UMBC

Chemical Biochemical and Environmental Engineering

LEAN THEOREM PROVER

Lean for Scientists and Engineers

Tyler R. Josephson

AI & Theory-Oriented Molecular Science (ATOMS) Lab

University of Maryland, Baltimore County

Cloudspin
BB Yukus & Electric Dad

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. Provably-correct scientific computing
2. Derivations in science and engineering are math proofs
3. Formalizing mathematics with computers
4. Lean 4 and Mathlib
5. Case studies in proofs: adsorption and gas law thermodynamics
6. Case study in programming: bug-free BET analysis
7. Outlook
    1. LeanMD
    2. LLMs for theorem proving
    3. SciLib

Intermission

1. Getting connected with this course
2. Getting started with Lean
3. Proofs about equality

Schedule for today

1. Provably-correct scientific computing
2. Derivations in science and engineering are math proofs
3. Formalizing mathematics with computers
4. Lean 4 and Mathlib
5. Case studies in proofs: adsorption and gas law thermodynamics
6. Case study in programming: bug-free BET analysis
7. Outlook
    1. LeanMD
    2. LLMs for theorem proving
    3. SciLib

Intermission

1. Getting connected with this course
2. Getting started with Lean
3. Proofs about equality

Impact of non-pharmaceutical interventions (NPIs) to reduce COVID-19 mortality and healthcare demand

Ferguson, N.M., et al. Imperial College London COVID-19 Response Team. March 16, 2020

"SimCity without the graphics"

The Telegraph

Coding that led to lockdown was 'totally unreliable' and a 'buggy mess', say experts

The code, written by Professor Neil Ferguson and his team at Imperial College London, was impossible to read, scientists claim

Failures of an Influential COVID-19 Model Used to Justify Lockdowns

May 18, 2020 4 min read

Code Review of Ferguson's Model

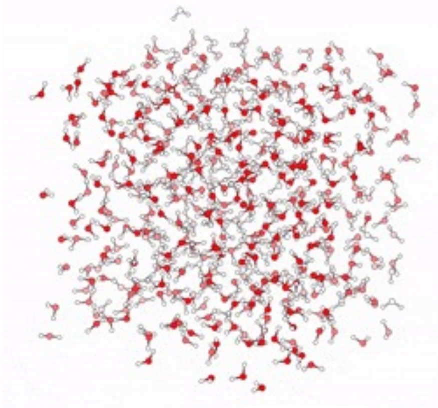
BY SUE DENIM 6 MAY 2020 3:16 PM

An open letter to software engineers criticizing Neil Ferguson's epidemics simulation code

2020-05-18 scientific software

The war over supercooled water

Palmer, Haji-Akbari, Singh, Martelli, Car, Panagiotopoulos, Debenedetti, J Chem Phys, 2018
Smart, "The war over super-cooled water," Physics Today, 2018



Video by Kmckiern

Does the ST2 model of liquid water below the freezing point have a liquid-liquid critical point?

NO
Limmer and Chandler
2011, 2013, 2016

YES
Palmer, Debenedetti, others
2014, 2018, 2018

Step in simulation violated equipartition of energy
→ artificially high temperature
→ just one instead of two phases

How to ensure quality simulations?

Thompson, Gilmer, Matsumoto, Quach, Shamprasad, Yang, Iacovella, McCabe, Cummings, Mol Phys, 2020

Transparent

Reproducible

Usable by others

Extensible

MolSSI

Molecular Simulation Design Framework
MoSDeF

NIST Standard Reference Simulation Website

Shen, Siderius, Krekelberg, Hatch, 2017-2024

Automated testing for physical validity

Merz and Shirts, PLOS One, 2018

Errors in scientific computing software

Category of error	Example	Intervention
Syntax	Not closing parentheses	Editor

Errors in scientific computing software

Category of error	Example	Intervention
Syntax	Not closing parentheses	Editor
Runtime	Accessing element in list that doesn't exist	Run the program, program gives error message
Semantic	Missing a minus sign, transposing tensor indices	Human inspection of the code; test-driven development; observing anomalous behavior

Errors in scientific computing software

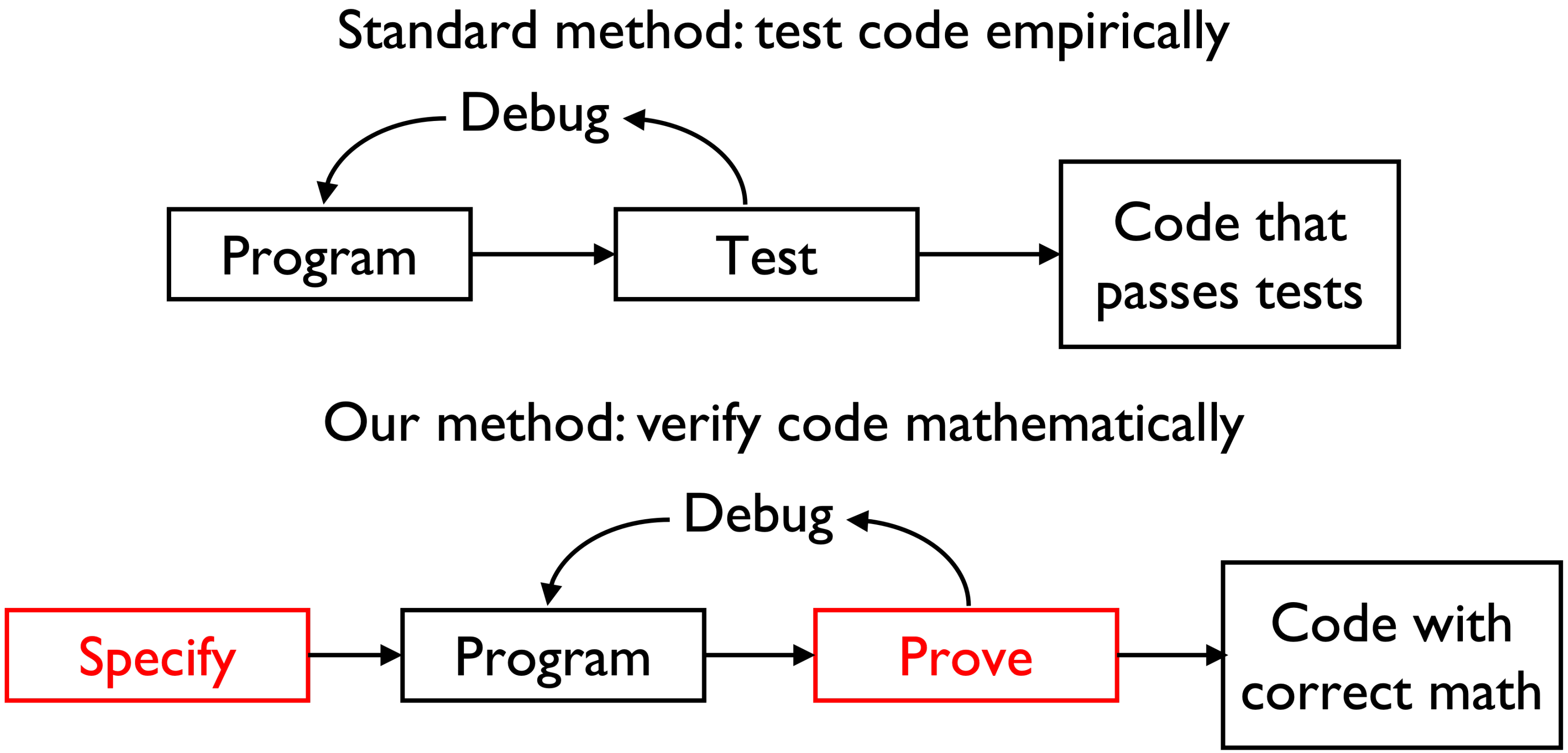
Category of error	Example	Intervention	Lean
Syntax	Not closing parentheses	Editor	Editor
Runtime	Accessing element in list that doesn't exist	Run the program, program gives error message	Editor
Semantic	Missing a minus sign, transposing tensor indices	Human inspection of the code; test-driven development; observing anomalous behavior	Editor

Errors in scientific computing software

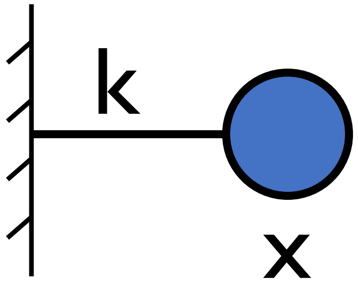
Category of error	Example	Intervention	Lean
Syntax	Not closing parentheses	Editor	Editor
Runtime	Accessing element in list that doesn't exist	Run the program, program gives error message	Editor
Semantic	Missing a minus sign, transposing tensor indices	Human inspection of the code; test-driven development; observing anomalous behavior	Editor
Floating point / Round off	Subtracting small values from large values	Checking energy conservation	

A vision for bug-free scientific computing

Selsam, Liang, Dill, "Developing Bug-Free Machine Learning Systems with Formal Mathematics," ICML 2017.



Example: mass on a spring



$$F = -kx \qquad E = k/2x^2 \qquad F = -\frac{\partial E}{\partial x}$$

In Python

```python
def force(x,k):
    return -k*x
def energy(x,k):
    return k/2*x**2
def test1():
    if force(5, 5) == -25:
        return 'Pass'
    else:
        return 'Fail'
test1()
```

In Lean

```lean
def force (k x : ℝ) : ℝ := -k * x
def energy (k x : ℝ) : ℝ := k/2*x^2


theorem force_is_derivative_of_energy :
∀ x : ℝ, deriv (fun x => energy k x) x = - force k x := by
```

Schedule for today

1. Provably-correct scientific computing
2. Derivations in science and engineering are math proofs
3. Formalizing mathematics with computers
4. Lean 4 and Mathlib
5. Case studies in proofs: adsorption and gas law thermodynamics
6. Case study in programming: bug-free BET analysis
7. Outlook
    1. LeanMD
    2. LLMs for theorem proving
    3. SciLib

Intermission

1. Getting connected with this course
2. Getting started with Lean
3. Proofs about equality

Adsorption

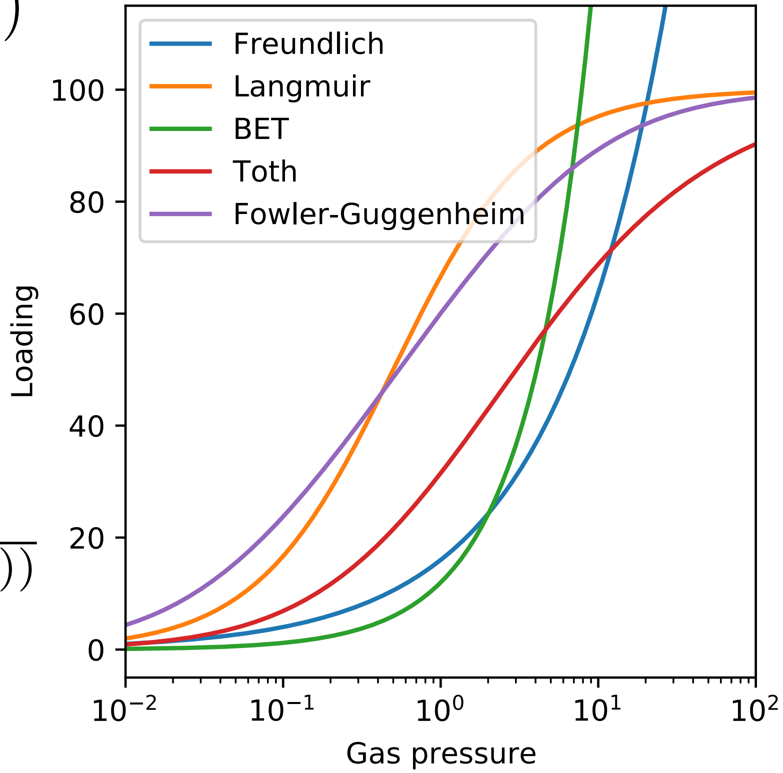
When molecules from a gas or liquid “stick” onto a solid material

a)

b)

c)

d)



e)

Freundlich	$q = K_{\mathrm{F}} p^n$
Langmuir	$q = \frac{q_{\max} K_{\mathrm{L}} p}{1+K_{\mathrm{L}} p}$
BET	$q = \frac{q_{\mathrm{m}} c_{\mathrm{BET}} p}{(p_0 - p)(1+(c_{\mathrm{BET}}-1)(p/p_0))}$
Toth	$q = \frac{q_{\max} p}{(b+p^t)^{1/t}}$
Fowler-Guggenheim	$K_{\mathrm{FG}} p = \frac{\theta}{1-\theta} \exp\left(\frac{2\theta w}{RT}\right)$

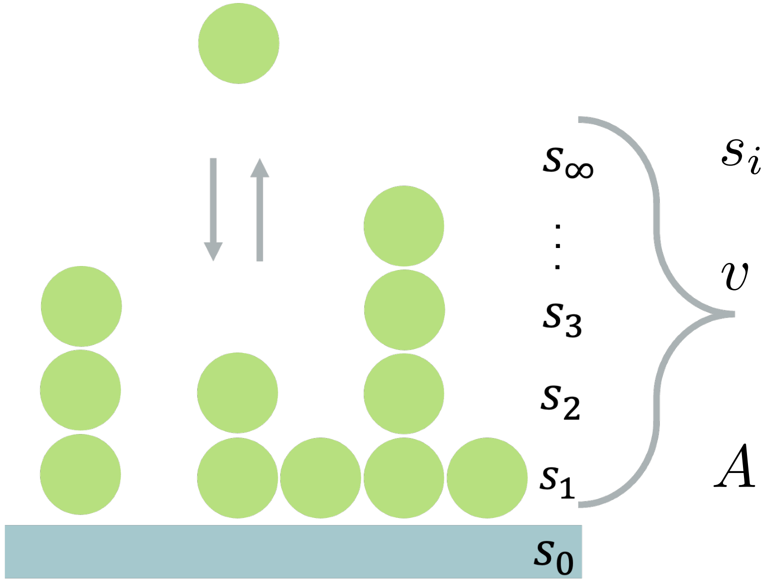
What is "theory"?

[CONTRIBUTION FROM THE BUREAU OF CHEMISTRY AND SOILS AND GEORGE WASHINGTON UNIVERSITY]

Adsorption of Gases in Multimolecular Layers

BY STEPHEN BRUNAUER, P. H. EMMETT AND EDWARD TELLER

$$v = \frac{v_m c p}{(p_0 - p)[1 + (c-1)(p/p_0)]}$$



$$s_i = C x^i s_0$$

$$v = v_0 \sum_{i=0}^{\infty} i s_i$$

$$A = v_0 \sum_{i=0}^{\infty} s_i$$

BET Adsorption

II. Generalization of Langmuir's Theory to Multimolecular Adsorption

With the help of a few simplifying assumptions it is possible to carry out an isotherm derivation for multimolecular layers that is similar to Langmuir's derivation for unimolecular layers.

