# Mathematical Logic for the Greater Good

Harley Eades III
School of Computer and Cyber Sciences
Augusta University

# for the Greater Good

Harley Eades III
School of Computer and Cyber Sciences
Augusta University

# Mathematical Logic for the Greater Good

Harley Eades III
School of Computer and Cyber Sciences
Augusta University

# About Me

- BS in Math and CS from Millikin University
- MS in Theoretical CS from University of Iowa
- PhD in Theoretical CS from University of Iowa
- Assistant Professor in CS at Augusta University

# About Me


VEGAS
RED & BRED @ HUMA
HOUSE

# Software Verification

# Example: Quicksort

[9,3,4,2,7]

Pivot

# Example: Quicksort

[3,2,4,9,7]

Pivot

# Example: Quicksort

[3,2] [4] [9,7]

Pivot

Pivot

# Example: Quicksort

[2,3] [4] [7,9]

# Example: Quicksort

[2,3,4,7,9]

# Example: Quicksort

```
algorithm quicksort(A:List Int, lo:Int, hi:Int) {
   p := partition(A, lo, hi)
   quicksort(A, lo, p)
   quicksort(A, p + 1, hi)
}
```

```
algorithm partition(A:List Int, lo:Int, hi:Int) {
    mid = (lo + hi) / 2
   pivot := A[mid]
   …
}
```

# Example: Quicksort

Problems:

- What if lo or hi is negative?
- What if hi < lo?
- What if the size of the list is 0?

# Verification in the Real World

- Average iPhone app - 10K LOC
- F-35 Fighter Jet - 25M LOC
- Windows - 40M LOC
- Facebook - 61M LOC
- Mac OS X - 80M LOC
- Car Software - 100M LOC

# How do we take the human guess work out of the picture?

# Language-Based Formal Methods

Programming Languages

+

Mathematical Logic

=

Software Verification

# Example: Quicksort

Problems:

- What if lo or hi is negative?
- What if hi < lo?
- What if the size of the list is 0?

```
algorithm quicksort(A:List Int, lo:Int, hi:Int) {
   p := partition(A, lo, hi)
   quicksort(A, lo, p)
   quicksort(A, p + 1, hi)
}
```

```
algorithm partition(A:List Int, lo:Int, hi:Int) {
    mid = (lo + hi) / 2
    pivot := A[mid]
    …
}
```

