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

Lemonade
PIKASONIC

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. Survey for attendees
2. Recap Lecture 2
    1. Revisit syntax vs. semantics
3. Proofs with intermediate steps
4. Proofs using lemmas from Mathlib
5. Junk values, and why 1/0 = 0
6. Logical operators
7. Proofs with AND and OR

Survey for attendees

https://forms.gle/pg5JGpTgD1aSCshY6

Poll questions:

How many hours did you spend with Lean last week (including time in class / listening to recordings?)

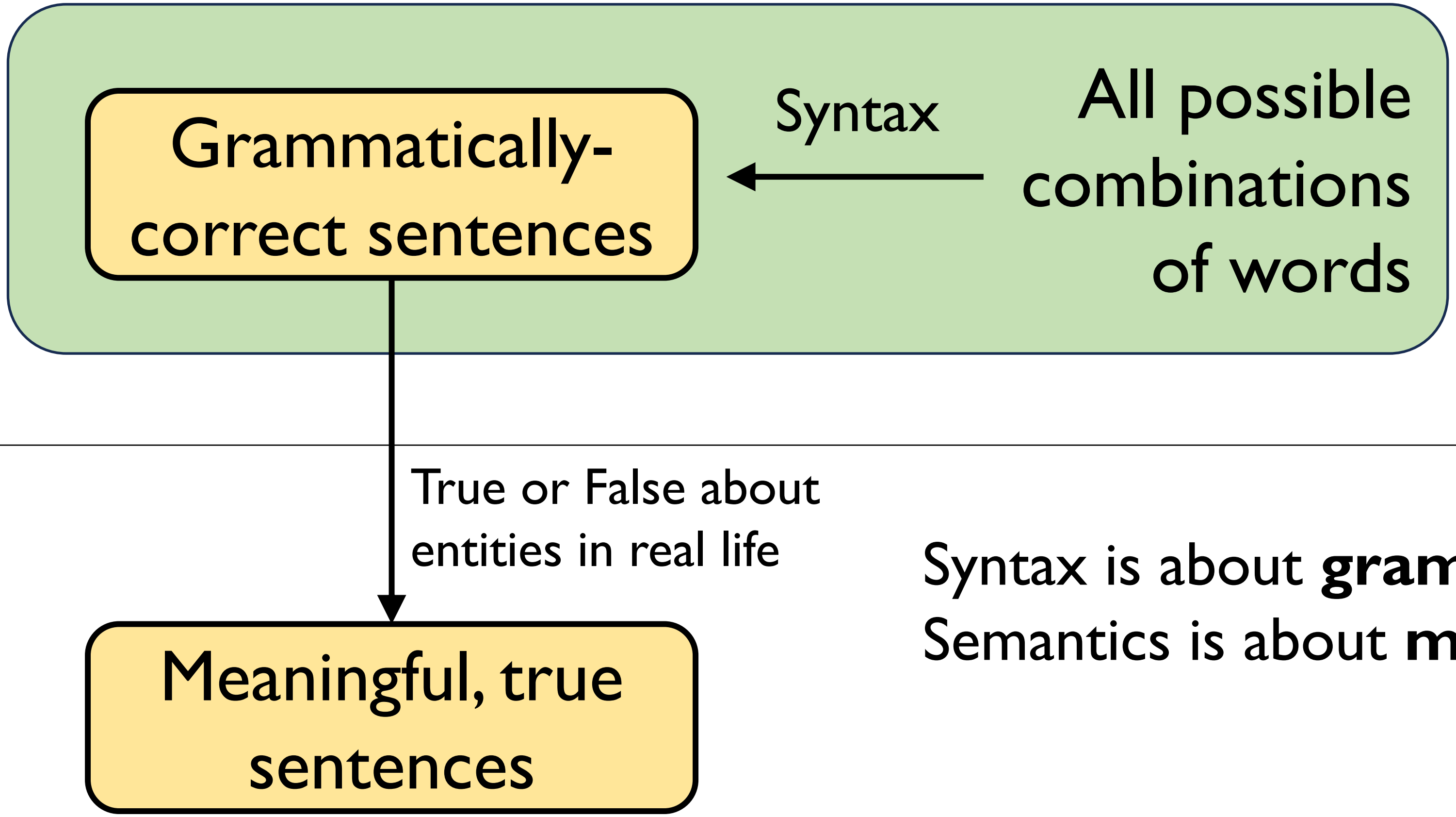
Did you explore more Mechanics of Proof exercises from Chapter 1?

