Lean for Scientists and Engineers

Tyler R. Josephson

AI & Theory-Oriented Molecular Science (ATOMS) Lab

University of Maryland, Baltimore County

Boba Beach
Grynpyret

Twitter: @trjosephson
Email: tjo@umbc.edu

Lean for Scientists and Engineers 2024

1. Logic and proofs for scientists and engineers
   1. Introduction to theorem proving
   2. Writing proofs in Lean
   3. Formalizing derivations in science and engineering
2. Functional programming in Lean 4
   1. Functional vs. imperative programming
   2. Numerical vs. symbolic mathematics
   3. Writing executable programs in Lean
3. Provably-correct programs for scientific computing

Schedule (tentative)

Logic and proofs for scientists and engineers
Functional programming in Lean 4
Provably-correct programs for scientific computing

July 9, 2024	Introduction to Lean and proofs
July 10, 2024	Equalities and inequalities
July 16, 2024	Proofs with structure
July 17, 2024	Proofs with structure II
July 23, 2024	Proofs about functions; types
July 24, 2024	Calculus-based-proofs
July 30-31, 2024	Prof. Josephson traveling
August 6, 2024	Functions, definitions, structures, recursion
August 8, 2024	Polymorphic functions for floats and reals, compiling Lean to C
August 13, 2024	Input / output, lists, arrays, and indexing
August 14, 2024	Lists, arrays, indexing, and matrices
August 20, 2024	LeanMD & BET Analysis in Lean
August 21, 2024	SciLean tutorial, by Tomáš Skřivan

Content inspired by:
Mechanics of Proof, by Heather Macbeth
Functional Programming in Lean, by David Christiansen

Guest instructor: Tomáš Skřivan

Schedule for today

1. Recap Lecture 4
2. Types
3. Functions
    1. Math vs. programming
    2. Examples of functions
    3. Computable vs. noncomputable
4. “Proportional to”

Slide from Lecture 3

How to find tactics

- Keep learning them one by one!
- Indexes for Mechanics of Proof, Mathematics in Lean
- Consult lists of useful tactics
  - https://github.com/madvorak/lean4-tactics
  - https://github.com/Colin166/Lean4/blob/main/UsefulTactics
- If you have a tactic in hand, mouseover in VS Code to see documentation and example(s)

Slide from Lecture 3

How to find theorems

- Keep practicing!
- Search Mathlib documentation
  - https://leanprover-community.github.io/mathlib4_docs/
  - Using the search bar, make a guess about what the theorem would be named, and start checking things that look promising
- Moogle
  - https://www.moogle.ai
  - Describe theorem (or definition) in natural language, the scroll through options
- Consult lists of useful theorems
  - https://github.com/Colin166/Lean4/blob/main/UsefulLemmas.lean
- If you have a theorem in hand, mouseover in VS Code to see documentation and example(s)

Slide from Lecture 4

Definitions

- Allow us to reuse terms outside of individual examples / theorems
- Facilitates modular code and verification of different parts of code
- Propositions
- Functions

Slide from Lecture 3

Glossary of logical symbols

∧ - and

∨ - or

¬ - not

→ - implies

↔ - if and only if (implies in both directions)

∃ - exists

∀ - for all

Slide from Lecture 4

Combining ∀ and ∃ to represent "proportional to"

- Empirical science often describes phenomena using proportions
- Kepler's Third Law – orbital period T vs. semi-major axis d

$$T^2 \propto d^3 \qquad T^2 = \left(\frac{4\pi^2}{G(m+M)}\right)d^3 \qquad T^2 = \left(\frac{4\pi^2}{GM}\right)d^3$$

Slide from Lecture 4

Combining ∀ and ∃ to represent "proportional to"

- Empirical science often describes phenomena using proportions
- Boyle's Law
  - Pressure is inversely proportional to volume

$$P \propto \frac{1}{V} \qquad P = \frac{k}{V} \qquad PV = k \qquad P_1V_1 = P_2V_2$$

Slide from Lecture 4

Combining ∀ and ∃ to represent "proportional to"

- Empirical science often describes phenomena using proportions
- Boyle's Law
  - Pressure is inversely proportional to volume

$$P \propto \frac{1}{V} \qquad P = \frac{k}{V} \qquad PV = k \qquad P_1V_1 = P_2V_2$$

- *There exists* some constant k, such that *for all* thermodynamic states, this relationship between pressure and volume holds

Slide from Lecture 4

Combining ∀ and ∃ to represent "proportional to"

- Empirical science often describes phenomena using proportions
- Boyle's Law
  - Pressure is inversely proportional to volume

$$P \propto \frac{1}{V} \qquad P = \frac{k}{V} \qquad PV = k \qquad P_1V_1 = P_2V_2$$

- *There exists* some constant k, such that *for all* thermodynamic states, this relationship between pressure and volume holds
- ∃ (k : ℝ), ∀ (n : ℕ), (P n)*(V n) = k
- Next time!