# Example: Quicksort

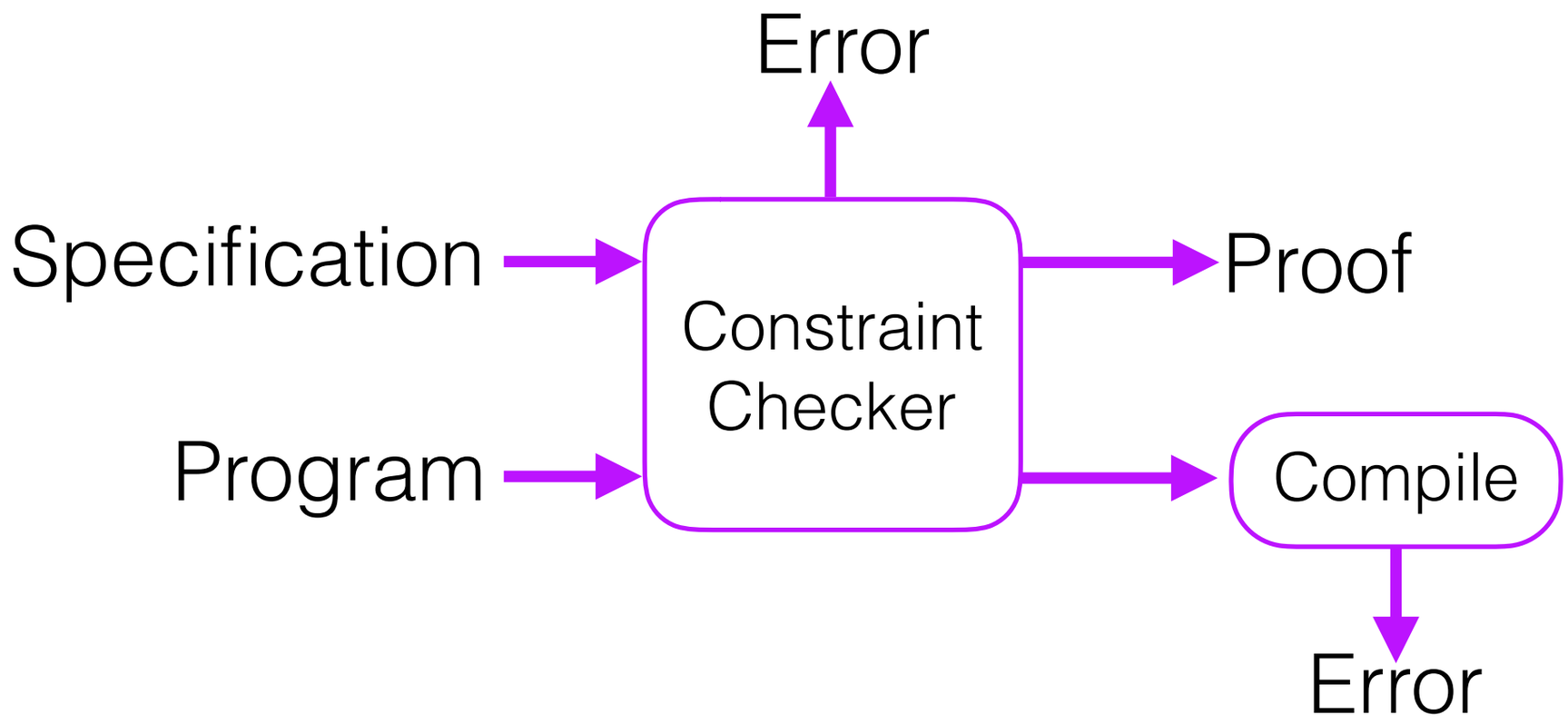
Specification

Program

```
{- lo >= 0 && hi >= 0 -}
{- lo < hi -}
{- length(A) > 0 -}
algorithm quicksort(A:List Int, lo:Int, hi:Int) {
   p := partition(A, lo, hi)
   quicksort(A, lo, p)
   quicksort(A, p + 1, hi)
}
```

```
algorithm partition(A:List Int, lo:Int, hi:Int) {
    mid = (lo + hi) / 2
    pivot := A[mid]
    …
}
```

# Compilation and Verification



# Example: Quicksort

```
{- lo >= 0 && hi >= 0 -}
{- lo < hi -}
{- length(A) > 0 -}
algorithm quicksort(A:List Int, lo:Int, hi:Int) {
   p := partition(A, lo, hi)
   quicksort(A, lo, p)
   quicksort(A, p + 1, hi)
}
```

```
algorithm partition(A:List Int, lo:Int, hi:Int) {
    mid = (lo + hi) / 2
    pivot := A[mid]
    …
}
```

# Example: Quicksort

```
{- lo >= 0 && hi >= 0 -}
{- lo < hi -}
{- length(A) > 0 -}
algorithm quicksort(A:List Int, lo:Int, hi:Int) {
   if (lo >= 0 && hi >= 0 && lo < hi && length(A) > 0) {
     p := partition(A, lo, hi)
     quicksort(A, lo, p)
     quicksort(A, p + 1, hi)
   } else {
      return ERROR
   }
}
```

```
algorithm partition(A:List Int, lo:Int, hi:Int) {
   mid = (lo + hi) / 2
   pivot := A[mid]
   …
}
```

# Example: Quicksort

How do we know the output is sorted?

# Example: Quicksort

$$isSorted([]) = \mathsf{True}$$
$$isSorted([x]) = \mathsf{True}$$
$$isSorted([x_1, x_2, \ldots, x_n]) = (x_1 \leq x_2) \wedge \cdots \wedge (x_{n-1} \leq x_n)$$