Slide from Lecture 2

Syntax vs. semantics in *natural* language

Language

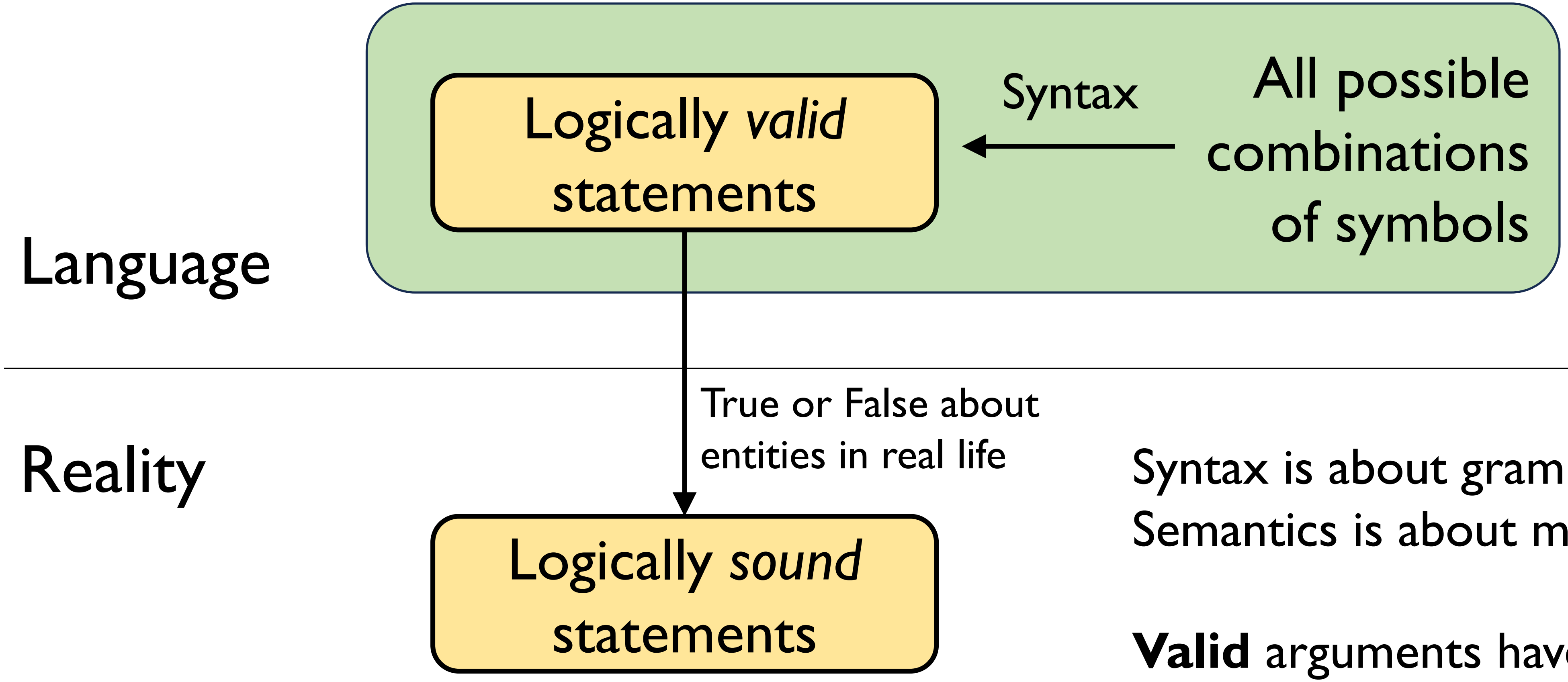
Reality



Syntax is about **grammar**
