{func}\ f(n) := 2n + 1$ if $n \geq 0$
  $f(n) := 0$ if $n < 0$
  Prove that $f(n) \geq n$ **for any** $n : \mathbb{Z}$

Case $\mathbf{n \geq 0}$:

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

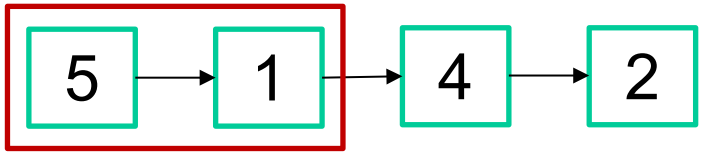
Case $\mathbf{n < 0}$:

$f(n) = 0$ def of f (since $n < 0$)
$\geq n$ since $n < 0$

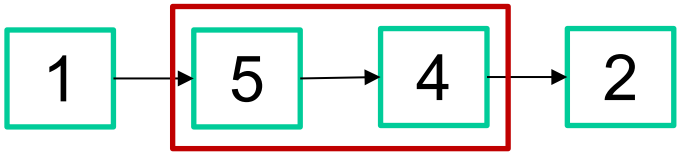
Since these 2 cases are *exhaustive*, $f(n) >= n$ holds in general

Question 1

pseudo-sort: takes a list of numbers as an argument, “looks at the first two numbers in the list, orders the pair to place the smaller of the two in the front, and then continues through the following pairs in the list after the first element“



5 > 1, switch



Now compare 5 & 4 and so on…

(a) Write a formal definition using recursion

(b) Show by example that pseudo-sort does not actually sort the list

Question 2

```
const s = sum(L);
...
return 2 * s;   // = sum(twice(L))
```

Prove this code is correct by showing that **sum(twice(S)) = 2 sum(S)** holds for any list S by structural induction.

$$
\begin{aligned}
&\textbf{func } \mathsf{sum}(\mathsf{nil}) &&:= \ 0 \\
&\quad \mathsf{sum}(\mathsf{cons}(a, L)) &&:= \ a + \mathsf{sum}(L) \quad \text{for any } a : \mathbb{Z} \text{ and } L : \mathsf{List} \\
&\textbf{func } \mathsf{twice}(\mathsf{nil}) &&:= \ \mathsf{nil} \\
&\quad \mathsf{twice}(\mathsf{cons}(a, L)) &&:= \ \mathsf{cons}(2a, \mathsf{twice}(L)) \quad \text{for any } a : \mathbb{Z} \text{ and } L : \mathsf{List}
\end{aligned}
$$

Question 4

```
func swap(nil)                    :=  nil
     swap(cons(a, nil))           :=  cons(a, nil)                    for any a : ℤ
     swap(cons(a, cons(b, L)))    :=  cons(b, cons(a, swap(L)))
                                                  for any a, b : ℤ and L : List
```

Prove by cases that swap(cons(a, L)) $\neq$ nil for any integer a : $\mathbb{Z}$ and list L.

Question 3

sum(twice-evens(L)) + sum(twice-odds(L)) = 3 sum(L)

Prove that this holds for any list S by structural induction.

$$
\begin{array}{llcll}
\textbf{func} & \text{twice-evens}(\text{nil}) & := & \text{nil} & \\
 & \text{twice-evens}(\text{cons}(a, L)) & := & \text{cons}(2a, \text{twice-odds}(L)) & \text{for any } a : \mathbb{Z} \text{ and } L : \text{List} \\
\textbf{func} & \text{twice-odds}(\text{nil}) & := & \text{nil} & \\
 & \text{twice-odds}(\text{cons}(a, L)) & := & \text{cons}(a, \text{twice-evens}(L)) & \text{for any } a : \mathbb{Z} \text{ and } L : \text{List}
\end{array}
$$