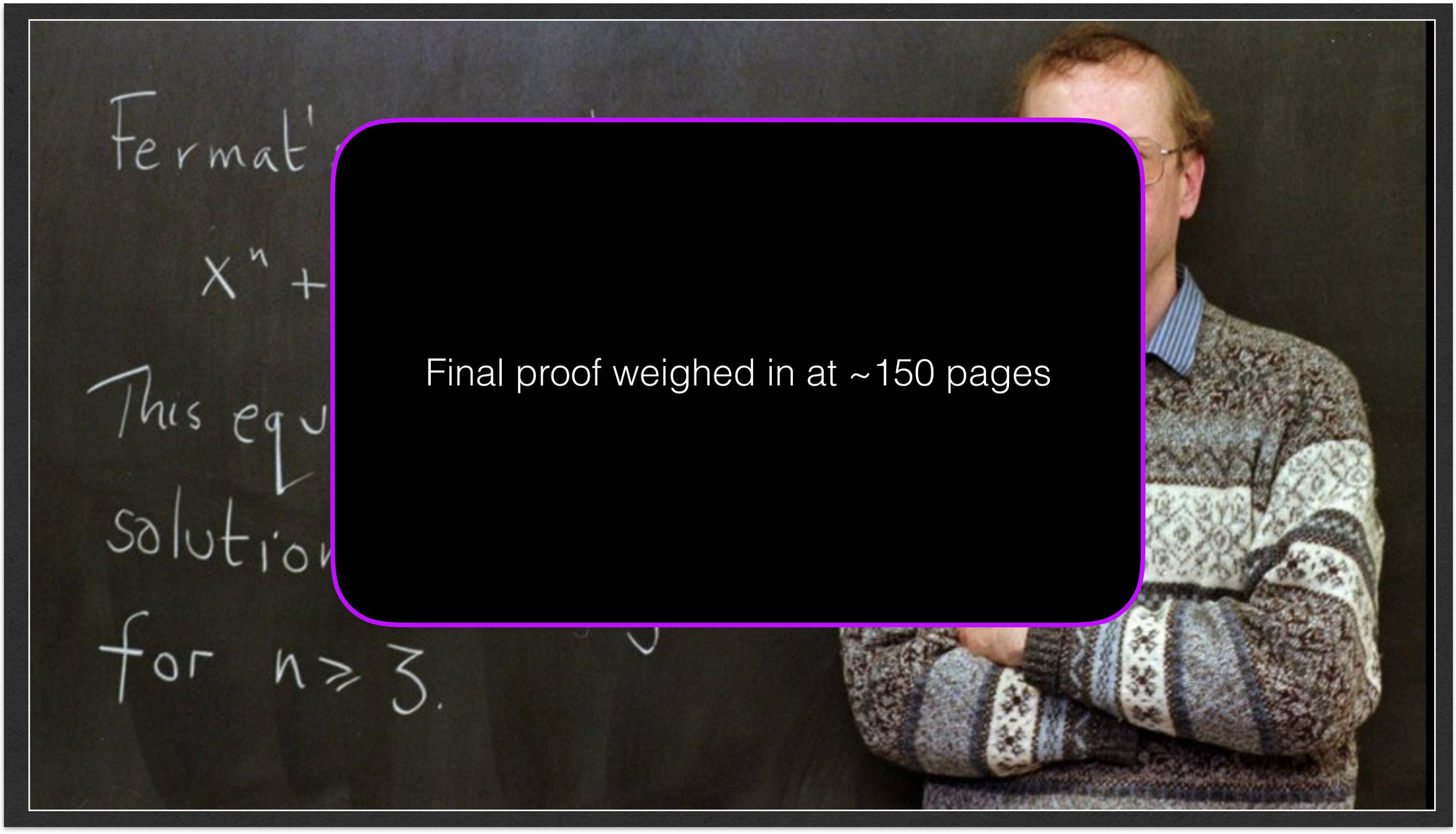
# Example: Quicksort

```
{- lo >= 0 && hi >= 0 -}
{- lo < hi -}
{- length(A) > 0 -}
algorithm quicksort(A:List Int, lo:Int, hi:Int) {
   if (lo >= 0 && hi >= 0 && lo < hi && length(A) > 0) {
     p := partition(A, lo, hi)
     quicksort(A, lo, p)
     quicksort(A, p + 1, hi)
   } else {
      return ERROR
   }
}
{- isSorted(A) -}
```

```
algorithm partition(A:List Int, lo:Int, hi:Int) {
   mid = (lo + hi) / 2
   pivot := A[mid]
   …
}
```