Semantics is about **meaning**

Slide from Lecture 2

Syntax vs. semantics in *logic*

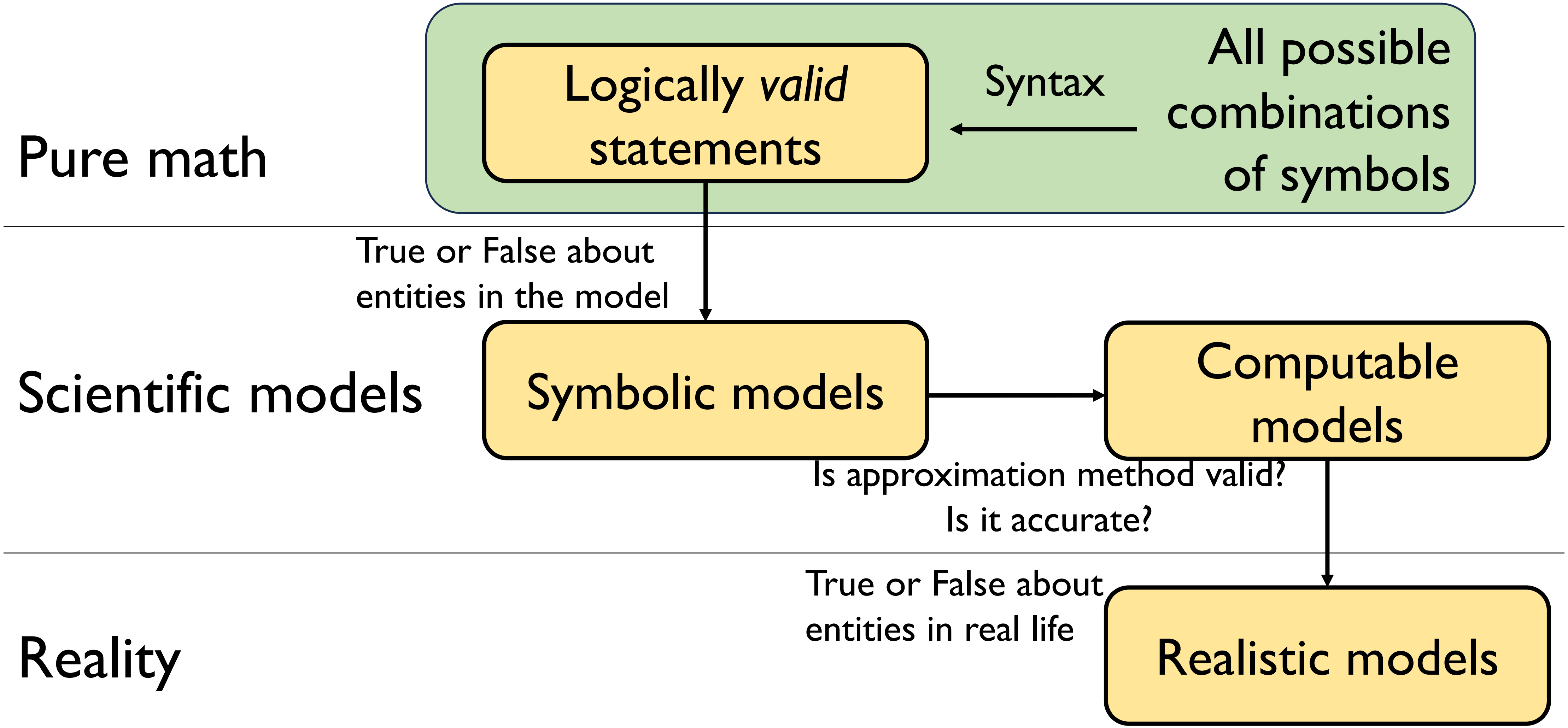


Syntax is about grammar