In carrying out this derivation we shall let s_0, s_1, s_2, . . . s_i, . . . represent the surface area that is covered by only 0, 1, 2, . . . i, . . . layers of adsorbed molecules. Since at equilibrium s_0 must remain constant the rate of condensation on the bare surface is equal to the rate of evaporation from the first layer

$$a_1 p s_0 = b_1 s_1 e^{-E_1/RT} \qquad (10)$$

where p is the pressure, E_1 is the heat of adsorption of the first layer, and a_1 and b_1 are constants. This is essentially Langmuir's equation for unimolecular adsorption, and involves the assumption that a_1, b_1, and E_1 are independent of the number of adsorbed molecules already present in the first layer.

...

of adsorbed gas. It follows that

$$\frac{v}{Av_0} = \frac{v}{v_m} = \frac{\sum_{i=0}^{\infty} i s_i}{\sum_{i=0}^{\infty} s_i} \qquad (15)$$

What is "theory"?

[CONTRIBUTION FROM THE BUREAU OF CHEMISTRY AND SOILS AND GEORGE WASHINGTON UNIVERSITY]

Adsorption of Gases in Multimolecular Layers

BY STEPHEN BRUNAUER, P. H. EMMETT AND EDWARD TELLER

Defining a model

II. Generalization of Langmuir's Theory to Multimolecular Adsorption

With the help of a few simplifying assumptions it is possible to carry out an isotherm derivation for multimolecular layers that is similar to Langmuir's derivation for unimolecular layers.

In carrying out this derivation we shall let s_0, s_1, s_2, . . . s_i, . . . represent the surface area that is covered by only 0, 1, 2, . . . i, . . . layers of adsorbed molecules. Since at equilibrium s_0 must remain constant the rate of condensation on the bare surface is equal to the rate of evaporation from the first layer

$$a_1 p s_0 = b_1 s_1 e^{-E_1/RT} \qquad (10)$$

where p is the pressure, E_1 is the heat of adsorption of the first layer, and a_1 and b_1 are constants. This is essentially Langmuir's equation for unimolecular adsorption, and involves the assumption that a_1, b_1, and E_1 are independent of the number of adsorbed molecules already present in the first layer.

. . .

of adsorbed gas. It follows that

$$\frac{v}{Av_0} = \frac{v}{v_m} = \frac{\sum_{i=0}^{\infty} i s_i}{\sum_{i=0}^{\infty} s_i} \qquad (15)$$

What is “theory”?

[CONTRIBUTION FROM THE BUREAU OF CHEMISTRY AND SOILS AND GEORGE WASHINGTON UNIVERSITY]

Adsorption of Gases in Multimolecular Layers

BY STEPHEN BRUNAUER, P. H. EMMETT AND EDWARD TELLER

Defining a model

Expressing terms mathematically

II. Generalization of Langmuir's Theory to Multimolecular Adsorption

With the help of a few simplifying assumptions it is possible to carry out an isotherm derivation for multimolecular layers that is similar to Langmuir's derivation for unimolecular layers.

In carrying out this derivation we shall let s_0, s_1, s_2, . . . s_i, . . . represent the surface area that is covered by only 0, 1, 2, . . . i, . . . layers of adsorbed molecules. Since at equilibrium s_0 must remain constant the rate of condensation on the bare surface is equal to the rate of evaporation from the first layer

$$a_1 p s_0 = b_1 s_1 e^{-E_1/RT} \qquad (10)$$

where p is the pressure, E_1 is the heat of adsorption of the first layer, and a_1 and b_1 are constants. This is essentially Langmuir's equation for unimolecular adsorption, and involves the assumption that a_1, b_1, and E_1 are independent of the number of adsorbed molecules already present in the first layer.

…

of adsorbed gas. It follows that

$$\frac{v}{Av_0} = \frac{v}{v_m} = \frac{\sum_{i=0}^{\infty} i s_i}{\sum_{i=0}^{\infty} s_i} \qquad (15)$$

What is “theory”?

Feb., 1938 ADSORPTION OF GASES IN MULTIMOLECULAR LAYERS 309

[CONTRIBUTION FROM THE BUREAU OF CHEMISTRY AND SOILS AND GEORGE WASHINGTON UNIVERSITY]

Adsorption of Gases in Multimolecular Layers

BY STEPHEN BRUNAUER, P. H. EMMETT AND EDWARD TELLER

Defining a model

Expressing terms mathematically

Specifying variables

II. Generalization of Langmuir's Theory to Multimolecular Adsorption

With the help of a few simplifying assumptions it is possible to carry out an isotherm derivation for multimolecular layers that is similar to Langmuir's derivation for unimolecular layers.

In carrying out this derivation we shall let s_0, s_1, s_2, . . . s_i, . . . represent the surface area that is covered by only 0, 1, 2, . . . i, . . . layers of adsorbed molecules. Since at equilibrium s_0 must remain constant the rate of condensation on the bare surface is equal to the rate of evaporation from the first layer

$$a_1 p s_0 = b_1 s_1 e^{-E_1/RT} \qquad (10)$$

where p is the pressure, E_1 is the heat of adsorption of the first layer, and a_1 and b_1 are constants. This is essentially Langmuir's equation for unimolecular adsorption, and involves the assumption that a_1, b_1, and E_1 are independent of the number of adsorbed molecules already present in the first layer.

. . .

of adsorbed gas. It follows that

$$\frac{v}{Av_0} = \frac{v}{v_m} = \frac{\sum_{i=0}^{\infty} i s_i}{\sum_{i=0}^{\infty} s_i} \qquad (15)$$

What is "theory"?

Feb., 1938 ADSORPTION OF GASES IN MULTIMOLECULAR LAYERS 309

[CONTRIBUTION FROM THE BUREAU OF CHEMISTRY AND SOILS AND GEORGE WASHINGTON UNIVERSITY]

Adsorption of Gases in Multimolecular Layers

BY STEPHEN BRUNAUER, P. H. EMMETT AND EDWARD TELLER

Defining a model

Expressing terms mathematically

Specifying variables

Making assumptions

II. Generalization of Langmuir's Theory to Multimolecular Adsorption

With the help of a few simplifying assumptions it is possible to carry out an isotherm derivation for multimolecular layers that is similar to Langmuir's derivation for unimolecular layers.

In carrying out this derivation we shall let s_0, s_1, s_2, . . . s_i, . . . represent the surface area that is covered by only 0, 1, 2, . . . i, . . . layers of adsorbed molecules. Since at equilibrium s_0 must remain constant the rate of condensation on the bare surface is equal to the rate of evaporation from the first layer

$$a_1 p s_0 = b_1 s_1 e^{-E_1/RT} \quad (10)$$

where p is the pressure, E_1 is the heat of adsorption of the first layer, and a_1 and b_1 are constants. This is essentially Langmuir's equation for unimolecular adsorption, and involves the assumption that a_1, b_1, and E_1 are independent of the number of adsorbed molecules already present in the first layer.

. . .

of adsorbed gas. It follows that

$$\frac{v}{Av_0} = \frac{v}{v_m} = \frac{\sum_{i=0}^{\infty} i s_i}{\sum_{i=0}^{\infty} s_i} \quad (15)$$

What is "theory"?

Feb., 1938 ADSORPTION OF GASES IN MULTIMOLECULAR LAYERS 309

[CONTRIBUTION FROM THE BUREAU OF CHEMISTRY AND SOILS AND GEORGE WASHINGTON UNIVERSITY]

Adsorption of Gases in Multimolecular Layers

BY STEPHEN BRUNAUER, P. H. EMMETT AND EDWARD TELLER

Defining a model

Expressing terms mathematically

Specifying variables

Making assumptions

Deriving new terms

II. Generalization of Langmuir's Theory to Multimolecular Adsorption

With the help of a few simplifying assumptions it is possible to carry out an isotherm derivation for multimolecular layers that is similar to Langmuir's derivation for unimolecular layers.

In carrying out this derivation we shall let s_0, s_1, s_2, . . . s_i, . . . represent the surface area that is covered by only 0, 1, 2, . . . i, . . . layers of adsorbed molecules. Since at equilibrium s_0 must remain constant the rate of condensation on the bare surface is equal to the rate of evaporation from the first layer

$$a_1 p s_0 = b_1 s_1 e^{-E_1/RT} \qquad (10)$$

where p is the pressure, E_1 is the heat of adsorption of the first layer, and a_1 and b_1 are constants. This is essentially Langmuir's equation for unimolecular adsorption, and involves the assumption that a_1, b_1, and E_1 are independent of the number of adsorbed molecules already present in the first layer.

. . .

of adsorbed gas. It follows that

$$\frac{v}{Av_0} = \frac{v}{v_m} = \frac{\sum_{i=0}^{\infty} i s_i}{\sum_{i=0}^{\infty} s_i} \qquad (15)$$

What is “theory”?

[CONTRIBUTION FROM THE BUREAU OF CHEMISTRY AND SOILS AND GEORGE WASHINGTON UNIVERSITY]

Adsorption of Gases in Multimolecular Layers

BY STEPHEN BRUNAUER, P. H. EMMETT AND EDWARD TELLER

Defining a model

Expressing terms mathematically

Specifying variables

Making assumptions

Deriving new terms

Relating to other theories

II. Generalization of Langmuir's Theory to Multimolecular Adsorption

With the help of a few simplifying assumptions it is possible to carry out an isotherm derivation for multimolecular layers that is similar to Langmuir's derivation for unimolecular layers.

In carrying out this derivation we shall let s_0, s_1, s_2, . . . s_i, . . . represent the surface area that is covered by only 0, 1, 2, . . . i, . . . layers of adsorbed molecules. Since at equilibrium s_0 must remain constant the rate of condensation on the bare surface is equal to the rate of evaporation from the first layer

$$a_1 p s_0 = b_1 s_1 e^{-E_1/RT} \quad (10)$$

where p is the pressure, E_1 is the heat of adsorption of the first layer, and a_1 and b_1 are constants. This is essentially Langmuir's equation for unimolecular adsorption, and involves the assumption that a_1, b_1, and E_1 are independent of the number of adsorbed molecules already present in the first layer.

...

of adsorbed gas. It follows that

$$\frac{v}{Av_0} = \frac{v}{v_m} = \frac{\sum_{i=0}^{\infty} i s_i}{\sum_{i=0}^{\infty} s_i} \quad (15)$$

Making scientific theories executable

Excerpt from informal derivation in Langmuir, *JACS*, 1918

flection. Therefore, the rate of condensation of the gas on the surface is $\alpha\theta\mu$, where θ represents the fraction of the surface which is bare. Similarly the rate of evaporation of the molecules from the surface is equal to $\nu_1\theta_1$, where ν_1 is the rate at which the gas would evaporate if the surface were completely covered and θ_1 is the fraction actually covered by the adsorbed molecules. When a gas is in equilibrium with a surface these two rates must be equal, so we have

$$\alpha\theta\mu = \nu_1\theta_1. \tag{4}$$

Furthermore,

$$\theta + \theta_1 = 1 \tag{5}$$

whence

$$\theta_1 = \frac{\alpha\mu}{\nu_1 + \alpha\,\mu}. \tag{6}$$

Let us place

$$\frac{\alpha}{\nu_1} = \sigma_1. \tag{7}$$

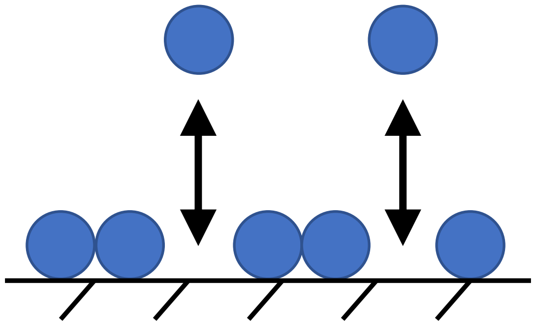
Equation 6 then becomes

$$\theta_1 = \frac{\sigma_1\mu}{1 + \sigma_1\mu}. \tag{8}$$

Formal derivation in Lean

```
-- Imports theory of real numbers
import Mathlib.Data.Real.Basic
-- Declares theorem and its arguments
theorem LangmuirAdsorption {θ K P r_ad r_d k_ad k_d A S_tot S : ℝ}
-- Premises
(hrad : r_ad = k_ad * P * S) -- Adsorption rate expression
(hrd : r_d = k_d * A) -- Desorption rate expression
(heq : r_ad = r_d) -- Equilibrium assumption
(hK : K = k_ad / k_d) -- Definition of adsorption constant
(hS_tot : S_tot = S + A) -- Site balance
(hθ : θ = A / S_tot) -- Definition of fractional coverage
-- Constraints
(hc1 : S + A ≠ 0)
(hc2 : k_d + k_ad * P ≠ 0)
(hc3 : k_d ≠ 0)
:
θ = K * P / (1 + K * P) -- Langmuir's adsorption law
:= by -- Proof starts here
  rw [hrad, hrd] at heq
  rw [hθ, hS_tot, hK]
  field_simp
  calc
    A * (k_d + k_ad * P) = k_d * A + k_ad * P * A := by ring
    _ = k_ad * P * S + k_ad * P * A := by rw[heq]
    _ = k_ad * P * (S + A) := by ring
```

Derivations in science are math proofs



Langmuir Adsorption

Langmuir, *JACS*, 1918

Proposition

5 premises — imply → conjecture

Site balance:	$S_0 = S + S_\mathrm{a}$
Adsorption rate model:	$r_\mathrm{ads} = k_\mathrm{ads} \cdot p \cdot S$
Desorption rate model:	$r_\mathrm{des} = k_\mathrm{des} \cdot S_\mathrm{a}$
Equilibrium assumption:	$r_\mathrm{ads} = r_\mathrm{des}$
Mass balance	$q = S_\mathrm{a}$

$$q = \frac{S_0 K_{eq} p}{1 + K_{eq} p}$$

Theorem

Proposition is TRUE

Proof

Derivation using algebraic manipulations (substitution, cancelling terms, etc.)

✓ ____
✓ ____
✓ ____
✓ ____

Schedule for today

1. Provably-correct scientific computing
2. Derivations in science and engineering are math proofs
3. Formalizing mathematics with computers
4. Lean 4 and Mathlib
5. Case studies in proofs: adsorption and gas law thermodynamics
6. Case study in programming: bug-free BET analysis
7. Outlook
    1. LeanMD
    2. LLMs for theorem proving
    3. SciLib

Intermission

1. Getting connected with this course
2. Getting started with Lean
3. Proofs about equality

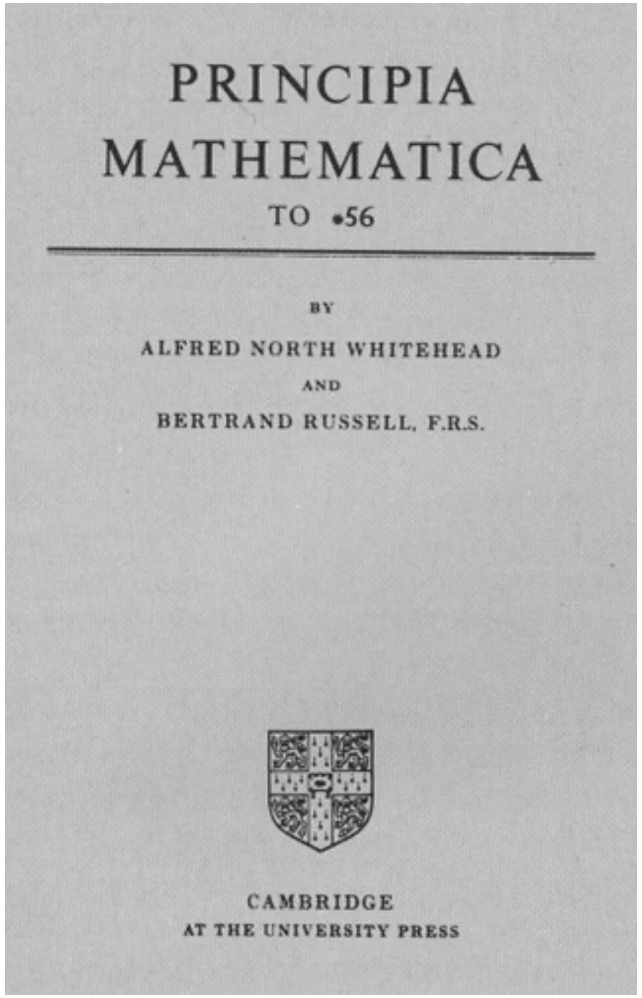
Two kinds of math proofs

Thomas C Hales. Formal proof. Notices of the AMS, 2008.