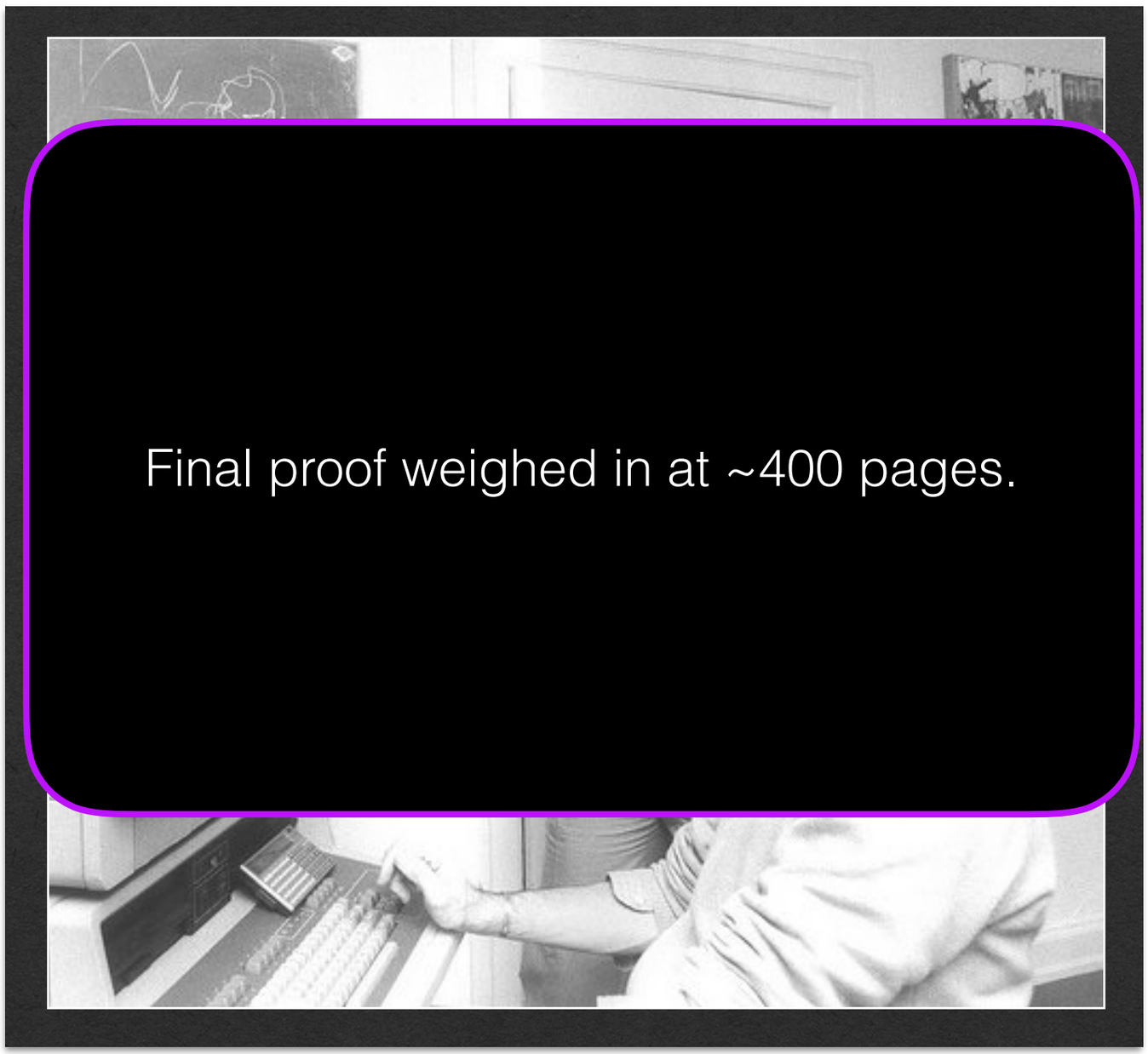
# Machine-Checked Mathematics

# Proof are hard!



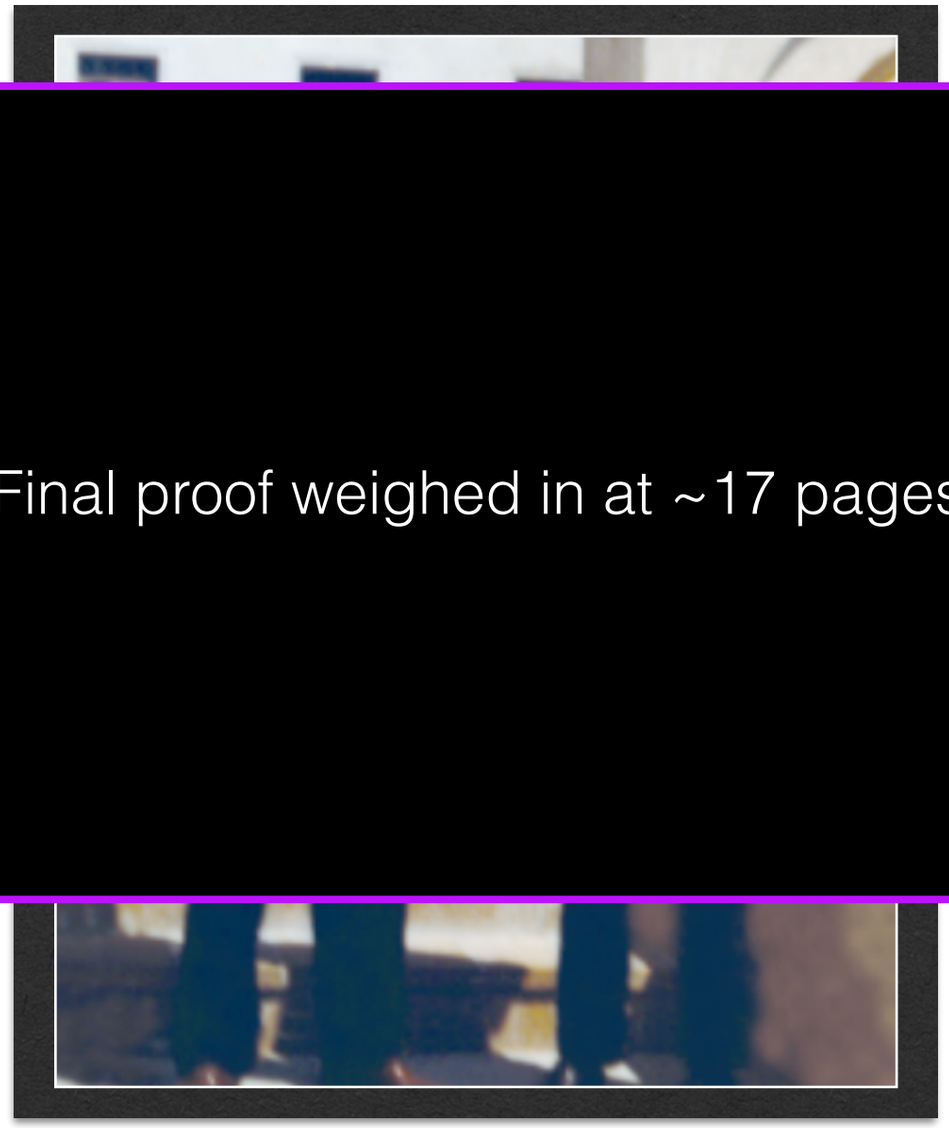
# Proof are hard!