Semantics is about meaning

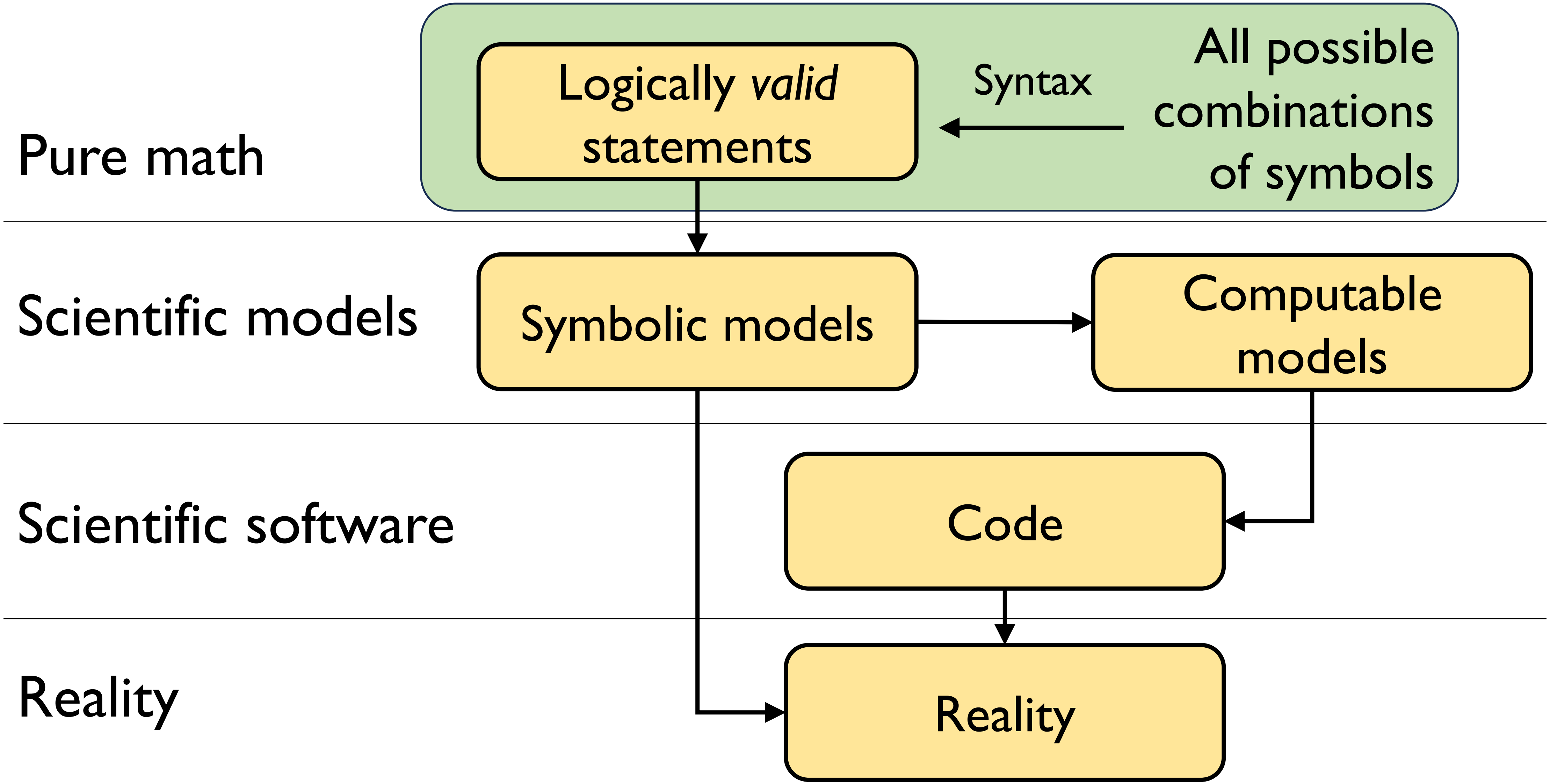
Valid arguments have correct logic
Sound arguments are valid and also have true premises