Slide from Lecture 1

A guide to number systems

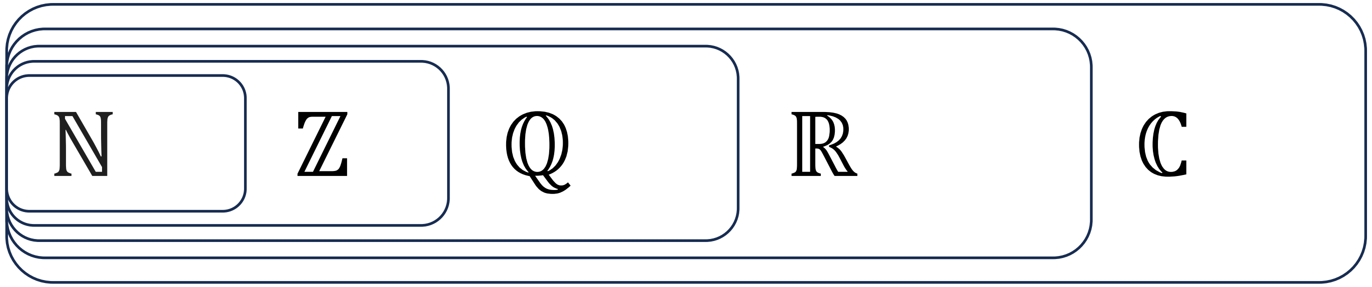
ℕ - Natural numbers (0, 1, 2, 3, 4, …)

ℤ - Integers (… -3, -2, -1, 0, 1, 2, …)

ℚ - Rational numbers (1/2, 3/4, 5/9, etc.)

ℝ - Real numbers (-1, 3.6, π, √2)

ℂ - Complex numbers (-1, 5 + 2i, √2 + 5i, etc.)



Types

- Lean has an unusually expressive type system (Dependent Type Theory) that underlies its ability to check math proofs
- You can think of types as sets
  - Terms are members of a type
- The various number systems ($\mathbb{N}$, $\mathbb{Z}$, $\mathbb{Q}$, $\mathbb{R}$, $\mathbb{C}$) are types
  - Most of these are *constructed* from other operations

Types

- Lean has an unusually expressive type system (Dependent Type Theory) that underlies its ability to check math proofs
- You can think of types as sets
  - Terms are members of a type
- The various number systems ($\mathbb{N}$, $\mathbb{Z}$, $\mathbb{Q}$, $\mathbb{R}$, $\mathbb{C}$) are types
  - Most of these are *constructed* from other operations
- Lean also has familiar programming types
  - Float, String, List, Array, Bool, etc.

Types

- Lean has an unusually expressive type system (Dependent Type Theory) that underlies its ability to check math proofs
- You can think of types as sets
  - Terms are members of a type
- The various number systems ($\mathbb{N}$, $\mathbb{Z}$, $\mathbb{Q}$, $\mathbb{R}$, $\mathbb{C}$) are types
  - Most of these are *constructed* from other operations
- Lean also has familiar programming types
  - Float, String, List, Array, Bool, etc.
- Formulas and theorems have type Prop (for proposition)
- Functions have types, too
  - A function that doubles a natural number has type $\mathbb{N} \rightarrow \mathbb{N}$

Exercises with Types

- Theorem Proving in Lean 4, Chapter 2
  - https://leanprover.github.io/theorem_proving_in_lean4/dependent_type_theory.html
- What's the type of "Type"?
  - Type 1
  - What's the type of "Type 1"?
    - Type 2
    - What's the type of "Type 2"?
      - Type 3, etc.
- Lean has a hierarchy of Type universes
  - See above for more details, we don't usually need to deal with this

Corinne's Shibboleth: Functions in Physics vs Math

From blog: https://physicsteacher.blog/2018/02/16/corinnes-shibboleth/

Dray & Manogue, 2002, Vector calculus bridge project website
http://www.math.oregonstate.edu/bridge/ideas/functions

Suppose the temperature on a rectangular slab of metal is given by $T(x, y) = k(x^2 + y^2)$ where k is a constant.