Four Color Theorem



# Proof are hard!

Odd Order Theorem:



Final proof weighed in at ~17 pages.

# Interactive Theorem Provers

Allow the mathematician to write their proofs on the computer.

# Interactive Theorem Provers

The computer checks their proofs for errors.

# Interactive Theorem Provers

Gives the mathematician a higher confidence that their proof is correct.

# Interactive Theorem Provers

Based on the correspondence

between

proofs and programs.

# Example: Peano Arithmetic

| | |
|---|---|
| $0$ | $0$ |
| $1$ | $S(0)$ |
| $2$ | $S(S(0))$ |
| $3$ | $S(S(S(0)))$ |
| $\vdots$ | $\vdots$ |
| $n$ | $S^n(0)$ |

# Example: Peano Arithmetic

$$S^{p_1}(0) + S^{p_2}(0)$$

$$= S^{p_1}(S^{p_2}(0))$$

$$= S^{p_1+p_2}(0)$$

# Example: Peano Arithmetic

$$
\begin{aligned}
& S^{p_2}(0) + S^{p_1}(0) \\
= \; & S^{p_2}(S^{p_1}(0)) \\
= \; & S^{p_2+p_1}(0)
\end{aligned}
$$

# Example: Peano Arithmetic in Agda

```
_+_ : ℕ → ℕ → ℕ
0  + n = n
S(m) + n = S (m + n)
```

# Example: Peano Arithmetic in Agda

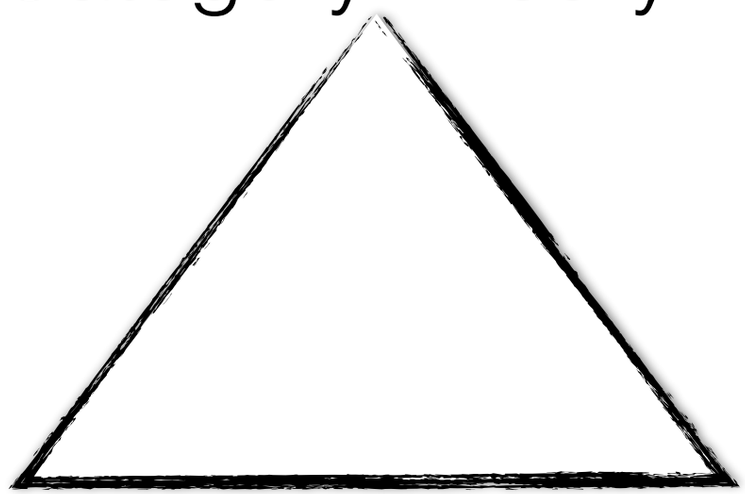
```
+comm : ∀ (x y : ℕ) → x + y ≡ y + x
+comm Z y rewrite +0 y = refl
+comm (S x) y rewrite +suc y x | +comm x y = refl
```

```
+suc : ∀ (x y : ℕ) → x + (S y) ≡ S(x + y)
+suc Z y = refl
+suc (S x) y rewrite +suc x y = refl
```

```
+0 : ∀ (x : ℕ) → x + Z ≡ x
+0 Z = refl
+0 (S x) rewrite +0 x = refl
```

