CSE 331
Software Design & Implementation

Winter 2026

Section 2 - Testing and Proofs

Administrivia

- HW2 will be released later tonight and is due next **Wednesday @ 11:59pm!**

Proof By Calculation Reminders

- The goal of proof by calculation is to *show* that an assertion is true *given* facts that you already know

- You should ***start*** the proof with either the left or the right side of the assertion and ***end*** the proof with the other side of the assertion.

- Every symbol (=, >, <, etc.) connecting each line of the proof is the current line’s relationship to the previous line in the proof (not any other lines)

- Only modify one side

- **Every** line requires justification (except for algebraic manipulations)

Proof By Calculation - Example

```
// Inputs x and y are positive integers
// Returns a positive integer.
public static int f(int x, int y) {
  return x * y;
};
```

- Known facts “$x \geq 1$” and “$y \geq 1$”
- Correct if the return value is a positive integer

$$
\begin{aligned}
x * y &\geq x * 1 \quad \text{since } y \geq 1 \\
&\geq 1 * 1 \quad \text{since } x \geq 1 \\
&= 1
\end{aligned}
$$

- Calculation shows that $x * y \geq 1$

Proof By Calculation - Citing Functions

sum(nil) := 0

sum(x :: L) := x + sum(L)

- Know "a ≥ 0", "b ≥ 0", and "L = a :: b :: nil"

- Prove the "sum(L)" is non-negative

sum(L)	= sum(a :: b :: nil)	since L = a :: b :: nil
	= a + sum(b :: nil)	def of sum
	= a + b + sum(nil)	def of sum
	= a + b	def of sum
	≥ 0 + b	since a ≥ 0
	≥ 0	since b ≥ 0

Proof By Calculation Bad Example

Suppose we have the facts: $x = 3$, $y = 4$, $z > 5$ and we want to use proof by calculation to prove $x^2 + y^2 < z^2$. Our proof by calculation would look like this:

$x^2 + y^2$	$< z^2$	beginning of a backwards proof
0	$< z^2 - x^2 - y^2$	
0	$< z^2 - 3^2 - y^2$	since $x = 3$
0	$< z^2 - 3^2 - 4^2$	since $y = 4$
0	$< z^2 - 25$	
25	$< z^2$	
5	$< \|z\|$	

Since $z > 5$, we know $x^2+y^2<z^2$ by above.

Manipulates both sides of the equation

Not a single chain of equalities

doesn't end with right side of the assertion (z^2)

What is wrong with this proof?

Proof by Calculation Bug: Explanation

The previous proof is an example of *Circular Reasoning*. We begin the proof with the conclusion manipulating both sides until we reach one of the given facts.

Just because we can prove one direction does **not** mean the other direction necessarily holds.

We must always start from what we know and end with what we want to prove.

Facts → Conclusion ✔

Conclusion → Facts ✖

Proof By Calculation Example Correct

Suppose we have the facts: $x = 3$, $y = 4$, $z > 5$ and we want to use proof by calculation to prove $x^2 + y^2 < z^2$. Our proof by calculation would look like this:

$x^2 + y^2$	$= 3^2 + y^2$	since $x = 3$
	$= 3^2 + 4^2$	since $y = 4$
	$= 25$	
	$= 5^2$	
	$< z^2$	since $z > 5$

note that each line shows the relationship *only* to the previous line

start with left side of assertion

end with right side of assertion

note that every line has justification (except for algebraic manipulations)

Defining Functions By Cases – Review

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

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

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

Proof By Cases – Review

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

- **Ex:** $f(n) := 2n + 1$ if $n \geq 0$
  $f(n) := 0$ if $n < 0$

  Prove that $f(n) \geq n$ **for any** $n : \mathbb{Z}$

Case $n \geq 0$:

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

Case $n < 0$:

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

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

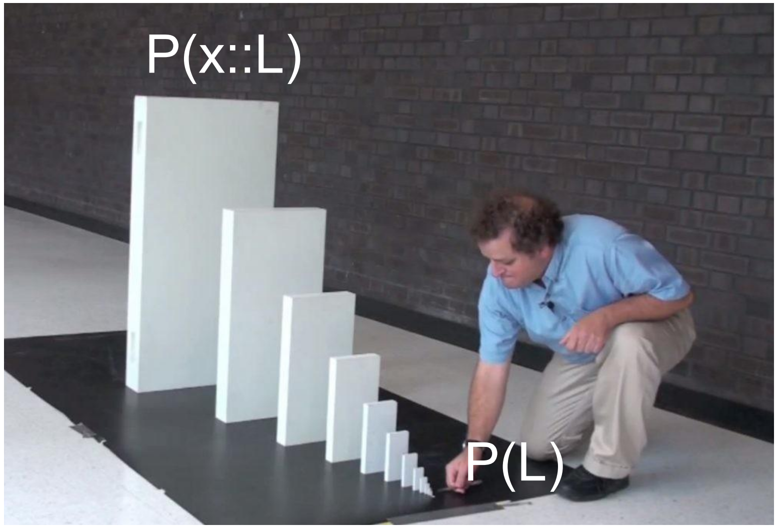
Structural Induction – Review

- 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 known 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(x :: L)** for any x : *Z*, L : List
    - Direct proof
    - Use known facts and definitions and Inductive Hypothesis
- Assuming we know P(L), if we prove P(x :: L), we then prove recursively that P(S) holds for any List

Structural Induction - 331 Format

The following is the structural induction format we recommend for using in your homework (the staff solution also follows this format)

1) **Introduction** - define P(S) to be what we are trying to prove
2) **Base Case** - show P(nil) holds
3) **Inductive Hypothesis** - assume P(L) is true for an arbitrary list
4) **Inductive Step** - show P(x :: L) holds
5) **Conclusion** - “We have shown that P(S) holds for any list”



Review - Testing Heuristics

- **Statement Coverage**
  - Test every executable statement reachable by an allowed input
- **Branch Coverage**
  - For every conditional, test all branches for allowed inputs
- **Loop Coverage**
  - Every loop/recursive call must be tested on 0, 1, any 2+ iterations for allowed inputs
- **Exhaustive Testing**
  - Test all possible inputs for functions with <= 10 allowed inputs

[Notes on Testing Requirements]