Handwritten proofs	**Formal proofs**
Informal syntax	Strict, computer language syntax
Only readable for human	Machine-readable and executable
Might exclude information	Cannot miss assumptions or steps
Might contain mistakes	Rigorously verified by computer
Requires humans to proofread	Automated proof checking
Easy to write	Challenging to write

The axiomatic perspective: *Principia Mathematica*

Alfred North Whitehead and Bertrand Russell, 1910-1927


PRINCIPIA
MATHEMATICA
TO ∗56

BY
ALFRED NORTH WHITEHEAD
AND
BERTRAND RUSSELL, F.R.S.

CAMBRIDGE
AT THE UNIVERSITY PRESS

Precisely express mathematics in symbolic logic

Minimize number of axioms and inference rules

∗110·643. $\vdash . 1 +_c 1 = 2$

Dem.

$\vdash . *110 \cdot 632 . *101 \cdot 21 \cdot 28 . \supset$

$\vdash . 1 +_c 1 = \hat{\xi}\{(\exists y) . y \in \xi . \xi - \iota \text{‘} y \in 1\}$

$[*54 \cdot 3] \quad = 2 . \supset \vdash . \text{Prop}$

The above proposition is occasionally useful. It is used at least three times, in ∗113·66 and ∗120·123·472.

Volume II, page 86: 1 + 1 = 2

"The above proposition is occasionally useful.
It is used at least three times."

Zermelo-Frenkel set theory (1922)

1. Extensionality
2. Regularity
3. Specification
4. Pairing
5. Union
6. Replacement
7. Infinity
8. Power set
9. Well-ordering
10. Choice

~Any mathematical expression and proof

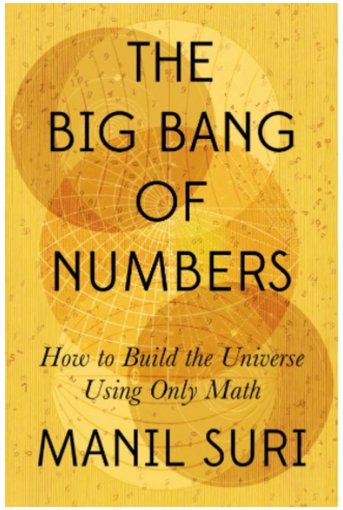
Zermelo-Frenkel set theory (1922)

1. Extensionality
2. Regularity
3. Specification
4. Pairing
5. Union
6. Replacement
7. Infinity
8. Power set
9. Well-ordering
10. Choice

1. Two sets are equal if they have the same elements.

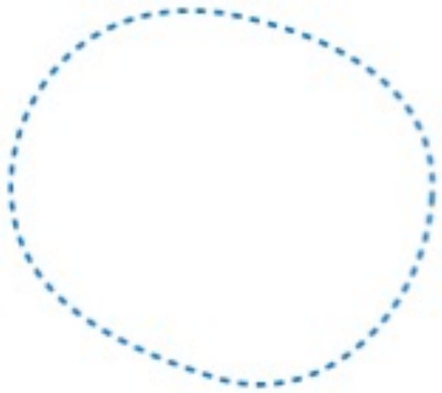
4. If x and y are sets, then there exists a set which contains x and y as elements.

7. There exists a set having infinitely many members



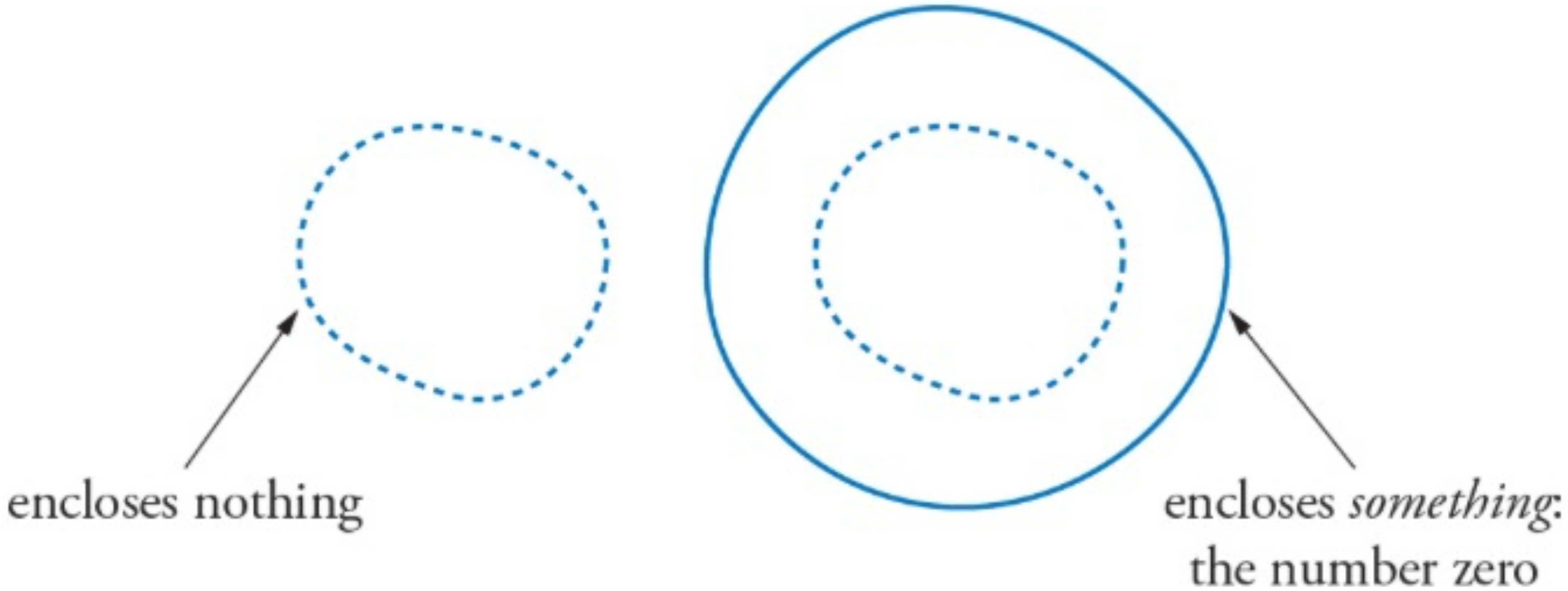
How to count with sets

the empty set

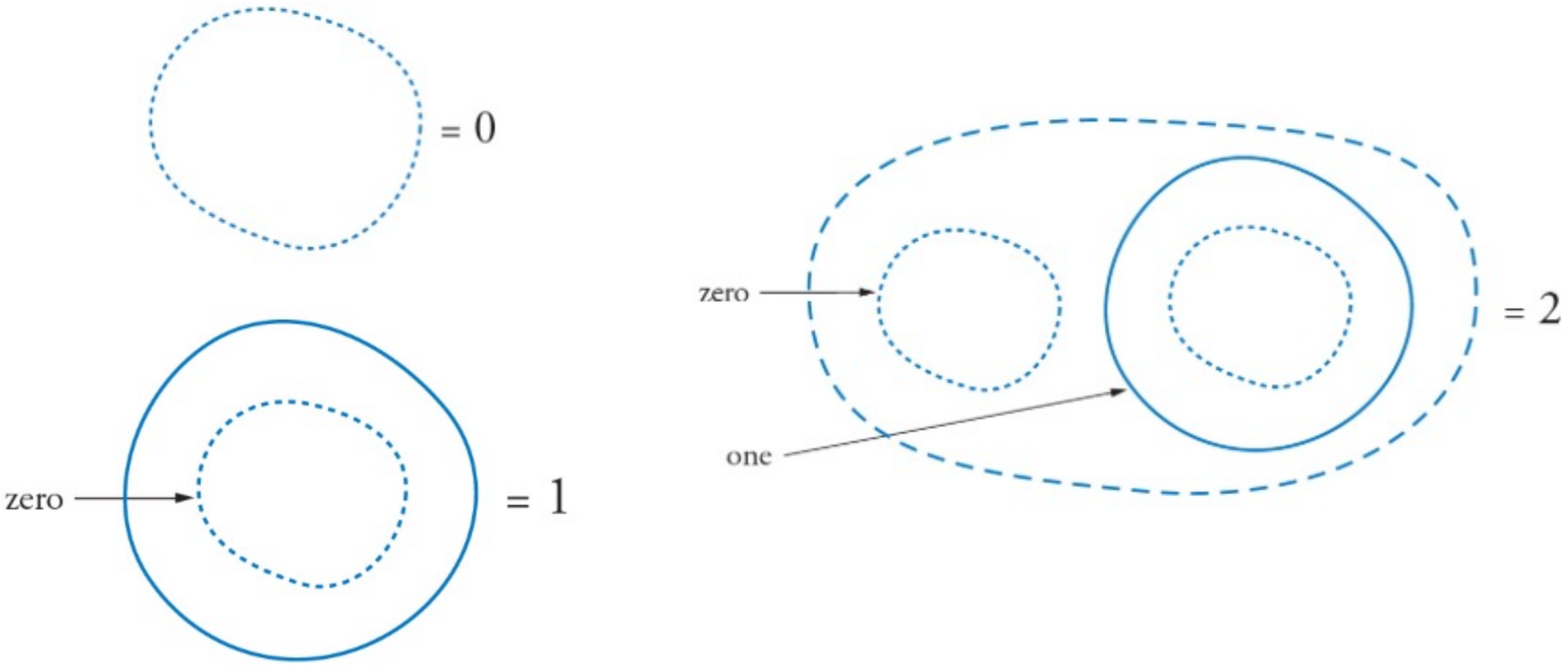


= 0

How to count with sets



How to count with sets



Constructing the natural numbers with set theory

$$
\begin{aligned}
0 &= \{\} && = \emptyset, \\
1 &= \{0\} && = \{\emptyset\}, \\
2 &= \{0, 1\} && = \{\emptyset, \{\emptyset\}\}, \\
3 &= \{0, 1, 2\} && = \{\emptyset, \{\emptyset\}, \{\emptyset, \{\emptyset\}\}\}
\end{aligned}
$$

Constructing the natural numbers with set theory

$$
\begin{aligned}
0 &= \{\} &&= \emptyset, \\
1 &= \{0\} &&= \{\emptyset\}, \\
2 &= \{0, 1\} &&= \{\emptyset, \{\emptyset\}\}, \\
3 &= \{0, 1, 2\} &&= \{\emptyset, \{\emptyset\}, \{\emptyset, \{\emptyset\}\}\}
\end{aligned}
$$

Formal definition of counting is "succession"

$$S(0) = 1$$

$$S(1) = 2$$

Constructing the natural numbers with set theory

$$
\begin{aligned}
0 &= \{\} &&= \emptyset, \\
1 &= \{0\} &&= \{\emptyset\}, \\
2 &= \{0, 1\} &&= \{\emptyset, \{\emptyset\}\}, \\
3 &= \{0, 1, 2\} &&= \{\emptyset, \{\emptyset\}, \{\emptyset, \{\emptyset\}\}\}
\end{aligned}
$$

Formal definition of counting is "succession"

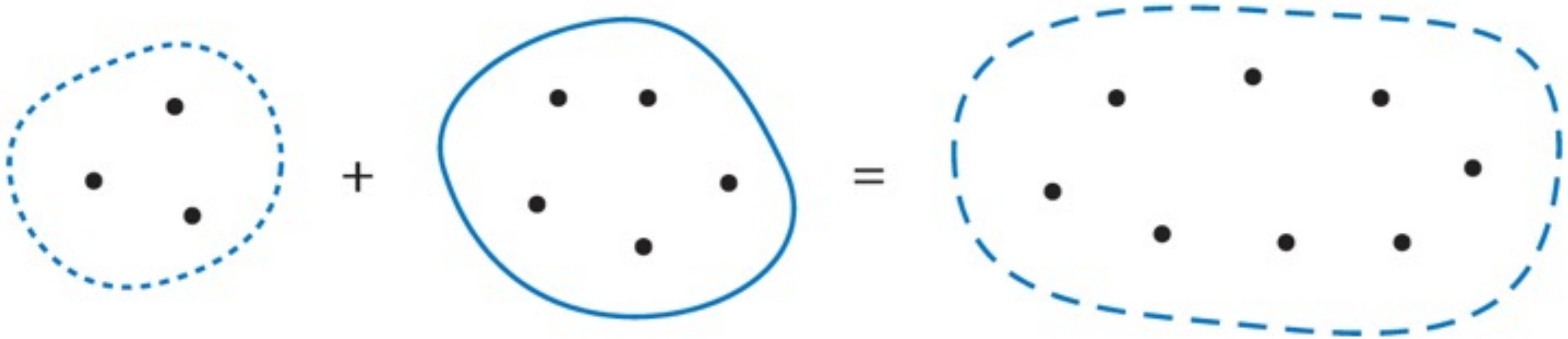
$$S(0) = 1$$

$$S(1) = 2$$

Natural numbers are defined recursively using succession and the empty set

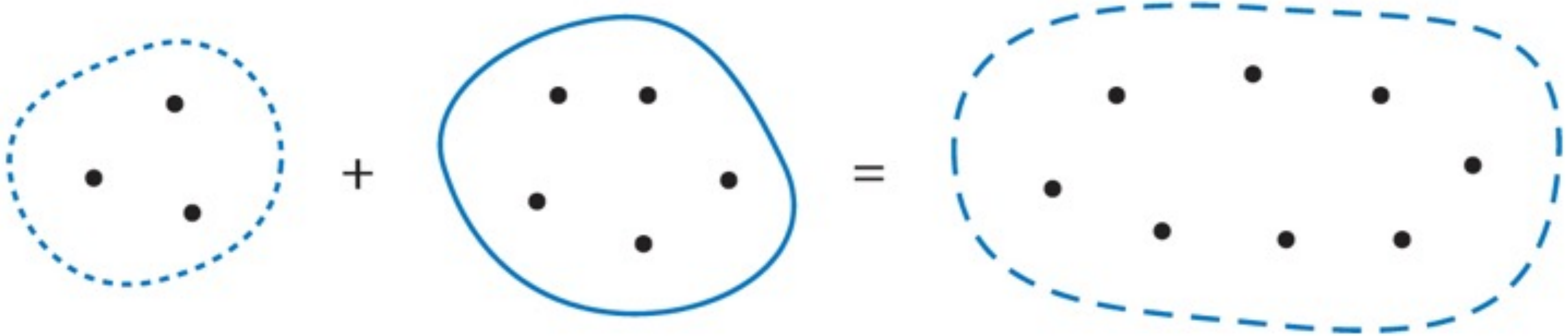
$$
\mathbb{N} \quad
\begin{aligned}
0 &= \{\} \\
n + 1 &= S(n) = n \cup \{n\}
\end{aligned}
$$