Slide from Lecture 2

Semantic errors in scientific computing

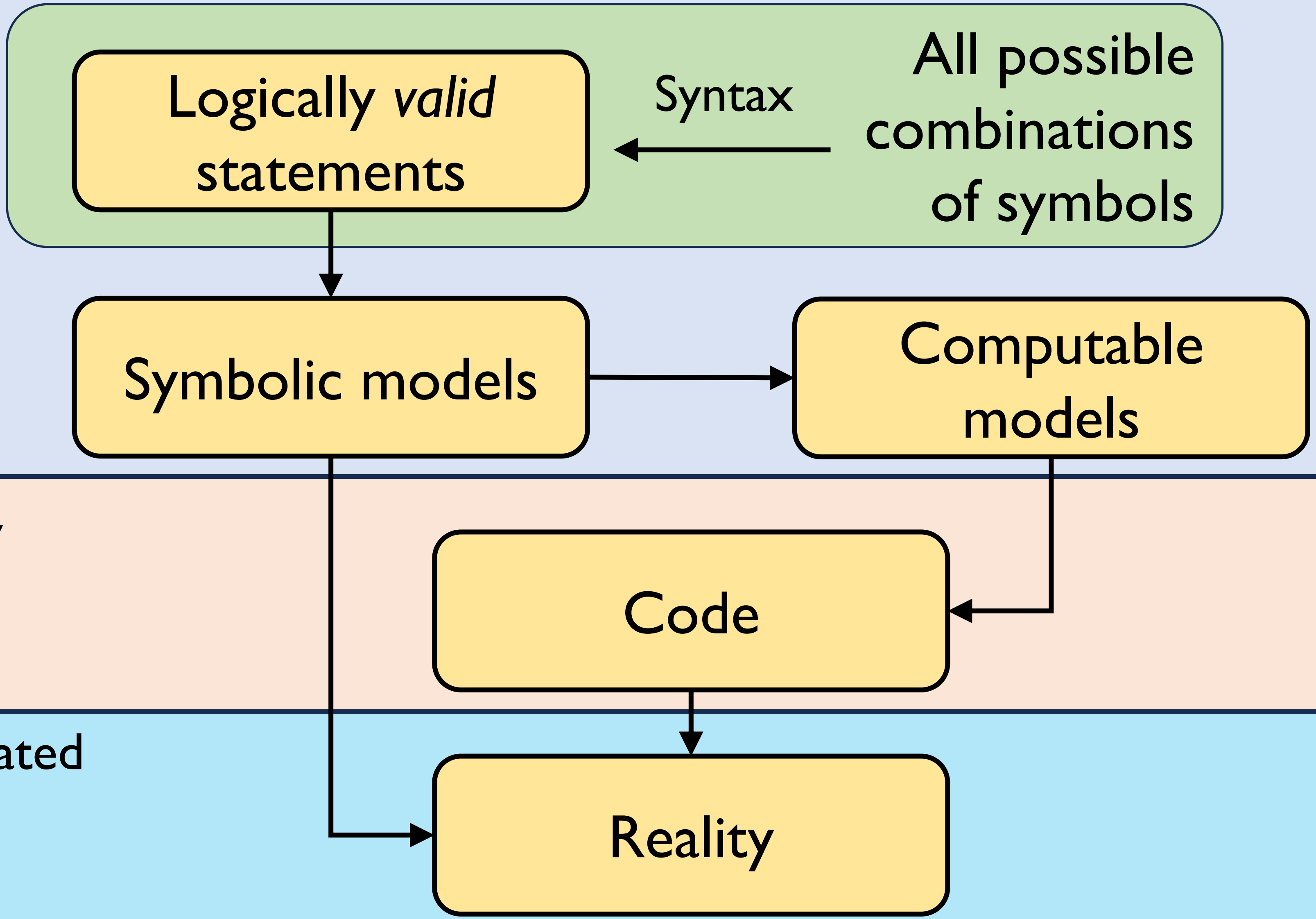


Syntax and semantics in scientific computing



Syntax and semantics in scientific computing

Traditionally, the validity of the mathematics and the scientific theory are established **by hand**



