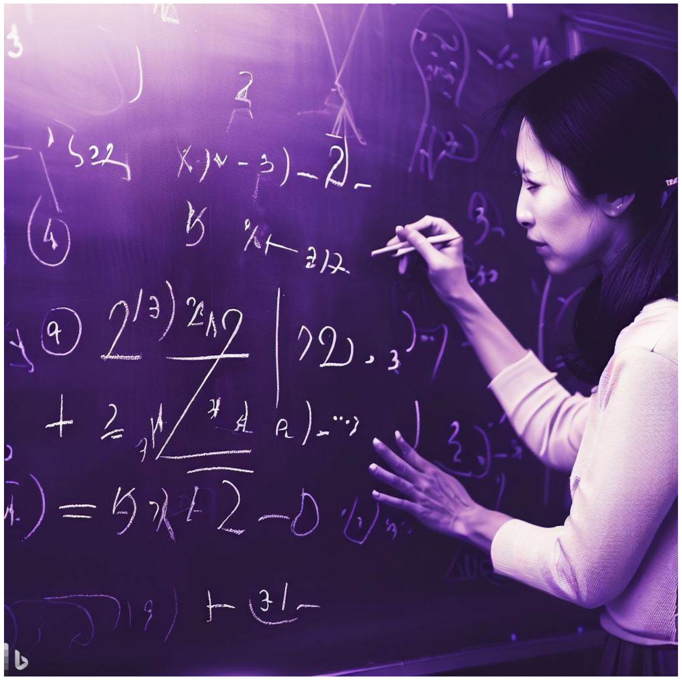

CSE 331

Basics of Reasoning

Kevin Zatloukal

Review

- These three lectures
    1. Data types (data)
    2. Functions (code)
    3. Proofs (reasoning)

- Saw inductive data types
    - most primitive way to build new types

- Structurally recursive functions
    - safest type of recursion
    - only works for recursion on inductive types

Facts

- Basic inputs to reasoning are “facts”
  - things we know to be true about the variables
  - typically, “=” or “<” or “≤”

```
// n must be a natural number
function f(n: number): number {
  const m = 2*n;
  return (m + 1) * (m - 1);
}
```

- At the return statement, we know these facts:
  - $n \in \mathbb{N}$ (or $n \in \mathbb{Z}$ and $n \geq 0$)
  - $m = 2n$

Facts

- Basic inputs to reasoning are “facts”
  - things we know to be true about the variables
  - typically, “=” or “<” or “≤”

```
// n must be a natural number
function f(n: number): number {
  const m = 2*n;
  return (m + 1) * (m - 1);
}
```

- No need to include the fact that n is a number ($n \in \mathbb{R}$)
  - that is true, but the type checker takes care of that
  - no need to repeat reasoning done by the type checker

Implications

- We can use the facts we know to prove more facts
- If we can prove R using facts P and Q,
  we say that R “follows from” / “is implied by” P and Q
  - checking correctness is just proving implications
  - other reasoning tools output implications for us to prove
- The techniques we will learn are
  - proof by calculation
  - proof by cases
  - structural induction

} gives us two implications, each usually proven by calculation

Proof by Calculation

- Proves an implication
  - fact to be shown is an equation or inequality

- Uses known facts and definitions
  - latter includes, e.g., the fact that $\text{len}(\text{nil}) = 0$

Example Proof by Calculation

- Given $x = y$ and $z < 10$, prove that $x + z < y + 10$
  - show the third fact follows from the first two

- Start from the left side of the inequality to be proved

$$x + z$$

Example Proof by Calculation

- Given $x = y$ and $z < 10$, prove that $x + z < y + 10$
  - show the third fact follows from the first two

- Start from the left side of the inequality to be proved

$x + z$	$= y + z$	since $x = y$
	$< y + 10$	since $z < 10$

  - "calculation block", includes explanations in right column

Calculation Blocks

- Chain of “=” shows first = last

$$\begin{aligned} a &= b \\ &= c \\ &= d \end{aligned}$$

  - proves that $a = d$
  - all 4 of these are the same number

Calculation Blocks

- **Chain of "=" and "<" shows <u>first</u> < <u>last</u>**

$x + z$	$= y + z$	**since** $x = y$
	$< y + 10$	**since** $z < 10$
	$= y + 3 + 7$	
	$< w + 7$	**since** $y + 3 < w$