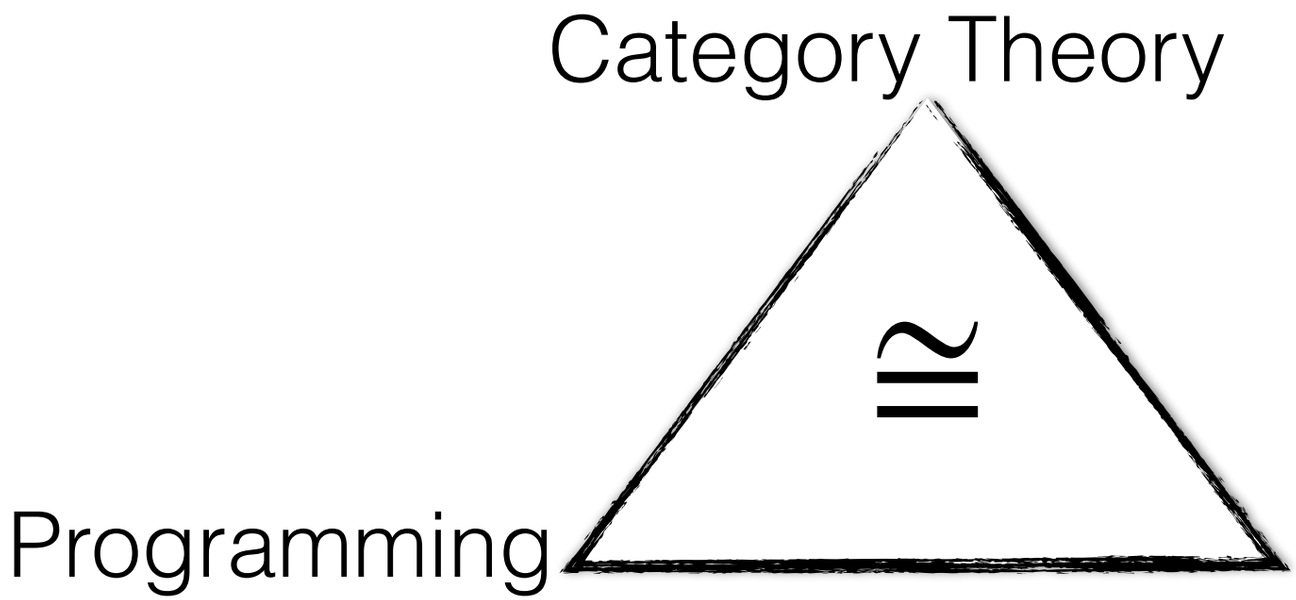
# The Three Perspectives of Computation

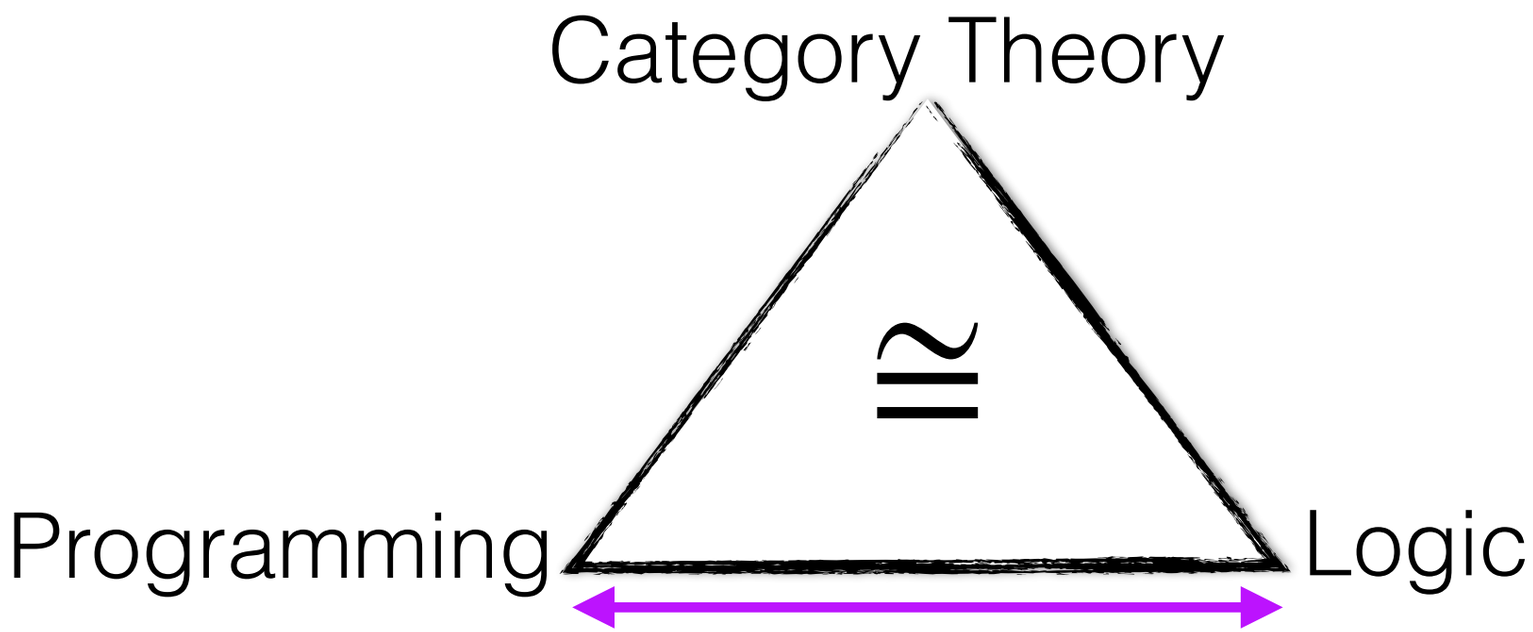
Category Theory



Programming

Logic


Category Theory
≅
Programming
Logic


Category Theory
≅
Programming
Logic

# Where do I fit in?

I work on the design and analysis of the next generation of programming languages that support software verification.