Defining math operations



Addition isn't a stand-alone rule; we define it using the axioms and rules of logic
Addition is *built on top of* succession

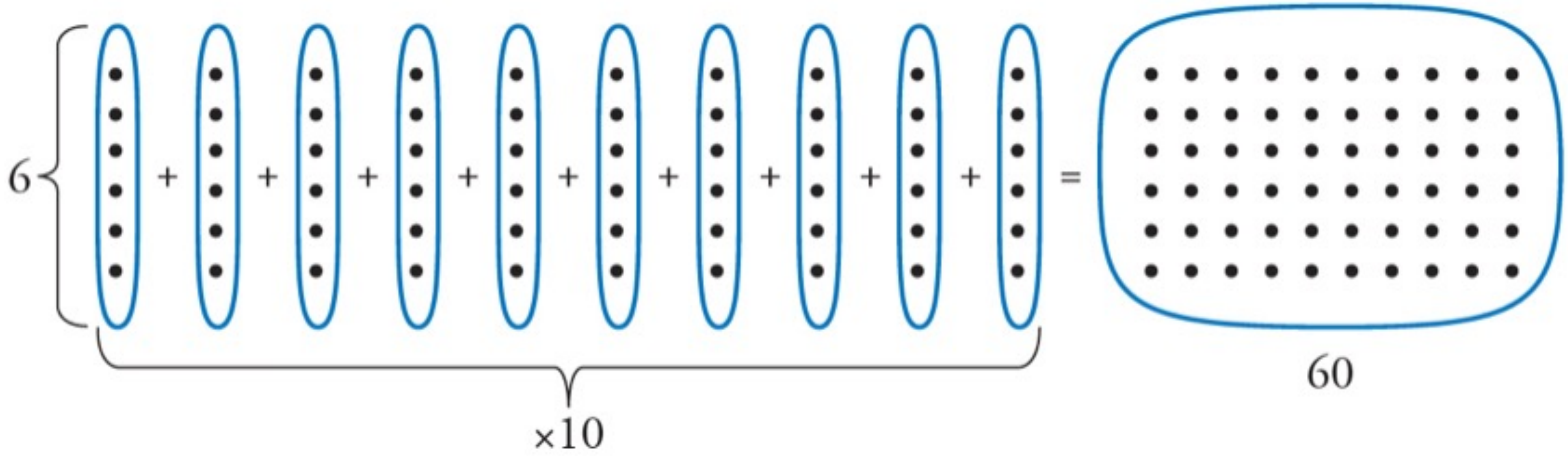
Defining math operations



Addition isn't a stand-alone rule; we define it using the axioms and rules of logic
Addition is *built on top of* succession, being defined recursively as

$$m + 0 = m,$$
$$m + S(n) = S(m + n)$$

Defining math operations

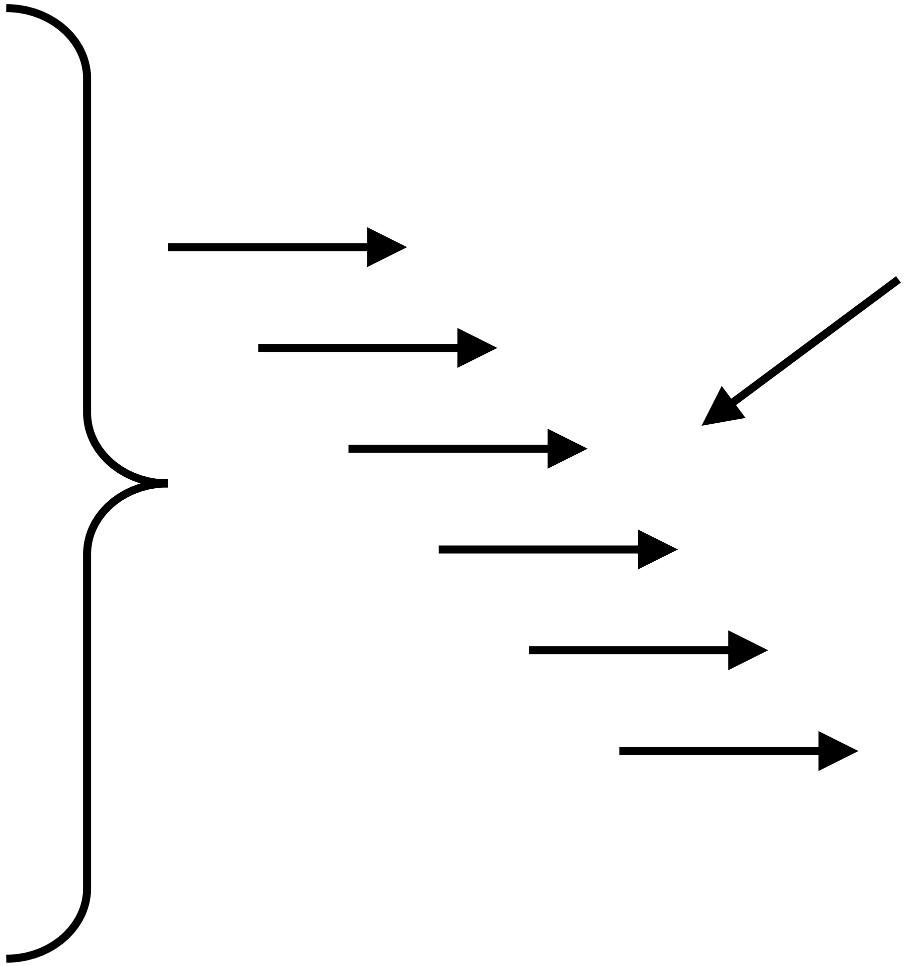


Multiplication is *built on top of* addition, defined recursively as

$$m * 0 = 0,$$
$$m * S(n) = m * n + m$$

How do we *actually* construct mathematics?

1. Extensionality
2. Regularity
3. Specification
4. Pairing
5. Union
6. Replacement
7. Infinity
8. Power set
9. Well-ordering
10. Choice



"A wide gulf separates traditional proof from formal proof."

– Thomas Hales, 2008

~Any mathematical expression and proof

Proving theorems with computers

1954
Davis

First proof that sum of even numbers is even

1968
de Bruijn

Construction of real numbers using Dedekind cuts

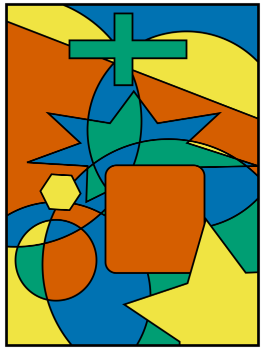
Proving theorems with computers

1954
Davis

First proof that sum of even numbers is even

1968
de Bruijn

Construction of real numbers using Dedekind cuts

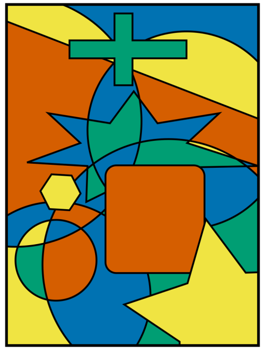


1977
Appel, Haken, Koch

Four color theorem (brute force)

1,482 reducible configurations checked one-by-one by computer
>400 pages!

Proving theorems with computers



1977
Appel, Haken, Koch

Four color theorem (brute force)

1980s
New software: HOL and Coq
Type theory becomes more popular among theorem provers than set theory

1954
Davis

First proof that sum of even numbers is even

1968
de Bruijn

Construction of real numbers using Dedekind cuts

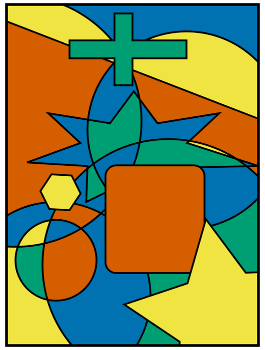
1986
Shankar

Gödel's first incompleteness theorem

1996
Harrison

Fundamental theorem of integral calculus

Proving theorems with computers



1977
Appel, Haken, Koch

Four color theorem (brute force)

2004
Gonthier

Four color theorem (axiomatic)

1954
Davis

First proof that sum of even numbers is even

1968
de Bruijn

Construction of real numbers using Dedekind cuts

1986
Shankar

Gödel's first incompleteness theorem

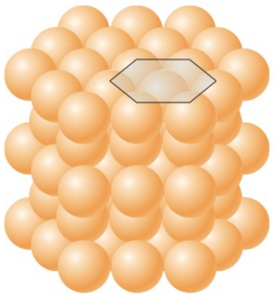
1996
Harrison

Fundamental theorem of integral calculus

"[The two proofs] differ in the same way that adding 1+1=2 on a calculator differs from the mathematical justification of 1+1=2 by definitions, recursions, and a rigorous construction of the natural numbers."

Thomas Hales, 2008

Proving theorems with computers



1998
Hales, Ferguson

Kepler conjecture (brute force)

2017
Hales, et al.

Kepler conjecture (axiomatic)

1954
Davis

First proof that sum of even numbers is even

1968
de Bruijn

Construction of real numbers using Dedekind cuts

1986
Shankar

Gödel's first incompleteness theorem

1996
Harrison

Fundamental theorem of integral calculus

Proving theorems with computers

1954
Davis
First proof that sum of even numbers is even

1968
de Bruijn
Construction of real numbers using Dedekind cuts

1977
Appel, Haken, Koch
Four color theorem (brute force)

1986
Shankar
Gödel's first incompleteness theorem

1996
Harrison
Fundamental theorem of integral calculus

1998
Hales, Ferguson
Kepler conjecture (brute force)

2004
Gonthier
Four color theorem (axiomatic)

2012
Scholze
Perfectoid spaces introduced

2017
Hales, et al.
Kepler conjecture (axiomatic)

2019
Buzzard, Commelin, Massot
Perfectoid spaces formalized

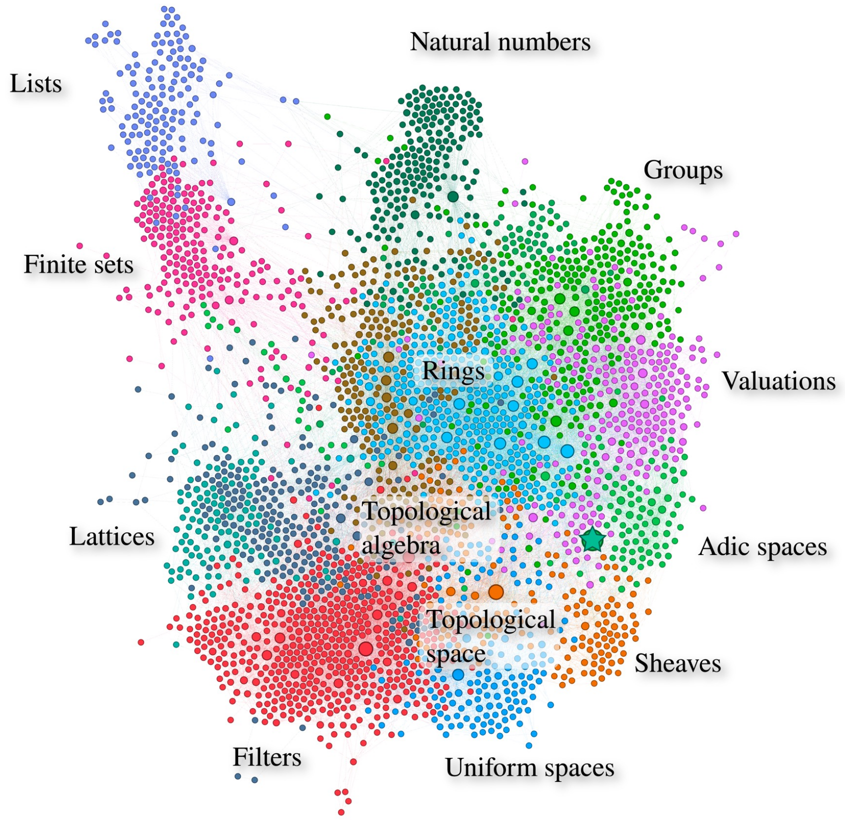
Formalizing Perfectoid Spaces

P. Scholze. Perfectoid spaces. arXiv:1111.4914, 2011.
K. Buzzard, J. Commelin, and P. Massot. ACM SIGPLAN, 2020.

Perfectoid spaces: **2018 Fields Medal**

"To define a perfectoid space, the three mathematicians had to combine more than 3,000 definitions of other mathematical objects and 30,000 connections between them. The definitions sprawled across many areas of math, from algebra to topology to geometry."

from "*Building the mathematical library of the future*", Kevin Hartnett, Quanta magazine, 10/01/2020



Visualizing the definitions and theorems required to establish perfectoid spaces, by Patrick Massot

Faster than peer review?

In early 2022, Thomas Bloom solved a problem posed by Paul Erdős and Ronald Graham.

The headline in Quanta read "Math's 'Oldest Problem Ever' Gets a New Answer."

Within in a few months, Bloom and Bhavik Mehta verified the correctness of the proof in Lean.

Example highlighted by Jeremy Avigad at ASL

Faster than peer review?

Timothy Gowers @wtgowers@mathstodon.xyz
@wtgowers

Very excited that Thomas Bloom and Bhavik Mehta have done this. I think it's the first time that a serious contemporary result in "mainstream" mathematics doesn't have to be checked by a referee, because it has been checked formally. Maybe the sign of things to come ... 1/

> **Kevin Buzzard** @XenaProject · Jun 12, 2022
>
> Happy to report that Bloom went on to learn Lean this year and, together with Bhavik Mehta, has now formalised his proof in Lean b-mehta.github.io/unit-fractions/ (including formalising the Hardy-Littlewood circle method), finishing before he got a referee's report for the paper ;-)
>
> Show this thread

5:12 AM · Jun 13, 2022

25 Retweets **1** Quote Tweet **138** Likes

Example highlighted by Jeremy Avigad at ASL

Schedule for today

1. Provably-correct scientific computing
2. Derivations in science and engineering are math proofs
3. Formalizing mathematics with computers
4. Lean 4 and Mathlib
5. Case studies in proofs: adsorption and gas law thermodynamics
6. Case study in programming: bug-free BET analysis
7. Outlook
    1. LeanMD
    2. LLMs for theorem proving
    3. SciLib

Intermission

1. Getting connected with this course
2. Getting started with Lean
3. Proofs about equality

Lean theorem prover and programming language

Coquand and Huet, PhD thesis, INRIA, 1986.
de Moura, Kong, Avigad, van Doorn, von Raumer, CADE 25, 2015.