  - **each number is equal or strictly larger that previous**
    last number is strictly larger than the first number
  - **likewise for "=" and "≤"**
    numbers are equal or larger, so last number is largest
  - **analogous for ">" and "≥" cases**

Using Calculation to Prove Correctness

```
// Inputs x and y are positive integers
// Returns a positive integer.
function f(x: number, y, number): number {
  return x + y;
}
```

- Known facts "$x > 0$" and "$y > 0$"

- Correct if the return value is a positive integer

$$x + y$$

Using Calculation to Prove Correctness

```
// Inputs x and y are positive integers
// Returns a positive integer.
function f(x: number, y, number): number {
  return x + y;
}
```

- **Known facts "$x > 0$" and "$y > 0$"**
- **Correct if the return value is a positive integer**

$x + y$	$> x + 0$	since $y > 0$
	$= x$	
	> 0	since $x > 0$

  - calculation shows that $x + y > 0$

Using Calculation to Prove Correctness

```
// Inputs x and y are positive integers
// Returns a positive integer.
function f(x: number, y, number): number {
  return x + y;
}
```

- Known facts "$x \in \mathbb{Z}$" and "$y \in \mathbb{Z}$"

- Correct if the return value is a positive integer
  - we know that "x + y" is an integer
  - should be second nature from Java programming
  - unless there is *division* involved, we will skip this

Using Calculation to Prove Correctness

```
// Inputs x and y are integers with x > 9 and y > -9
// Returns a positive integer.
function f(x: number, y, number): number {
  return x + y;
}
```

- Known facts "$x > 9$" and "$y > -9$"

- Correct if the return value is a positive integer

$$x + y$$

Using Calculation to Prove Correctness

```
// Inputs x and y are integers with x > 9 and y > -9
// Returns a positive integer.
function f(x: number, y, number): number {
  return x + y;
}
```

- Known facts "$x > 9$" and "$y > -9$"

- Correct if the return value is a positive integer

$$
\begin{aligned}
x + y &> x + -9 && \text{since } y > -9 \\
&> 9 - 9 && \text{since } x > 9 \\
&= 0
\end{aligned}
$$

Using Calculation to Prove Correctness

```
// Inputs x and y are integers with x > 3 and y > 4
// Returns an integer that is 10 or larger.
function f(x: number, y, number): number {
  return x + y;
}
```

- **Known facts "$x > 3$" and "$y > 4$"**

- **Correct if the return value is 10 or larger**

$$x + y$$

Using Calculation to Prove Correctness

```
// Inputs x and y are integers with x > 3 and y > 4
// Returns an integer that is 10 or larger.
function f(x: number, y, number): number {
  return x + y;
}
```

- Known facts “$x > 3$” and “$y > 4$”

- Correct if the return value is 10 or larger

$x + y$	$> x + 4$	since $y > 4$
	$> 3 + 4$	since $x > 3$
	$= 7$	

proof doesn’t work because the **code is wrong**!

Using Calculation to Prove Correctness

```
// Inputs x and y are integers with x > 8 and y > -9
// Returns a positive integer.
function f(x: number, y, number): number {
  return x + y;
}
```

- Known facts "$x > 8$" and "$y > -9$"

- Correct if the return value is a positive integer

$$x + y$$

Using Calculation to Prove Correctness

```
// Inputs x and y are integers with x > 8 and y > -9
// Returns a positive integer.
function f(x: number, y, number): number {
  return x + y;
}
```

- **Known facts "$x > 8$" and "$y > -9$"**
- **Correct if the return value is a positive integer**

$x + y$	$> x + -9$	since $y > -9$
	$> 8 - 9$	since $x > 8$
	$= -1$	

proof doesn't work because
the **proof is insufficient**

Using Calculation to Prove Correctness

```
// Inputs x and y are integers with x > 8 and y > -9
// Returns a positive integer.
function f(x: number, y, number): number {
  return x + y;
}
```

- Known facts "$x > 8$" and "$y > -9$"
  - equivalent (since these are integers) to $x \geq 9$ and $y \geq -8$

- Correct if the return value is a positive integer

$$x + y$$

Using Calculation to Prove Correctness

```
// Inputs x and y are integers with x > 8 and y > -9
// Returns a positive integer.
function f(x: number, y, number): number {
  return x + y;
}
```

- Known facts “$x > 8$” and “$y > -9$”
  - equivalent (since these are integers) to $x \geq 9$ and $y \geq -8$