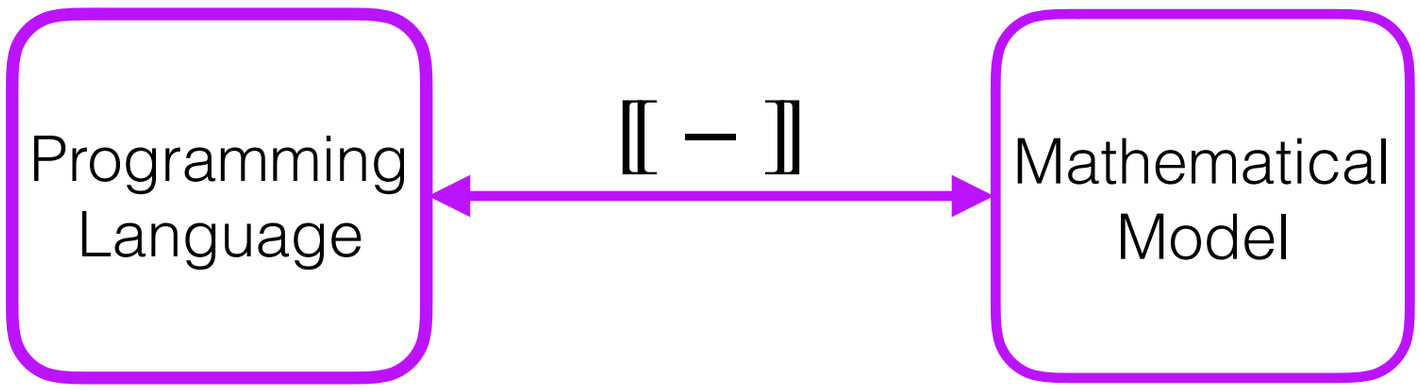
Make it easier to verify real-world software that use external resources like memory, secure information, network protocols, etc.

The new languages I am working on move well beyond the state of the art like Agda.

# Language Design

- Design the syntax of the language.
- Add new features such as:
  - Resource sensitivity