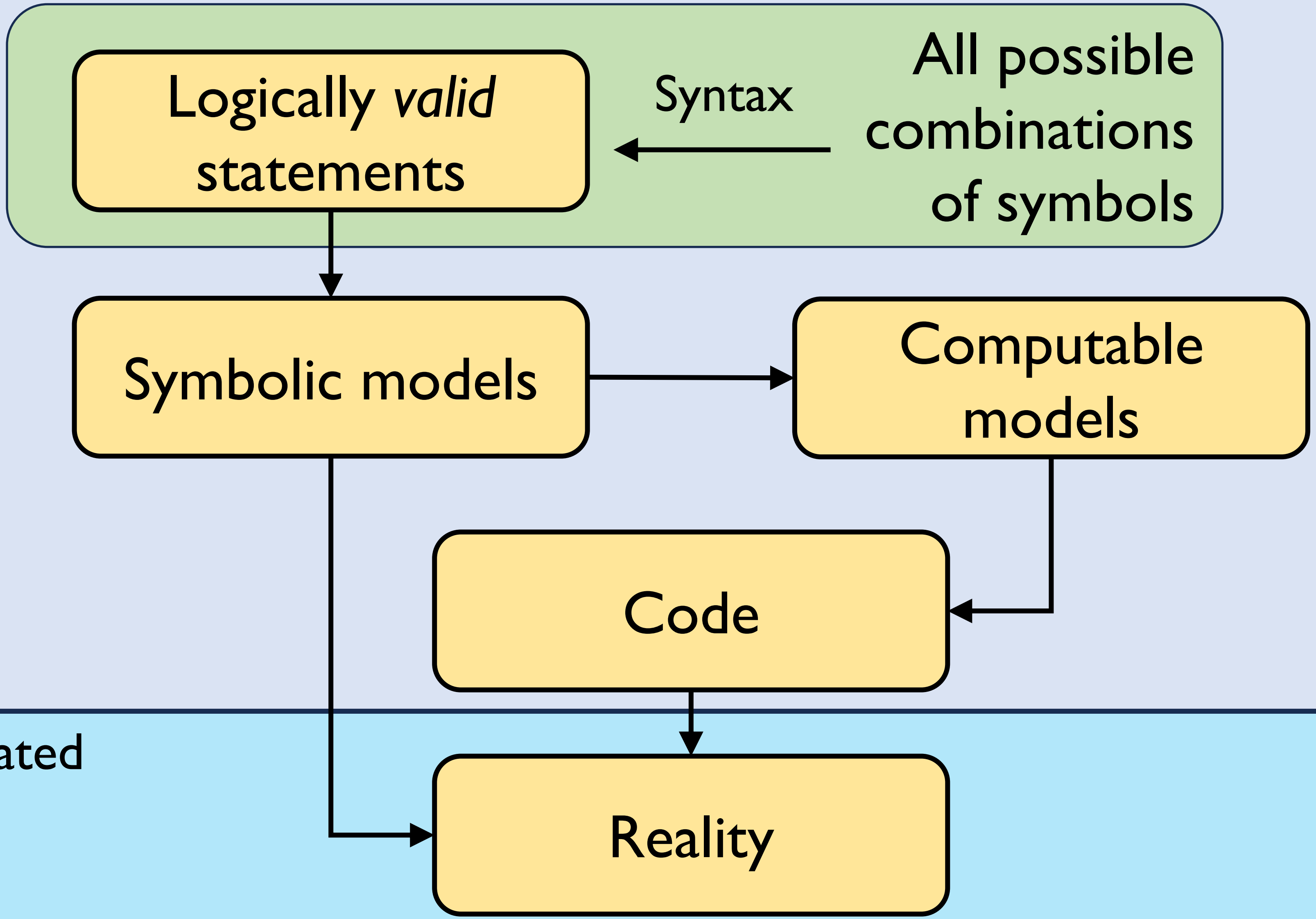
Humans read the theory and write the code as best as they can

Then use various automated and manual means to compare to experiment

Syntax and semantics in scientific computing

Can we represent all of this in Lean, and validate the construction of the math, scientific models, and software, in one system?

Then use various automated and manual means to compare to experiment



Proofs using intermediate steps

- Sometimes, it's helpful to prove a little thing that helps you prove the main thing
- At scale, this is how Mathlib works, as an interconnected web of proofs
- Can also internally define a statement and prove it
- https://github.com/ATOMSLab/LeanChemicalTheories/blob/kepler'sLaw/src/physics/kepler'sLaw

Should you use have or add a hypothesis?

Using have

```
example {a b : ℝ}
  (h1 : a - 5 * b = 4)
  (h2 : b + 2 = 3) :
  a = 9 := by
  have hb : b = 1 := by linarith
  calc
    a = a - 5 * b + 5 * b := by ring
    _ = 4 + 5 * 1 := by rw [h1, hb]
    _ = 9 := by ring
```

New hypothesis

```
example {a b : ℝ}
  (h1 : a - 5 * b = 4)
  (h2 : b + 2 = 3)
  (hb : b = 1):
  a = 9 := by
  calc
    a = a - 5 * b + 5 * b := by ring
    _ = 4 + 5 * 1 := by rw [h1, hb]
    _ = 9 := by ring
```

Should you use have or add a hypothesis?