What is $T(r, \theta)$?

Did you answer:

- **A:** $T(r, \theta) = kr^2$
- **B:** $T(r, \theta) = k(r^2 + \theta^2)$
- **C:** Neither

Corinne's Shibboleth: Functions in Physics vs Math

From blog: https://physicsteacher.blog/2018/02/16/corinnes-shibboleth/

Dray & Manogue, 2002, Vector calculus bridge project website
http://www.math.oregonstate.edu/bridge/ideas/functions

Suppose the temperature on a rectangular slab of metal is given by $T(x, y) = k(x^2 + y^2)$ where k is a constant.

What is $T(r, \theta)$?

Did you answer:

- **A:** $T(r, \theta) = kr^2$
- **B:** $T(r, \theta) = k(r^2 + \theta^2)$
- **C:** Neither

Mathematicians usually choose A
Most scientists and engineers choose B

Functions: Programming vs. Math

Programming perspective

A function takes arguments, performs calculations, and produces an output

Examples in Python

```
def squared(x):
    y = x*x
    return y
```

Functions: Programming vs. Math

Programming perspective

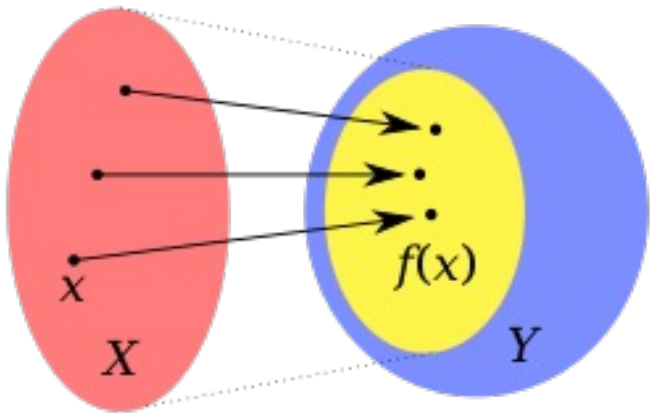
A function takes arguments, performs calculations, and produces an output

Examples in Python

```
def squared(x):
    y = x*x
    return y
```

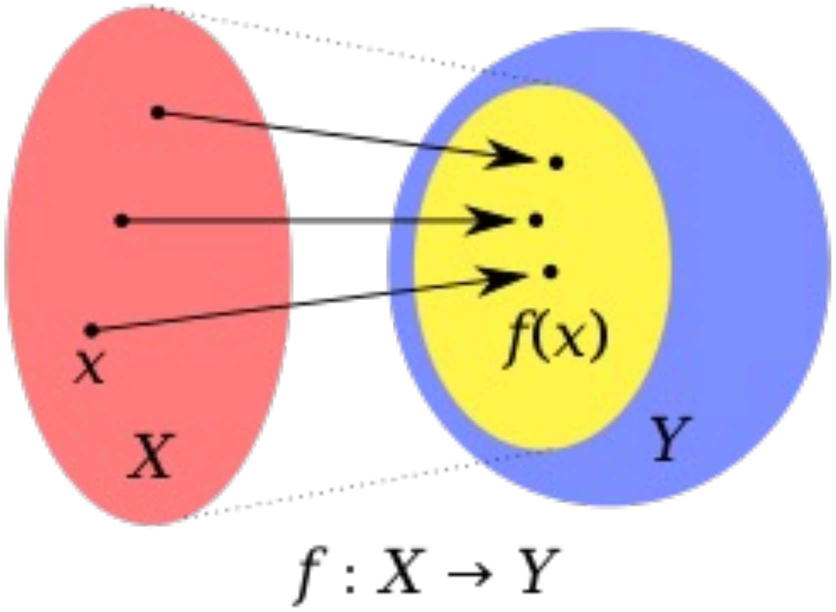
Math perspective

A function maps values from a domain to a co-domain



$f : X \to Y$

Functions: Programming vs. Math




```
def squared(x):
    y = x*x
    return y
```