Mathematics constructed from dependent type theory

Trusted kernel with just 6k lines of code

→ >150k theorems
→ >1.5 million lines of verified proofs

Tactics to facilitate proof automation

Compile Lean code to efficient C code

"We're going to digitize mathematics, and it's going to make it better."
– Kevin Buzzard, Imperial College London

Lean's mathematical library: Mathlib

What do we need for the real numbers?

Real numbers include

-1, 3.6, Euler's number, π, $\sqrt{2}$, *etc.*

Lean's mathematical library: Mathlib

What do we need for the real numbers?

```
import Mathlib.Data.Real.Basic
```

Real numbers include

-1, 3.6, Euler's number, π, $\sqrt{2}$, *etc.*

Lean's mathematical library: Mathlib

What do we need for the real numbers?

Real numbers include

-1, 3.6, Euler's number, π, $\sqrt{2}$, *etc.*

```
import Mathlib.Data.Real.Basic
```

What about Stirling's Approximation?

https://en.wikipedia.org/wiki/Stirling's_approximation

$$\ln(n!) = n \ln n - n + \mathcal{O}(\ln n)$$

Lean's mathematical library: Mathlib

What do we need for the real numbers?

Real numbers include

-1, 3.6, Euler's number, π, $\sqrt{2}$, *etc.*

```
import Mathlib.Data.Real.Basic
```

What about Stirling's Approximation?

https://en.wikipedia.org/wiki/Stirling's_approximation

$$\ln(n!) = n\ln n - n + \mathcal{O}(\ln n)$$

```
import Mathlib.Analysis.SpecialFunctions.Stirling
```

Lean's mathematical library: Mathlib

What do we need for the real numbers?

```
import Mathlib.Data.Real.Basic
```

Real numbers include

-1, 3.6, Euler's number, π, $\sqrt{2}$, *etc.*

What about Stirling's Approximation?

https://en.wikipedia.org/wiki/Stirling's_approximation

$$\ln(n!) = n \ln n - n + \mathcal{O}(\ln n)$$

```
import Mathlib.Analysis.SpecialFunctions.Stirling
```

https://eric-wieser.github.io/mathlib-import-graph/

Boyle's Law

```
import Mathlib.Data.Real.Basic

-- Variables
theorem Boyle {P1 P2 V1 V2 T1 T2 n1 n2 R : ℝ}

-- Assumptions
(h1: P1*V1 = n1*R*T1)
(h2: P2*V2 = n2*R*T2)
(h3: T1=T2)
(h4: n1=n2) :

-- Conjecture
(P1*V1 = P2*V2) :=

-- Proof
by
rw [h3] at h1
rw [h4] at h1
rw [← h2] at h1
exact h1
```

Prove that an ideal gas follows Boyle's Law

$$PV = nRT$$

$$T_1 = T_2$$

$$n_1 = n_2$$

$$P_1V_1 = P_2V_2$$

LIVE

Schedule for today

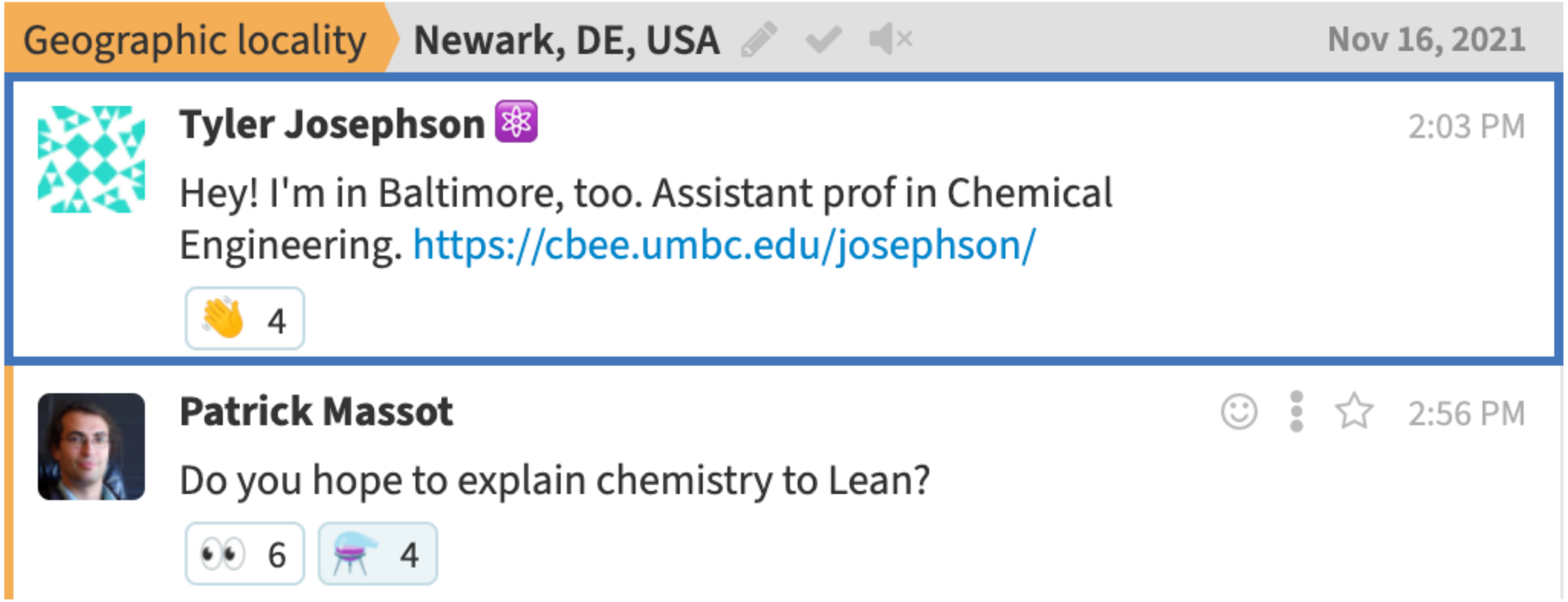
1. Provably-correct scientific computing
2. Derivations in science and engineering are math proofs
3. Formalizing mathematics with computers
4. Lean 4 and Mathlib
5. Case studies in proofs: adsorption and gas law thermodynamics
6. Case study in programming: bug-free BET analysis
7. Outlook
    1. LeanMD
    2. LLMs for theorem proving
    3. SciLib

Intermission

1. Getting connected with this course
2. Getting started with Lean
3. Proofs about equality

Can we explain chemistry to Lean?

Zulip Online Forum



Formalizing Chemical Physics

Bobbin, Sharlin, Feyzishendi, Dang, Wraback, Josephson, Digital Discovery, 2024

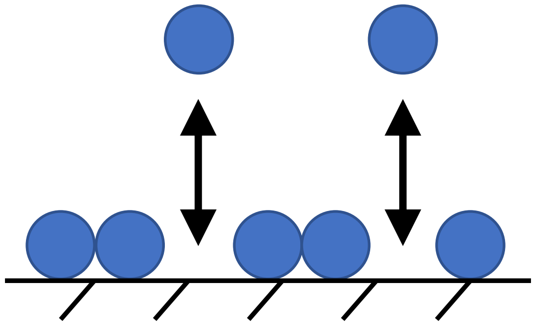
Caution: Proofs written in Lean 3, not Lean 4

Derivations of Langmuir and BET adsorption theory

Logical connections among gas laws

Deriving the kinematic equations using calculus

Formalizing Langmuir's theory of adsorption



Langmuir Adsorption

Langmuir, *JACS*, 1918

Site balance:	$S_0 = S + S_\mathrm{a}$
Adsorption rate model:	$r_\mathrm{ads} = k_\mathrm{ads} \cdot p \cdot S$
Desorption rate model:	$r_\mathrm{des} = k_\mathrm{des} \cdot S_\mathrm{a}$
Equilibrium assumption:	$r_\mathrm{ads} = r_\mathrm{des}$
Mass balance	$q = S_\mathrm{a}$

$$\longrightarrow \quad q = \frac{S_0 K_{eq} p}{1 + K_{eq} p}$$

eqn. 5

$$\frac{[A_\mathrm{ad}]}{[S_0]} = \frac{\frac{k_\mathrm{ad}}{k_\mathrm{d}} p_\mathrm{A}}{1 + \frac{k_\mathrm{ad}}{k_\mathrm{d}} p_\mathrm{A}}$$

§ The manuscript we first submitted for peer review included a typo in eqn (5), with $[S_0]$ appearing as $[S]$. Neither the authors nor the peer reviewers detected this; it was identified by a community member who accessed the paper on arXiv. Of course, Lean catches such typos immediately.

Boyle's Law: Proof #1

```
import Mathlib.Data.Real.Basic

-- Variables
theorem Boyle {P1 P2 V1 V2 T1 T2 n1 n2 R : ℝ}

-- Assumptions
(h1: P1*V1 = n1*R*T1)
(h2: P2*V2 = n2*R*T2)
(h3: T1=T2)
(h4: n1=n2) :

-- Conjecture
(P1*V1 = P2*V2) :=

-- Proof
by
rw [h3] at h1
rw [h4] at h1
rw [← h2] at h1
exact h1
```

Prove that an ideal gas follows Boyle's Law

$$PV = nRT$$

$$T_1 = T_2$$

$$n_1 = n_2$$

$$P_1 V_1 = P_2 V_2$$

Boyle's Law: Proof #2

https://atomslab.github.io/LeanChemicalTheories/thermodynamics/basic.html

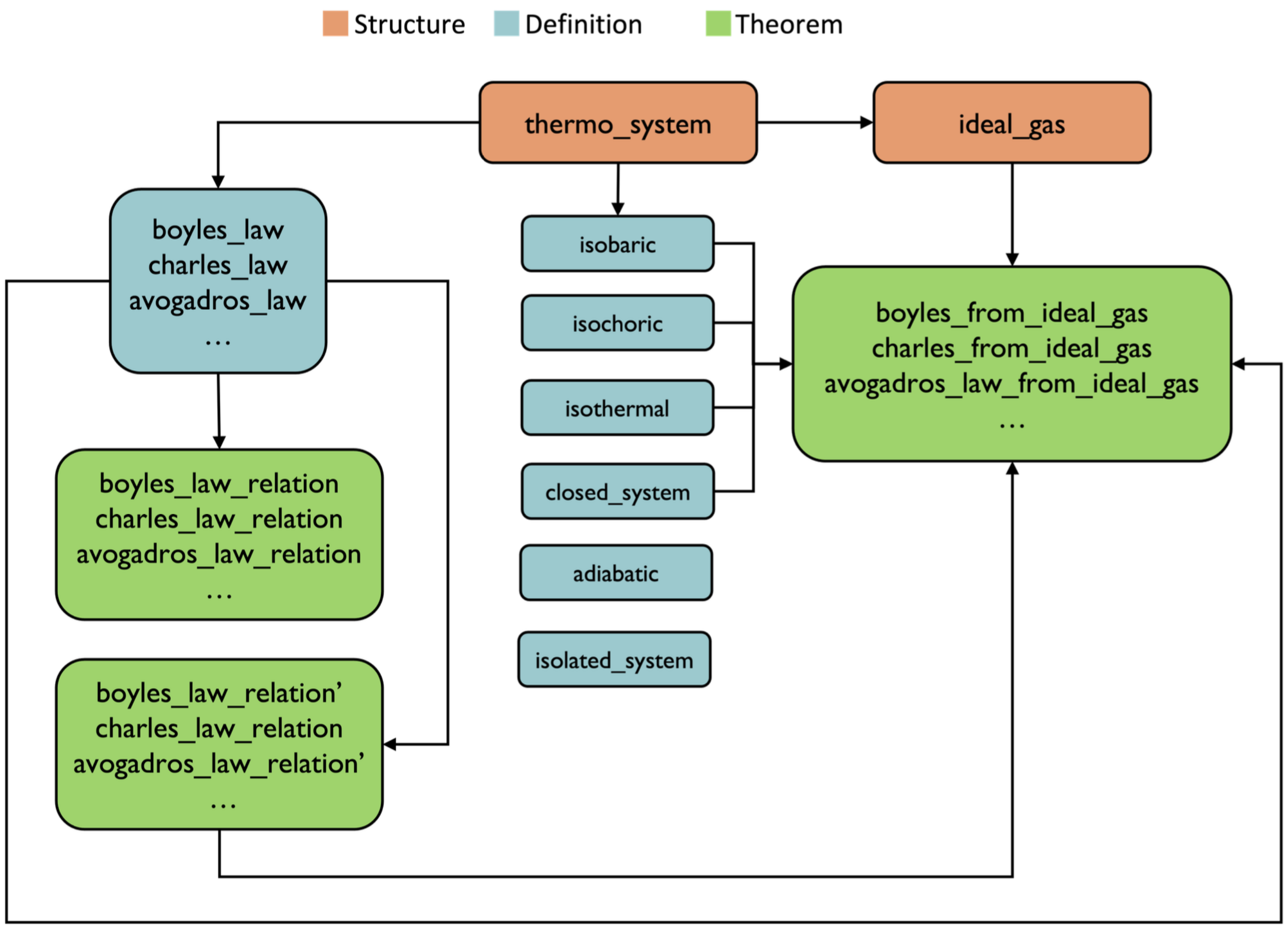
Specify concepts using *definitions* and *structures* so they can be reused in multiple proofs

Boyle's Law relation

$$P_n V_n = k$$

Boyle's Law relation'

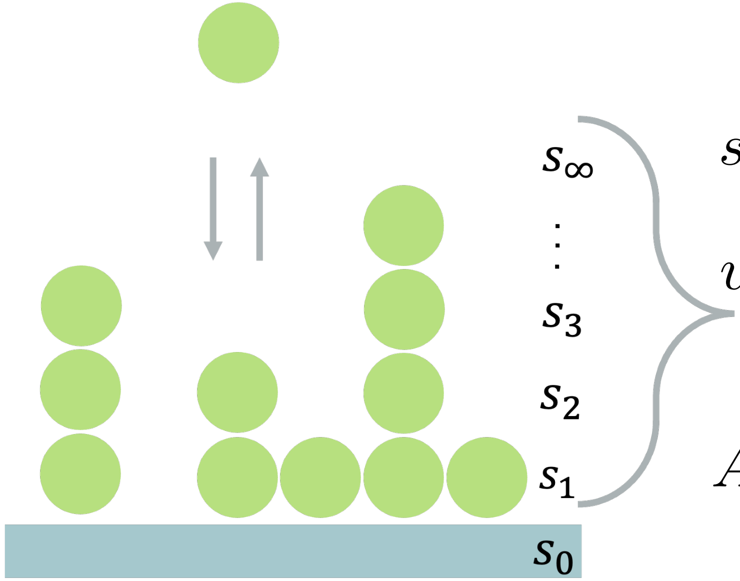
$$P_1 V_1 = P_2 V_2$$



Formalizing BET Adsorption Theory

https://atomslab.github.io/LeanChemicalTheories/adsorption/BETInfinite.html

$$v = \frac{v_m c p}{(p_0 - p)[1 + (c-1)(p/p_0)]}$$



$$s_i = C x^i s_0$$

$$v = v_0 \sum_{i=0}^{\infty} i s_i$$

$$A = v_0 \sum_{i=0}^{\infty} s_i$$

BET Adsorption

Six main premises define the model

1. Define the sequence of adsorbed layers
2. Layer 1 adsorption rate
3. Layer *n* adsorption rate
4. Total volume adsorbed v_m
5. Total area of the surface
6. Define constant *c*