# Language Analysis

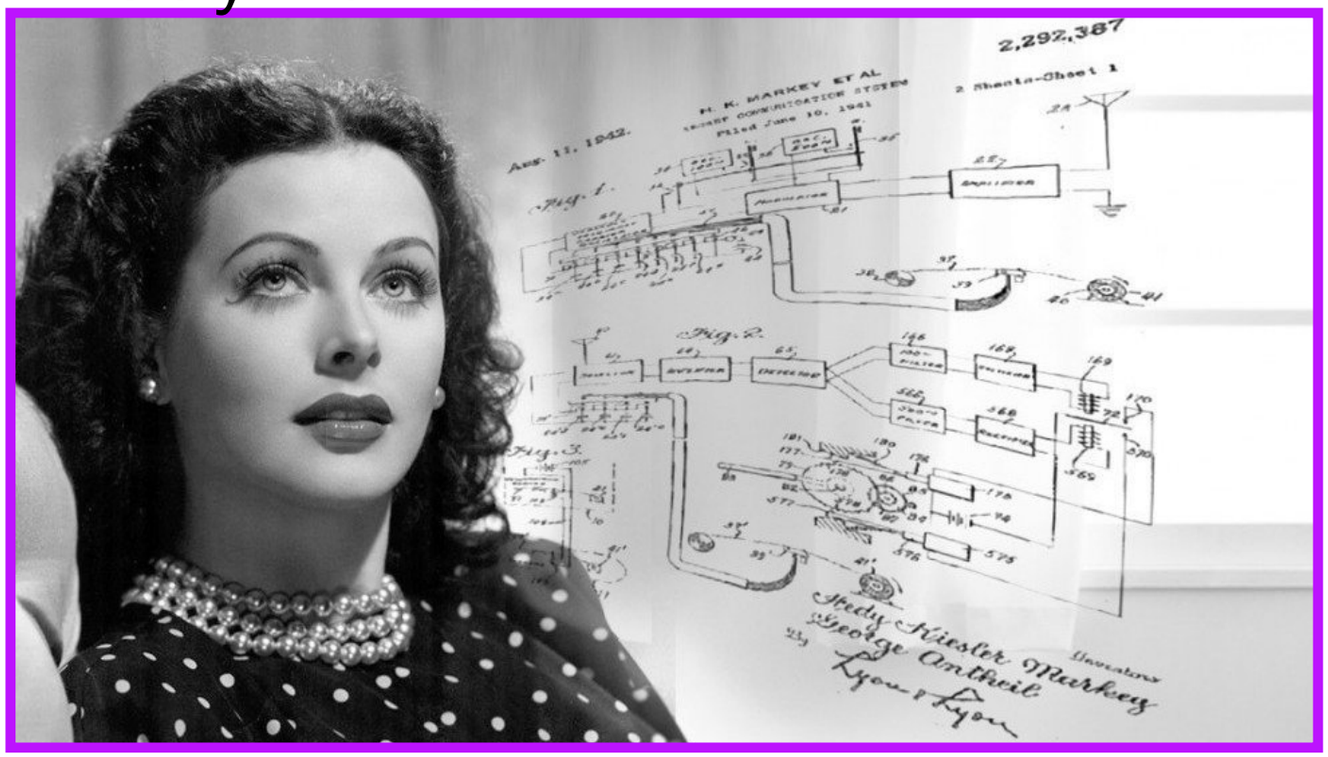


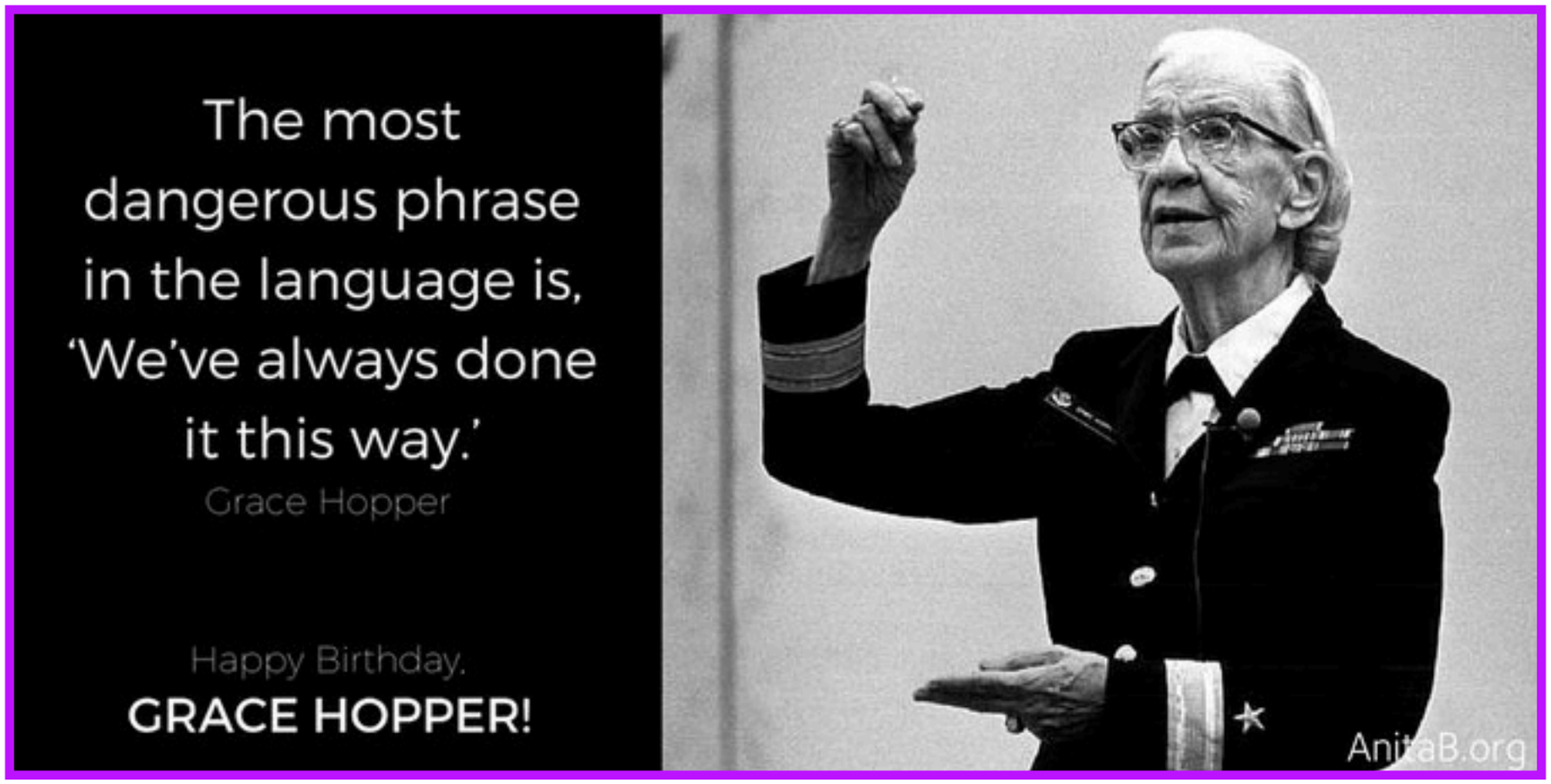
## Valeria de Paiva

## Stephanie Weirich

# Emmy Noether

# Hedy Lamarr




The most dangerous phrase in the language is, 'We've always done it this way.'
Grace Hopper
Happy Birthday.
GRACE HOPPER!
AnitaB.org

# Questions?