$$f(x) = x^2$$

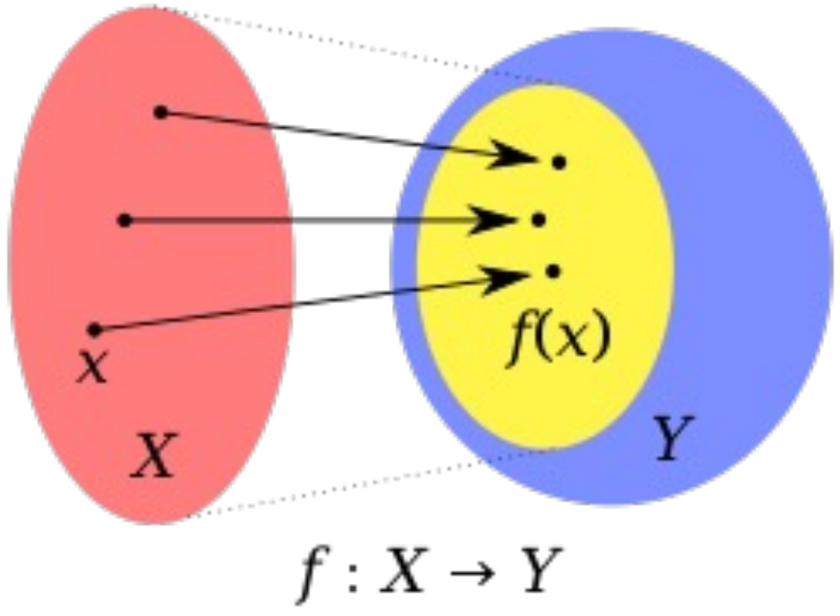
A function!
With type $\mathbb{Z} \rightarrow \mathbb{Z}$
Or $\mathbb{R} \rightarrow \mathbb{R}$

Domain

Co-domain

Image

Functions: Programming vs. Math




```
def squareroot(x):
    y = x**(1/2)
    return y
```

$$f(x) = \sqrt{x}$$

Not always a function!
With type $\mathbb{Z} \rightarrow \mathbb{Z}$ or $\mathbb{R} \rightarrow \mathbb{R}$, there is no mapping from the x < 0 part of the domain

With type $\mathbb{N} \rightarrow \mathbb{R}$ or $\mathbb{R} \rightarrow \mathbb{C}$, it *is* a function; every part of the domain maps to a value in the co-domain

Domain
Co-domain
Image

Corinne's Shibboleth: Functions in Physics vs Math

From blog: https://physicsteacher.blog/2018/02/16/corinnes-shibboleth/

Dray & Manogue, 2002, Vector calculus bridge project website
http://www.math.oregonstate.edu/bridge/ideas/functions

Suppose the temperature on a rectangular slab of metal is given by $T(x, y) = k(x^2 + y^2)$ where k is a constant.

What is $T(r, \theta)$?

Did you answer:

- **A:** $T(r, \theta) = kr^2$
- **B:** $T(r, \theta) = k(r^2 + \theta^2)$
- **C:** Neither

Mathematicians usually choose A
Most scientists and engineers choose B

Slide from Lecture 4

→ : implies

P → Q means "if P, then Q"

P: detecting argon

Q: detecting a noble gas

P → Q: detecting argon implies detecting a noble gas

P: true, Q: true – then P → Q: true

P: false, Q: true – then P → Q: true

P: true, Q: false – then P → Q: false

P: false, Q: false – then P → Q: true

P	Q	(P → Q)
true	true	true
false	true	true
true	false	false
false	false	true

Glossary

- Equation
  - Proposition about equality statement
- Formula
  - Proposition about expressions, includes equalities, inequalities, as well as logial operators
- Expression
  - Like the "right hand side" of an equation
  - Type depends on the types and operations of things inside
- Function (aka pure function)
  - An expression that maps from domain to co-domain
- Partial function
  - An expression that maps from *part* of domain to co-domain

Examples of functions

What's a good type?