Also require constraints – e.g. $p_0 > 0$

Mathlib has many useful theorems

Extra required conditions are made explicit in Lean

$$\sum_{i=1}^{\infty} x^i = \frac{x}{1-x} \qquad hx_1 : x < 1 \qquad hx_2 : x > 0$$

Minor logical correction to one step of the author's reasoning

Schedule for today

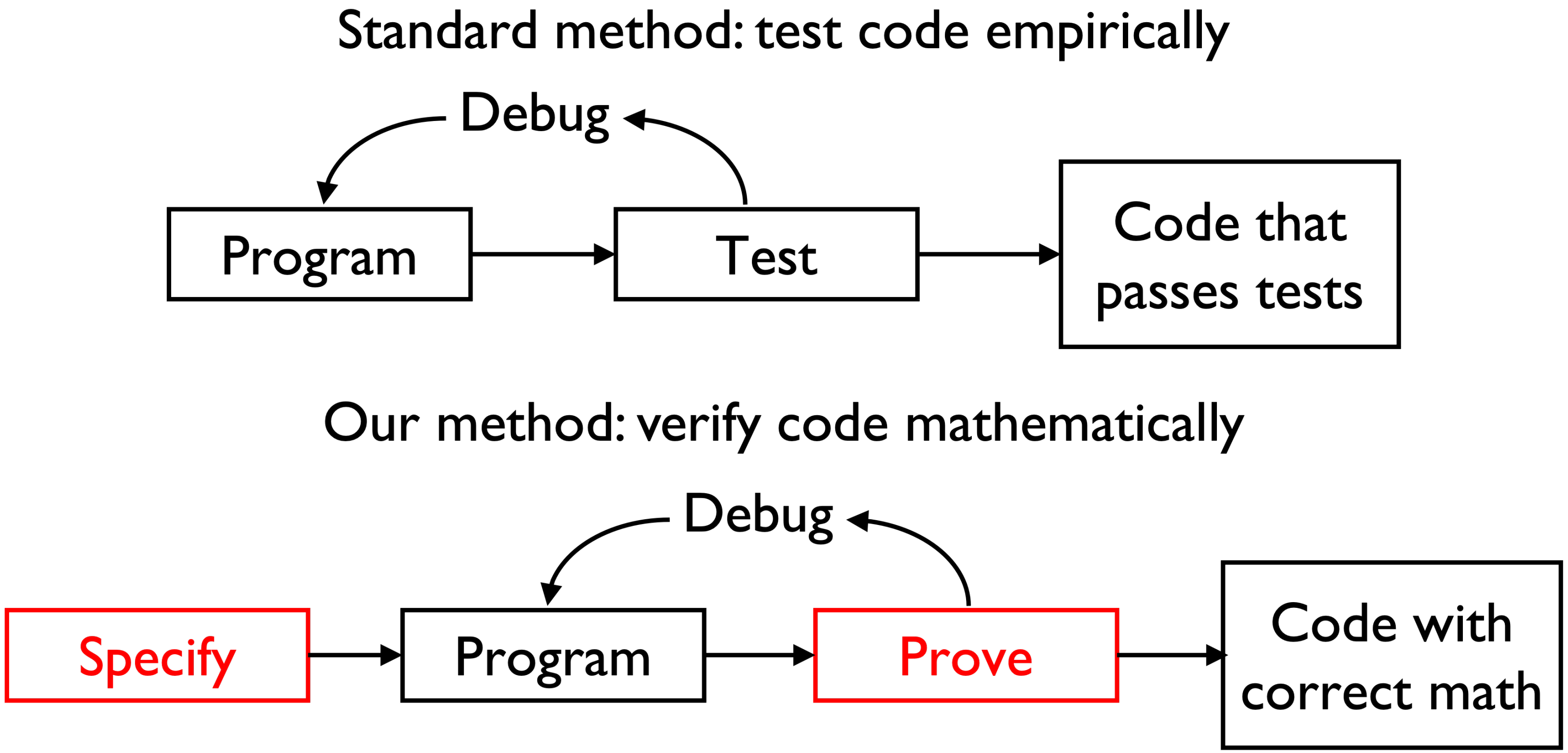
1. Provably-correct scientific computing
2. Derivations in science and engineering are math proofs
3. Formalizing mathematics with computers
4. Lean 4 and Mathlib
5. Case studies in proofs: adsorption and gas law thermodynamics
6. Case study in programming: bug-free BET analysis
7. Outlook
    1. LeanMD
    2. LLMs for theorem proving
    3. SciLib

Intermission

1. Getting connected with this course
2. Getting started with Lean
3. Proofs about equality

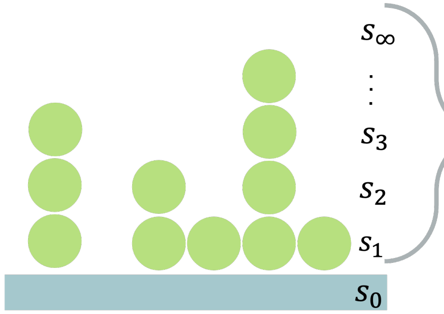
A vision for bug-free scientific computing

Selsam, Liang, Dill, "Developing Bug-Free Machine Learning Systems with Formal Mathematics," ICML 2017.



Adsorption Analysis using BET Theory

Osterrieth, et al. Adv. Mat. 2022



$$V = V_0 \sum_{i}^{\infty} i\, s_i$$

$$s_i = Cx^i s_0$$

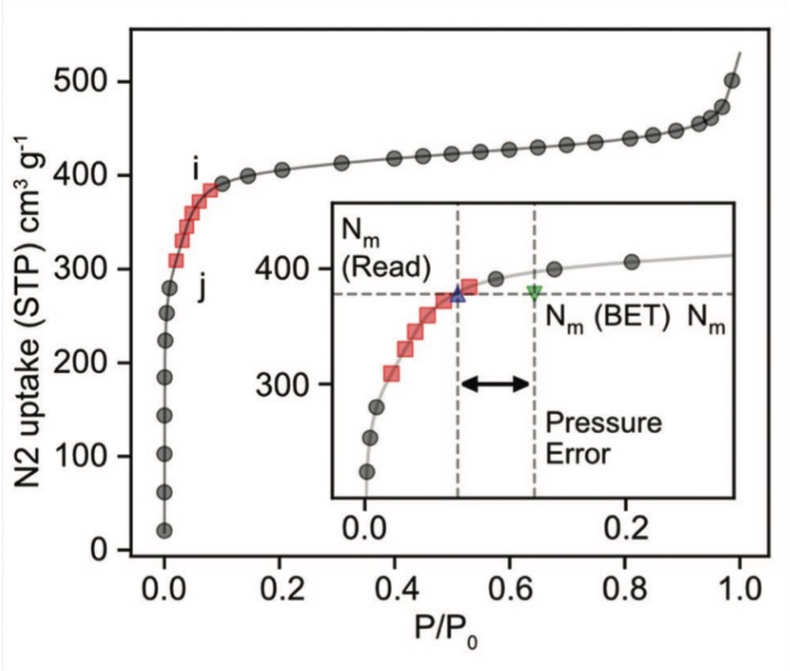
BET Adsorption

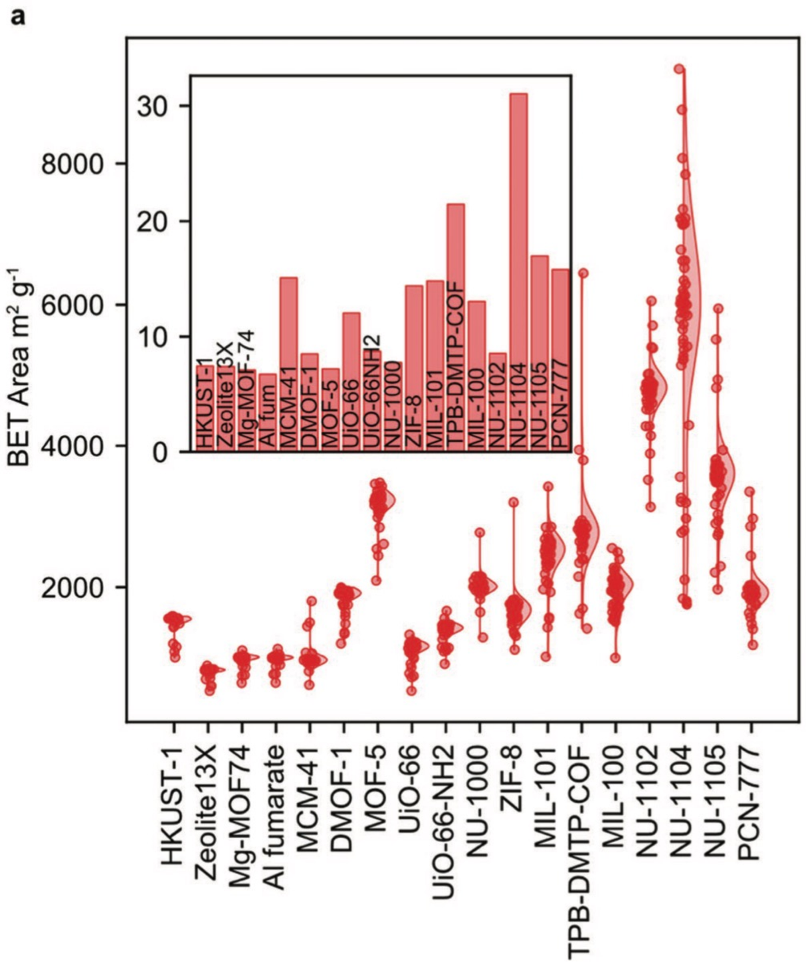
Loading = f(p)

$$q = \frac{v_m c p}{(p_0 - p)(1 + (c-1)(p/p_0))}$$

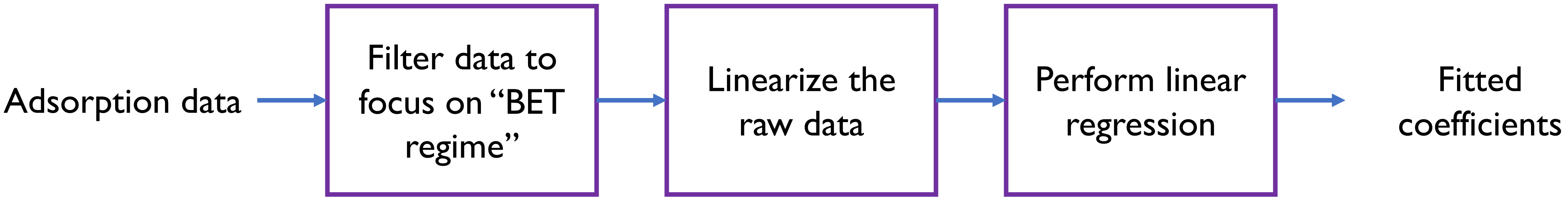
Linearized form

$$\frac{p}{q(p_0 - p)} = \frac{1}{v_m} + \frac{c-1}{v_m c}\frac{p}{p_0}$$





Bug-Free BET Analysis



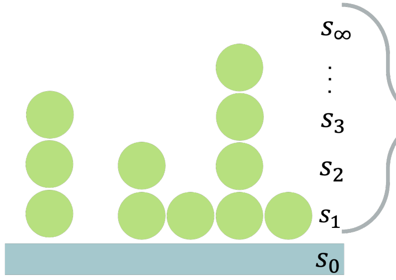
Bug-Free BET Analysis

Adsorption data → Filter data to focus on "BET regime" → Linearize the raw data → Perform linear regression → Fitted coefficients