Using have

```
example {a b : ℝ}
  (h1 : a - 5 * b = 4)
  (h2 : b + 2 = 3) :
  a = 9 := by
  have hb : b = 1 := by linarith
  calc
    a = a - 5 * b + 5 * b := by ring
    _ = 4 + 5 * 1 := by rw [h1, hb]
    _ = 9 := by ring
```

New hypothesis

```
example {a b : ℝ}
  (h1 : a - 5 * b = 4)
  (h2 : b + 2 = 3)
  (hb : b = 1):
  a = 9 := by
  calc
    a = a - 5 * b + 5 * b := by ring
    _ = 4 + 5 * 1 := by rw [h1, hb]
    _ = 9 := by ring
```

We've changed the theorem statement.
h2 is "unused"
We don't know if hb is true!
If hb contradicts any other hypotheses, we're in real trouble

Principle of logical explosion

- You MUST NOT assume a set of premises with a contradiction
- "Principle of explosion"
- https://en.wikipedia.org/wiki/Principle_of_explosion
- Also known as "proving false"
- You can prove anything, which isn't actually helpful
- Lean has tactic "slim_check" that can sometimes detect this by searching for counterexamples
  - Examples here: https://github.com/leanprover-community/mathlib4/blob/master/test/slim_check.lean

Proofs using existing theorems

- apply tactic directly updates the goal using a theorem

- Some tactics are aware of a bunch of theorems already
- Other tactics can be "told about" theorems to make them smarter

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)

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)

Glossary of logical symbols

$\wedge$ - and

$\vee$ - or

$\neg$ - not

$\rightarrow$ - implies

$\leftrightarrow$ - if and only if (implies in both directions)

$\exists$ - exists

$\forall$ - for all

$\wedge$: and

P: molecule is aromatic

Q: molecule is an alcohol

$P \wedge Q$: molecule is aromatic and an alcohol

P: true, Q: true – then $P \wedge Q$: true

P: false, Q: true – then $P \wedge Q$: false

P: true, Q: false – then $P \wedge Q$: false

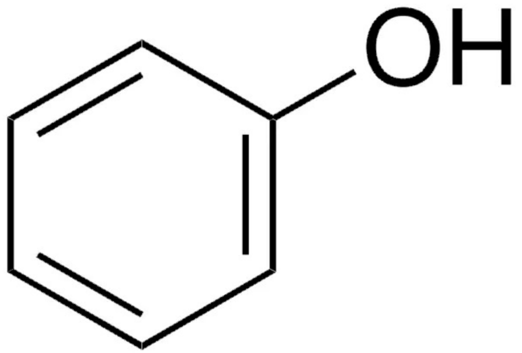
P: false, Q: false – then $P \wedge Q$: false

∧ : and

P: molecule is aromatic

Q: molecule is an alcohol

P ∧ Q: molecule is aromatic and an alcohol



Phenol

P: true, Q: true – then P ∧ Q: true

P: false, Q: true – then P ∧ Q: false

P: true, Q: false – then P ∧ Q: false

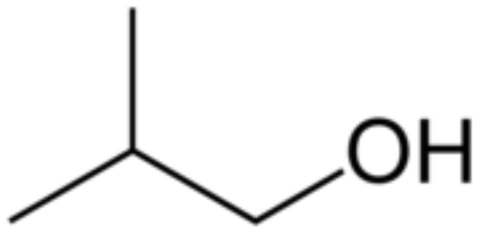
P: false, Q: false – then P ∧ Q: false

∧ : and

P: molecule is aromatic

Q: molecule is an alcohol

P ∧ Q: molecule is aromatic and an alcohol



isobutanol

P: true, Q: true – then P ∧ Q: true

P: false, Q: true – then P ∧ Q: false

P: true, Q: false – then P ∧ Q: false

P: false, Q: false – then P ∧ Q: false

∧ : and

P: molecule is aromatic

Q: molecule is an alcohol

P ∧ Q: molecule is aromatic and an alcohol

P: true, Q: true – then P ∧ Q: true

P: false, Q: true – then P ∧ Q: false

P: true, Q: false – then P ∧ Q: false

P: false, Q: false – then P ∧ Q: false

P	Q	(P ∧ Q)
true	true	true
false	true	false
true	false	false
false	false	false

∨ : or

P: contains acrolein

Q: contains hydrogen cyanide

P ∨ Q: acute toxicity

P: true, Q: true – then P ∨ Q: true

P: false, Q: true – then P ∨ Q: true

P: true, Q: false – then P ∨ Q: true

P: false, Q: false – then P ∨ Q: false

P	Q	(P v Q)
true	true	true
false	true	true
true	false	true
false	false	false