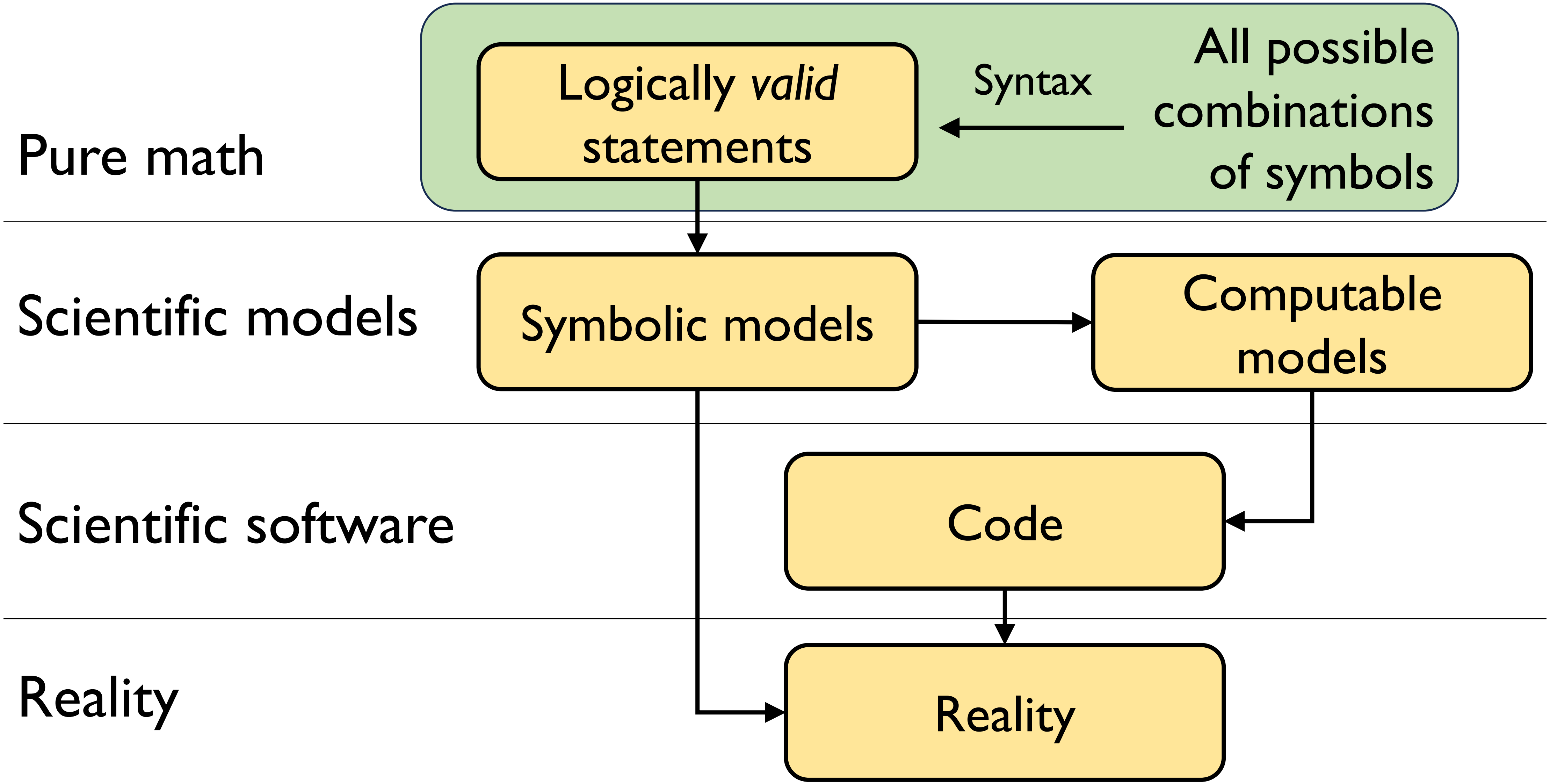
$I(t)$	Electric current as a function of time
$T(x, y)$	Temperature as a function of position, Cartesian coordinates
$T(r, \theta)$	Temperature as a function of position, polar coordinates
P_n	Pressure as a function of thermodynamic state
$\delta(x)$	Detector threshold as a function of measurement

Functions in Lean

- Further discussion in Lecture 7
- No parentheses needed – just a space will do
  - f(x) is written as f x
- We can *prove* things about pure functions; it's much harder with partial functions
- Lean requires you to label "noncomputable" functions
  - Noncomputable means "incapable of being computed by any algorithm in a finite amount of time"
  - Real.pi is noncomputable

Slide from Lecture 3

Syntax and semantics in scientific computing



Slide from Lecture 4

Combining ∀ and ∃ to represent "proportional to"

- Empirical science often describes phenomena using proportions
- Boyle's Law
  - Pressure is inversely proportional to volume

$$P \propto \frac{1}{V} \qquad P = \frac{k}{V} \qquad PV = k \qquad P_1V_1 = P_2V_2$$

- *There exists* some constant k, such that *for all* thermodynamic states, this relationship between pressure and volume holds
- ∃ (k : ℝ), ∀ (n : ℕ), (P n)*(V n) = k
- Next time!

Building a network of proofs