Formal proof of BET Theory

$$q = \frac{v_m c p}{(p_0 - p)(1 + (c-1)(p/p_0))}$$

follows from a body of assumptions about



$$V = V_0 \sum_{i}^{\infty} i\, s_i$$

$$s_i = C x^i s_0$$

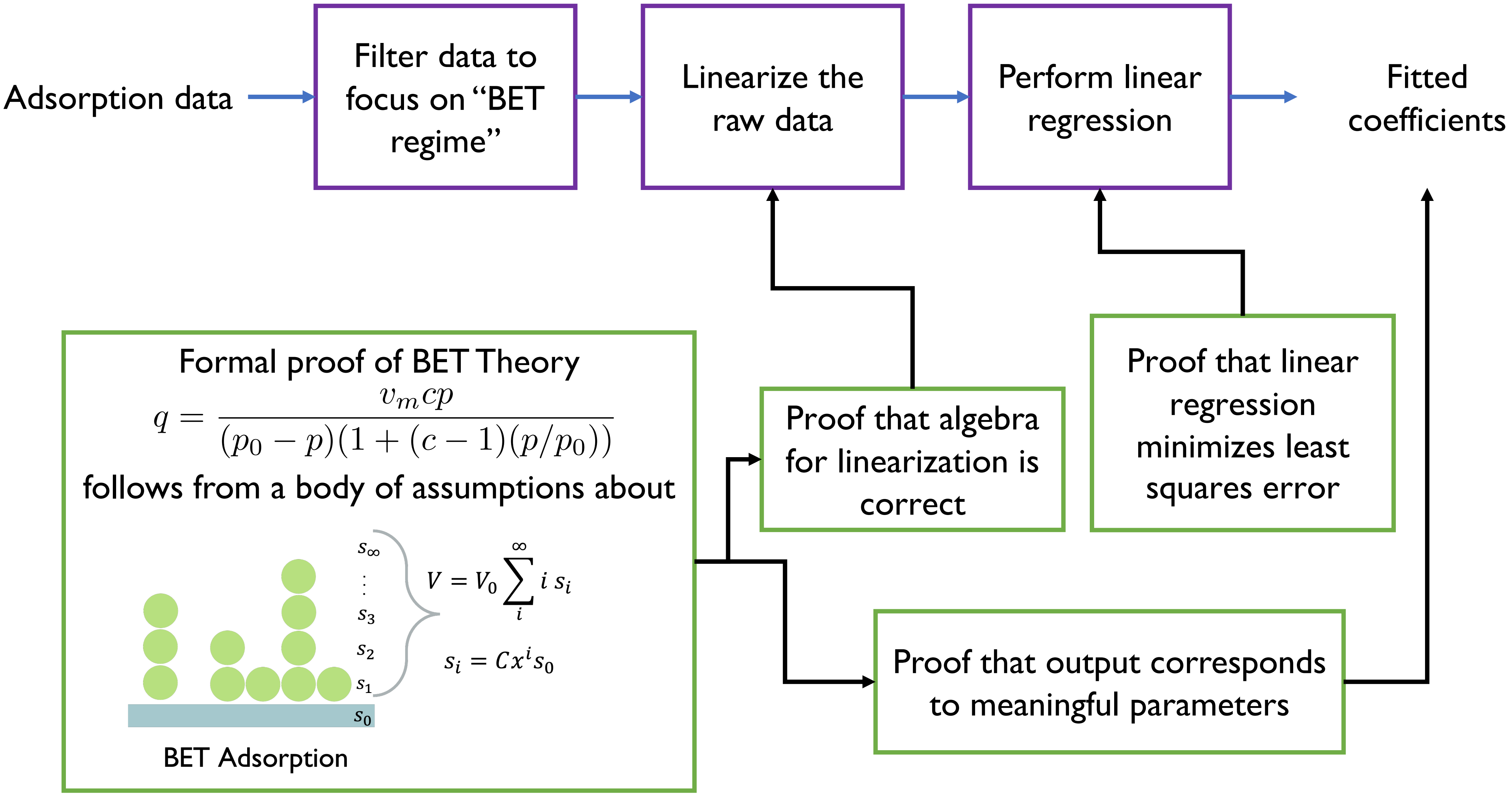
BET Adsorption

Proof that algebra for linearization is correct

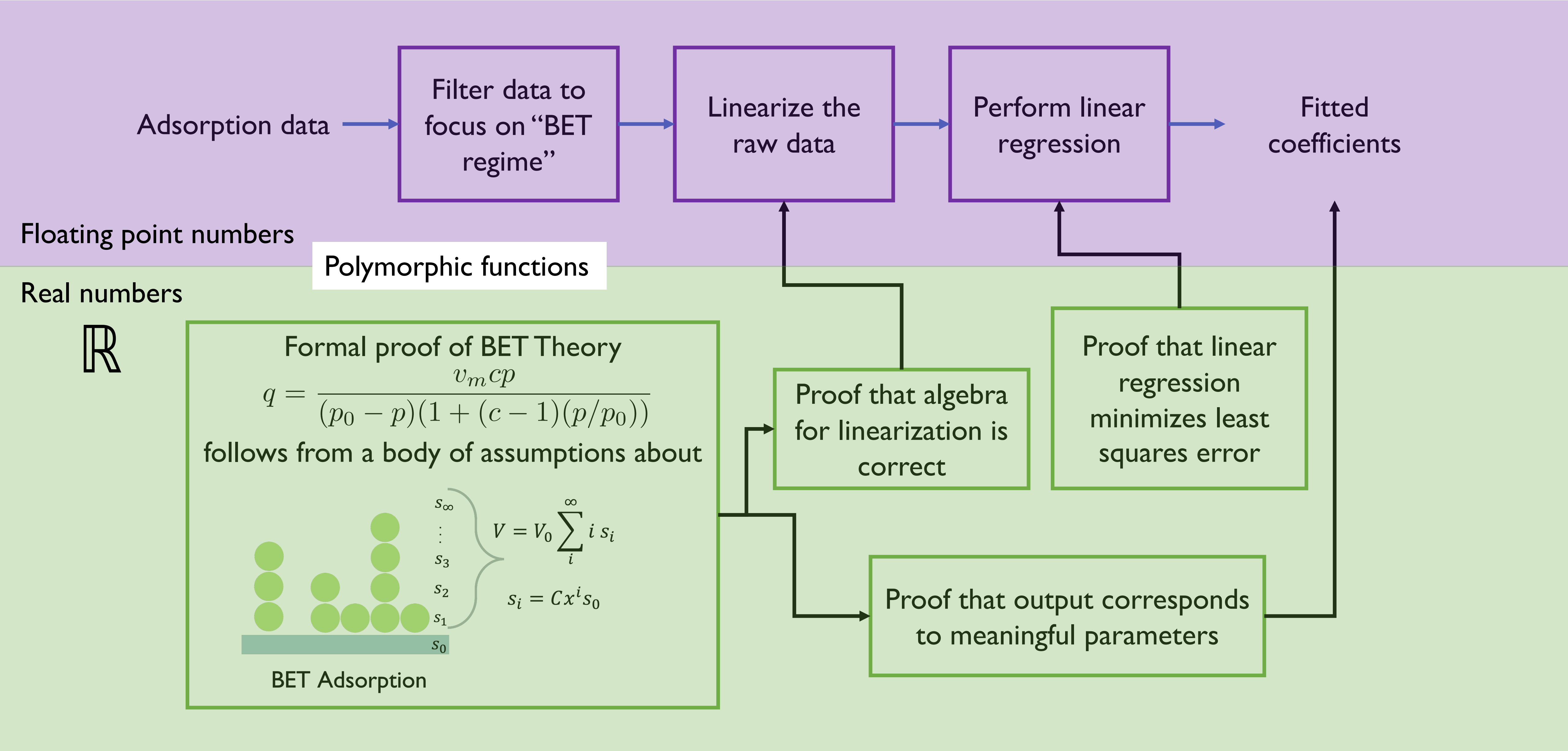
Proof that linear regression minimizes least squares error

Proof that output corresponds to meaningful parameters

Bug-Free BET Analysis

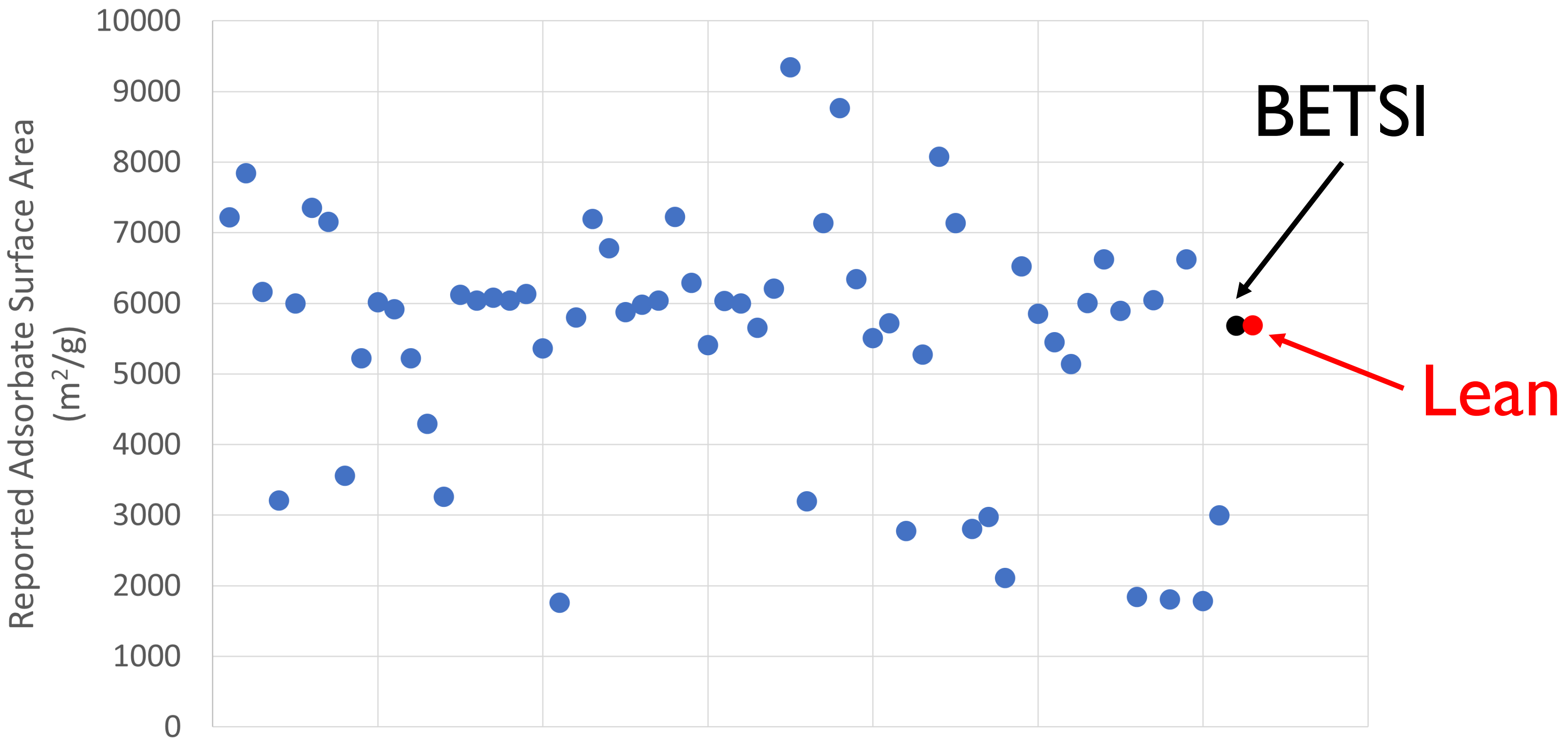


Polymorphic functions to bridge floats and reals



Regression with Lean matches BETSI standard

Osterrieth, et al. Adv. Mat. 2022



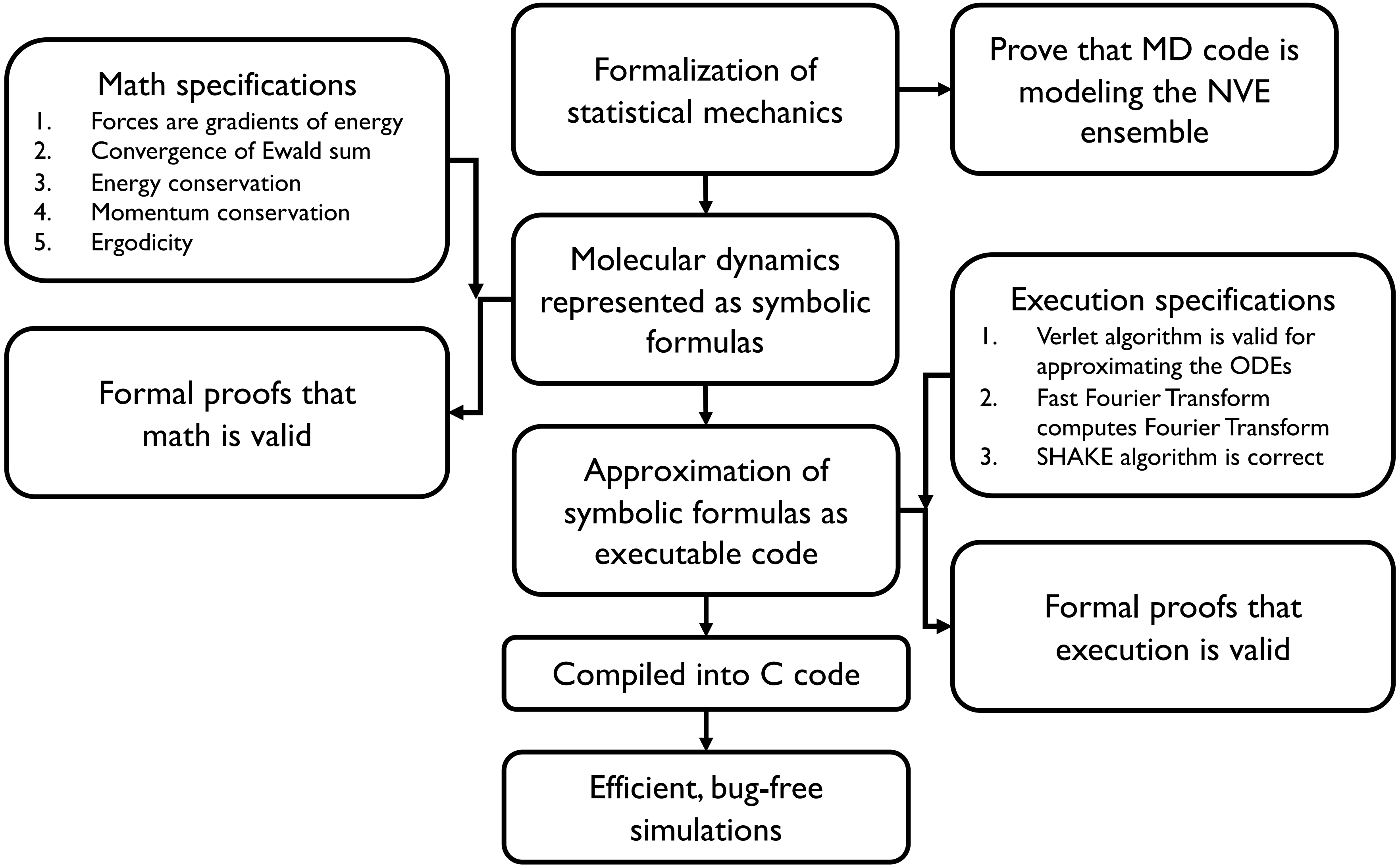
Schedule for today

1. Provably-correct scientific computing
2. Derivations in science and engineering are math proofs
3. Formalizing mathematics with computers
4. Lean 4 and Mathlib
5. Case studies in proofs: adsorption and gas law thermodynamics
6. Case study in programming: bug-free BET analysis
7. Outlook
    1. LeanMD
    2. LLMs for theorem proving
    3. SciLib

Intermission

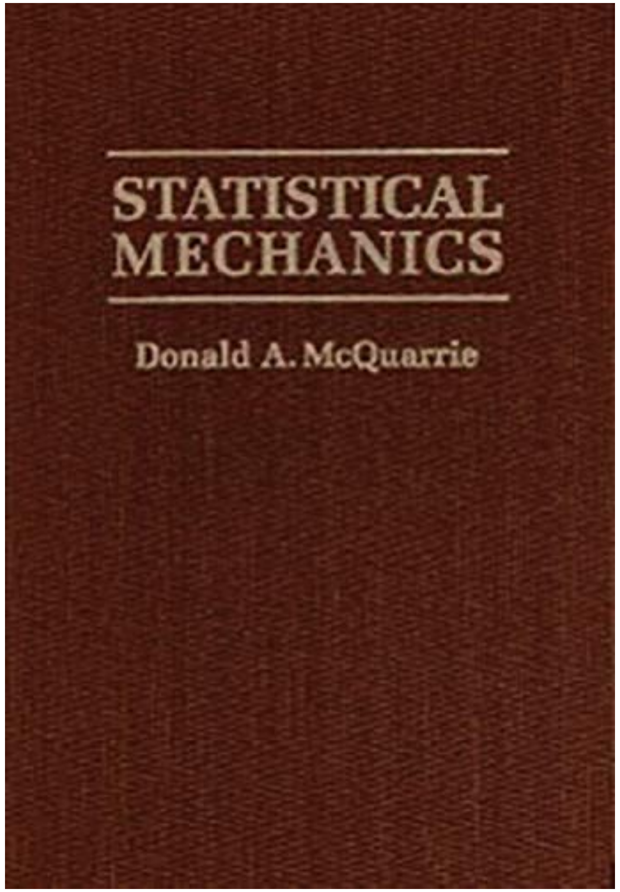
1. Getting connected with this course
2. Getting started with Lean
3. Proofs about equality

LeanMD: Formally-verified molecular dynamics



Gaps in Mathlib

- Method of Lagrange multipliers
- Maximum term method
- Much probability and statistics



But missing math can be proved and added!