- Correct if the return value is a positive integer

$x + y$	$\geq x + -8$	since $y \geq -8$
	$\geq 9 - 8$	since $x \geq 9$
	$= 1$	
	> 0	

What We Get from Reasoning

- If the proof works, the code is correct
  - why reasoning is useful for finding bugs
- If the code is incorrect, the proof will not work
- If the proof does not work, then either
  1. the code is wrong or
  2. the proof is insufficient (too weak)

  - need to **think** to figure out which
  - (but it's usually because the code is wrong)

Proving Correctness with Conditionals

```
// Inputs x and y are integers.
// Returns a number less than x.
function f(x: number, y, number): number {
  if (y < 0) {
    return x + y;
  } else {
    return x - 1;
  }
}
```

- **Known fact in then branch "$y < 0$"**

$$x + y$$

Proving Correctness with Conditionals

```
// Inputs x and y are integers.
// Returns a number less than x.
function f(x: number, y, number): number {
  if (y < 0) {
    return x + y;
  } else {
    return x - 1;
  }
}
```

- Known fact in then branch "$y < 0$"

$$
\begin{aligned}
x + y \quad & < x + 0 \qquad && \textbf{since } y < 0 \\
& = x
\end{aligned}
$$

Proving Correctness with Conditionals

```
// Inputs x and y are integers.
// Returns a number less than x.
function f(x: number, y, number): number {
  if (y < 0) {
    return x + y;
  } else {
    return x - 1;
  }
}
```

- Known fact in else branch "$y \geq 0$"

$x - 1$

Proving Correctness with Conditionals

```
// Inputs x and y are integers.
// Returns a number less than x.
function f(x: number, y, number): number {
  if (y < 0) {
    return x + y;
  } else {
    return x - 1;
  }
}
```

- Known fact in else branch "$y \geq 0$"

$$\begin{aligned} x - 1 &< x + 0 && \text{since } -1 < 0 \\ &= x \end{aligned}$$

Proving Correctness with Conditionals

```
// Inputs x and y are integers.
// Returns a number less than x.
function f(x: number, y, number): number {
  if (y < 0) {
    return x + y;
  } else {
    return x - 1;
  }
}
```

- Conditionals give us extra known facts
  - get known facts from
    1. specification
    2. conditionals
    3. constant declarations

Using Definitions in Calculations

- **Most useful with function calls**
  - cite the definition of the function to get the return value

- **For example**

func sum(nil) $:= 0$

sum(cons(x, L)) $:= x + \text{sum}(L)$ for any $x \in \mathbb{Z}$ and any $L \in \text{List}$

- **Can cite facts such as**
  - $\text{sum}(\text{nil}) = 0$
  - $\text{sum}(\text{cons}(a, \text{cons}(b, \text{nil}))) = a + \text{sum}(\text{cons}(b, \text{nil}))$

second case of definition with $x = a$ and $L = \text{cons}(b, \text{nil})$

Using Definitions in Calculations

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

- **Consider this code**

```
// Inputs a and b must be integers.
// Returns a non-negative integer.
function f(a: number, b: number): number {
  const L: List = cons(a, cons(b, nil));
  if (a >= 0 && b >= 0)
    return sum(L);
  ...
```

- **Known facts include "$a \geq 0$", "$b \geq 0$", and "$L = \text{cons}(\ldots)$"**

Using Definitions in Calculations

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

- **Know** "$a \geq 0$", "$b \geq 0$", **and** "$L = \text{cons}(a, \text{cons}(b, \text{nil}))$"

- **Prove the return value is non-negative**

sum(L)

Using Definitions in Calculations

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

- **Know** "$a \geq 0$", "$b \geq 0$", **and** "$L = \text{cons}(a, \text{cons}(b, \text{nil}))$"

- **Prove the return value is non-negative**

$\text{sum}(L)$	$= \text{sum}(\text{cons}(a, \text{cons}(b, \text{nil})))$	**since** $L = \text{cons}(a, \text{cons}(b, \text{nil}))$
	$= a + \text{sum}(\text{cons}(b, \text{nil}))$	**def of** sum
	$= a + b + \text{sum}(\text{nil})$	**def of** sum
	$= a + b$	**def of** sum
	$\geq 0 + b$	**since** $a \geq 0$
	≥ 0	**since** $b \geq 0$