Sometimes, very general math has been formalized, and specialization to useful forms is hard for non-mathematicians (e.g. partial derivatives)

"Autocomplete" Math Olympiad proofs with AI

Han, Rute, Wu, Ayers, Polu, arXiv:2102.06203, 2022
Polu, Han, Zheng, Baksys, Babuschkin, Sutskever, arXiv:2202.01344, 2022

1. Humans wrote massive proof database
2. Humans translated Math Olympiad problems into formal Lean statements
3. Train AI to predict the next word in proof
4. Execute code as Lean to verify correctness (or return errors)
5. Solve Math Olympiad problems with AI!

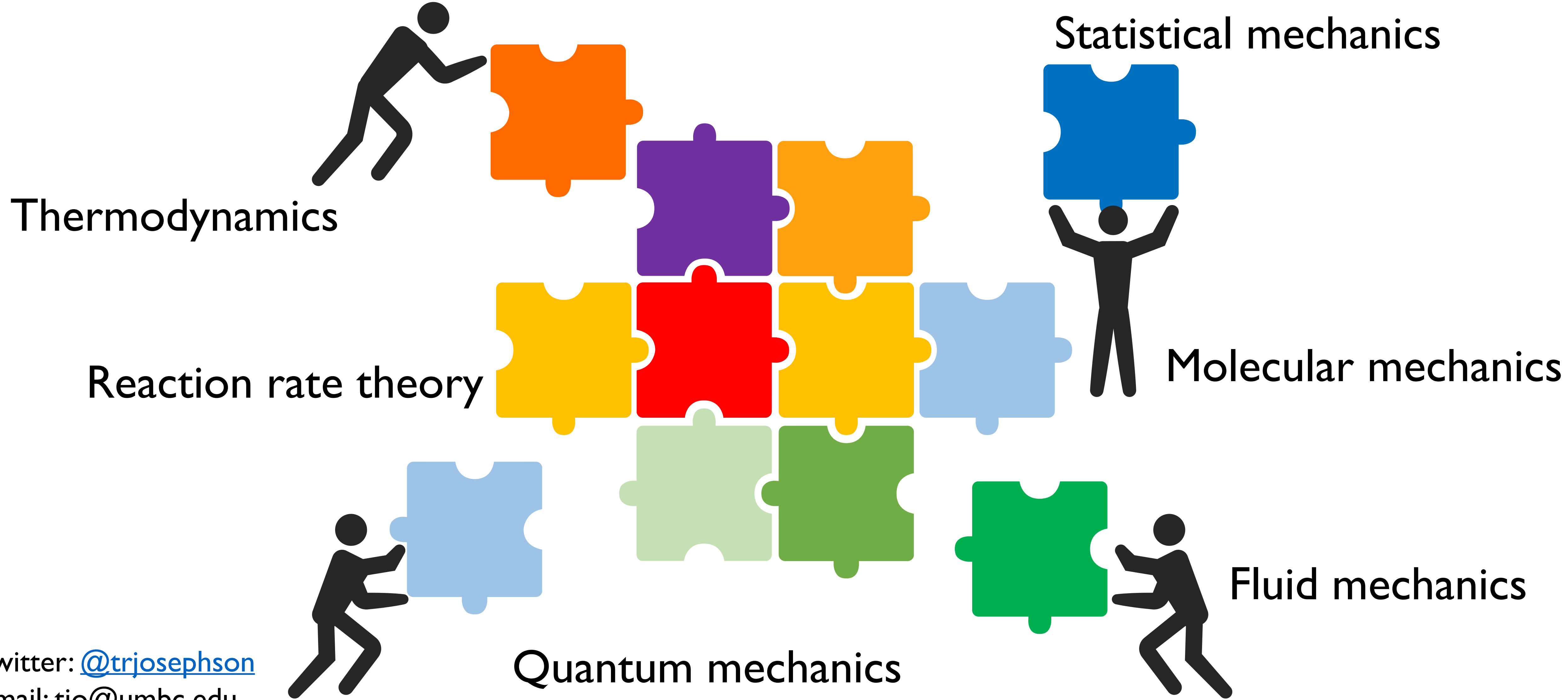
You can prompt ChatGPT to be a "Lean code assistant"

Adapted from AMC12B 2020 Problem 6

For all integers $n \geq 9$, prove that $((n+2)! - (n+1)!)/n!$ is a perfect square.

```
theorem amc12b_2020_p6
  (n : ℕ)
  (h0 : 9 ≤ n) :
  ∃ x : ℕ, (x:ℝ)^2 =
    (nat.factorial (n + 2) - nat.factorial (n + 1))
    / nat.factorial n :=
begin
  -- The model directly proposes `n + 1` as solution.
  use n + 1,
  field_simp [nat.factorial_ne_zero, pow_succ'],
  ring_exp
end
```

SciLib, database of formally verified science



Twitter: @trjosephson
Email: tjo@umbc.edu

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

Guest instructor: Tomáš Skřivan

Acknowledgements

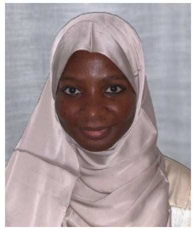
UMBC

Chemical Biochemical and Environmental Engineering

Molecular simulation and electronic structure


Samiha Sharlin

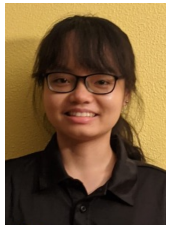

Kianoush Ramezani


Fariha Agbere

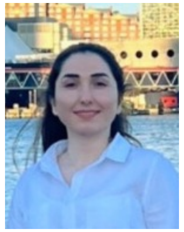
+Catherine Wraback, Bruke Hirgeto, Brayden Gruzs

Knowledge representation


Sharon Liu


An Hong Dang

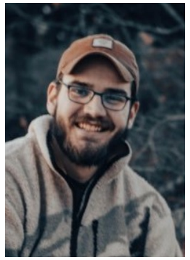
Theorem proving


Parivash Feyzishendi

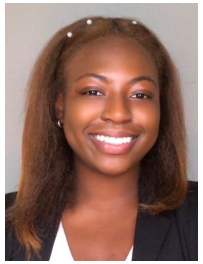

Ejike (David) Ugwuanyi


Max Bobbin


Jinyu Huang


John Velkey

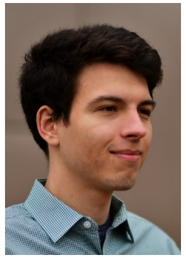

Colin Jones

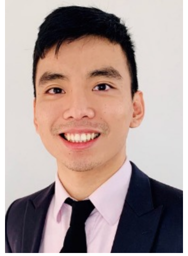

Shashane Anderson


Oscar Matemb

Not pictured:
Sophia Hamer
Rodrigo Lozano
Adhithi Varadarajan
Hanifah Shoneye
Ami Ashman
Timothy Cai
Charishma Puli
Joshua Davis-Carpenter
Kevin Ishimwe
Alan Vithayathil

Symbolic regression


Charlie Fox


Neil Tran


Nikki Nacion


Charishma Puli

Carnegie Mellon University

Jeremy Avigad
Tomáš Skřivan

Microsoft

Leonardo de Moura

IBM

Jason Rute

NSF

This is based upon work supported by NSF under ERI grant #2138938, CAREER grant #2236769, and UMBC startup funds.

Schedule for today

1. Provably-correct scientific computing
2. Derivations in science and engineering are math proofs
3. Formalizing mathematics with computers
4. Lean 4 and Mathlib
5. Case studies in proofs: adsorption and gas law thermodynamics
6. Case study in programming: bug-free BET analysis
7. Outlook
   1. LeanMD
   2. LLMs for theorem proving
   3. SciLib

Intermission

1. Getting connected with this course
2. Getting started with Lean
3. Proofs about equality

Sirena
BB Yukus & Electric Dad

Schedule for today

1. Provably-correct scientific computing
2. Derivations in science and engineering are math proofs
3. Formalizing mathematics with computers
4. Lean 4 and Mathlib
5. Case studies in proofs: adsorption and gas law thermodynamics
6. Case study in programming: bug-free BET analysis
7. Outlook
    1. LeanMD
    2. LLMs for theorem proving
    3. SciLib

Intermission

1. Getting connected with this course
2. Getting started with Lean
3. Proofs about equality

Who's registered for LfSE?

Attending?

17 plan to attend in person

243 plan to attend online

111 just want the videos

Career stage?

27% undergraduate students

36% graduate students

29% working outside academia

Field of study?

26% engineering

13% physical science

54% computer science

32% mathematics

14% scientific computing

Math?

79% taken / taking science core

29% independently study logic

33% taken course in logic

29% math major

Coding?

12% new to coding

10% write standalone scripts

22% comfortable writing functions

54% contributed to a collaborative software project

Lean?

27% never heard of Lean before

40% heard of Lean, wanted to try

19% tried Lean once or twice

12% basics in proofs or programs

3% fairly proficient

Getting connected to this course

Zulip Online Forum

Chat forum (all links are here)
https://leanprover.zulipchat.com/#narrow/stream/445230-Lean-for-Scientists-and-Engineers-2024

Lean files – I'm working on getting this organized. I'd love for future classes to be organized around an online textbook, written in and validated by Lean. For now, they'll be posted on Zulip prior to class.

Schedule
https://docs.google.com/spreadsheets/d/1ATL-RngI3IGM6uM1ZkXxQdZzYLOAxSn5ZN0MBrfq--o/edit?gid=2038742424#gid=2038742424

Getting started with Lean

- Instructions for installing Lean locally
  - https://lean-lang.org/lean4/doc/quickstart.html
  - Usually, you want to install with Mathlib
  - If you have problems, ask for help on Zulip!

- Run Lean in a browser
  - https://live.lean-lang.org/
  - http://lean.math.hhu.de/
  - Practice Lean in a pinch if local installation fails
  - Show Lean to newcomers (Zulip lets you launch any snippet of Lean code in the browser)

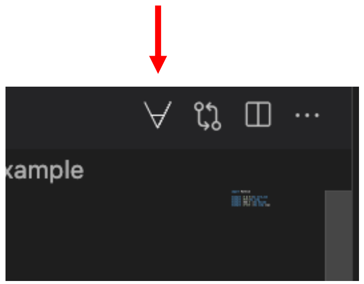
Most important VS Code tip

In VS Code, hover your mouse over symbols and variables to get information about types, order of operations, documentation on tactics, definitions of theorems, and links to more information

Another tip

If you lose your infoview in VS Code, don't panic! You can get it back by clicking on the ∀ symbol along the tabs, then "toggle infoview"

Or, use the shortcut "shift-⌘-enter"



Proofs about equality

Additional reference: Mechanics of Proof, Chapters 1.1 and 1.2

"Calculational"-style proofs

"We solve problems which feel pretty close to high school algebra – deducing equalities/inequalities from other equalities/inequalities – using a technique which is not usually taught in high school algebra: building a single chain of expressions connecting the left-hand side with the right."

– Heather Macbeth, Mechanics of Proof

A guide to number systems

$\mathbb{N}$ - Natural numbers (0, 1, 2, 3, 4, …)

$\mathbb{Z}$ - Integers (… -3, -2, -1, 0, 1, 2, …)

$\mathbb{Q}$ - Rational numbers (1/2, 3/4, 5/9, etc.)

$\mathbb{R}$ - Real numbers (-1, 3.6, π, √2)

$\mathbb{C}$ - Complex numbers (-1, 5 + 2i, √2 + 5i, etc.)

Thanks to Bethany M on YouTube:
https://www.youtube.com/watch?v=CwY62w5tfqM

First example:

Note Title 5/17/2011

43. Science and Medicine A light plane flies 450 mi with the wind in 3 h. Flying back against the wind, the plane takes 5 h to make the trip. What was the rate of the plane in still air? What was the rate of the wind?

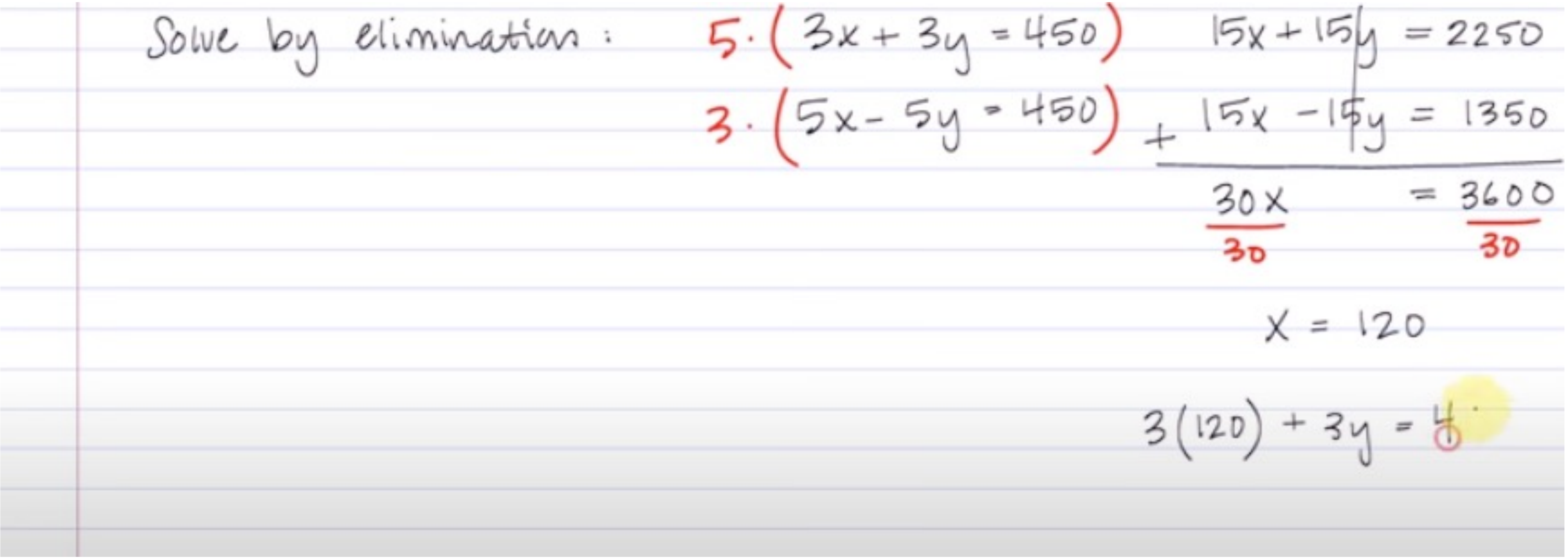
→ x = Speed (rate) of the plane in still air
→ y = Speed (rate) of the wind

	d =	r ·	t
with wind	450	$x+y$	3
against wind	450	$x-y$	5

→ $450 = (x+y)3$ $\qquad 3x+3y=450$

→ $450 = (x-y)5$ $\qquad 5x-5y=450$

Proof by elimination: BAD high school algebra technique



Go to Lean file for rigorous proof

Lean is not (yet) a computer algebra system

Theorem Provers

Do *proofs*
Symbolically transform formulae
Only perform correct transformations
Built off a small, trusted kernel

Computer Algebra Systems

Do *calculations*
Symbolically transform formulae
Human-checked correctness
Large program with many algorithms

LEAN THEOREM PROVER
Isabelle

Mathematica
SymPy
MATLAB

Theorem provers aren’t built to “solve for x”