Emily Riehl

Johns Hopkins University

Can machines think logically?

DSAI and KSAS Symposium Series: The Science of AI

Can machines think?

In a famous 1950 paper entitled "Computing machinery and intelligence," British mathematician Alan Turing opens with the question

"Can machines think?"

Sidestepping the task of giving precise meaning to the term "think," Turing instead describes what he calls "the imitation game" and asks whether an interrogator in another room would be able to distinguish between a man and machine impersonating a man by asking a series of typewritten questions.[a]

[a]Actually, the imitation game is much queerer than this: Turing asks whether a machine would do better than a man at impersonating a woman.

A. M. Turing (1950) Computing Machinery and Intelligence. ***Mind 49:*** **433-460.**

COMPUTING MACHINERY AND INTELLIGENCE

By A. M. Turing

1. The Imitation Game

The Turing test

Turing predicted that in 50 years time, machines would be able to pass what is now known as the Turing test.

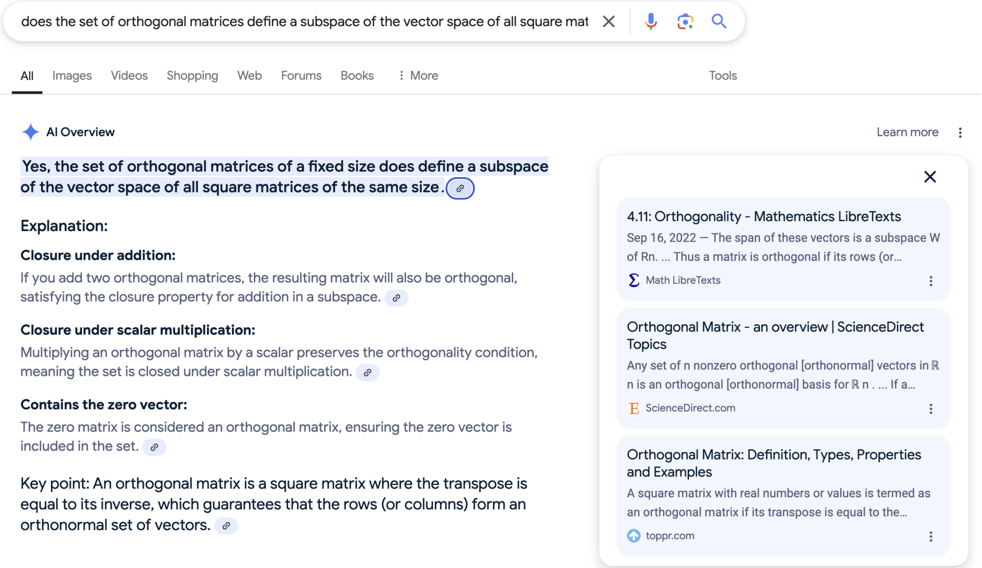
This is unequivocally the case for today's large language models.

But I want to ask a different question:

"Can machines think logically?"

In realms, such as mathematics, where logical reasoning can distinguish what is true from what is false, can generative AI reliably give us the correct answer?

A question from my practice final in Math 201: Linear Algebra



Do orthogonal matrices form a subspace?

My student asked Gemini:

"Does the set of orthogonal matrices define a subspace of the vector space of all square matrices?"

In the 2×2 case, orthogonal matrices have the form

$$\begin{bmatrix} s & -t \\ t & s \end{bmatrix} \quad \text{or} \quad \begin{bmatrix} s & t \\ t & -s \end{bmatrix} \quad \text{with} \quad s^2 + t^2 = 1,$$

while square matrices have the form $\begin{bmatrix} a & b \\ c & d \end{bmatrix}$ for any real numbers a, b, c, and d.

The set of 2×2 matrices forms a vector space because you can add them and multiply them by a real number to get another 2×2 matrix.

This question asks whether the sum or scalar multiple of orthogonal matrices is again orthogonal and whether the zero matrix is an orthogonal matrix.

Gemini's illogical "thinking"

"Does the set of orthogonal matrices define a subspace of the vector space of all square matrices?"

Gemini answered "Yes" with the following explanation:

- "If you add two orthogonal matrices, the resulting matrix will also be orthogonal, satisfying the closure property for addition in a subspace."

  This is incorrect.

- "Multiplying an orthogonal matrix by a scalar preserves the orthogonality condition, meaning the set is closed under scalar multiplication."

  This is incorrect.

- "The zero matrix is considered an orthogonal matrix, ensuring the zero vector is included in the set."

  This is incorrect.

Despite the fact that Gemini gave correct definitions of subspace and of orthogonal matrices (as those matrices whose transpose is the inverse), it answered wrong.

Gemini's illogical "thinking"

does the set of orthogonal matrices define a subspace of the vector space of all square mat

All Images Videos Shopping Web Forums Books ⋮ More Tools

AI Overview

Learn more

Yes, the set of orthogonal matrices of a fixed size does define a subspace of the vector space of all square matrices of the same size.

Explanation:

Closure under addition:

If you add two orthogonal matrices, the resulting matrix will also be orthogonal, satisfying the closure property for addition in a subspace.

Closure under scalar multiplication:

Multiplying an orthogonal matrix by a scalar preserves the orthogonality condition, meaning the set is closed under scalar multiplication.

Contains the zero vector:

The zero matrix is considered an orthogonal matrix, ensuring the zero vector is included in the set.

Key point: An orthogonal matrix is a square matrix where the transpose is equal to its inverse, which guarantees that the rows (or columns) form an orthonormal set of vectors.

4.11: Orthogonality - Mathematics LibreTexts

Sep 16, 2022 — The span of these vectors is a subspace W of Rn. ... Thus a matrix is orthogonal if its rows (or...

Math LibreTexts

Orthogonal Matrix - an overview | ScienceDirect Topics

Any set of n nonzero orthogonal [orthonormal] vectors in ℝ n is an orthogonal [orthonormal] basis for ℝ n . ... If a...

ScienceDirect.com

Orthogonal Matrix: Definition, Types, Properties and Examples

A square matrix with real numbers or values is termed as an orthogonal matrix if its transpose is equal to the...

toppr.com

Can machines think logically? — Not yet

In realms where correctness of an answer is provable and accuracy is required, generative AI is not yet reliable — but it could be in the future.

Currently existing software programs called computer proof assistants can

- certify the correctness of a mathematical proof (computer formalization) or
- verify that a software program satisfies a desired specification (formal methods).

In principle, a generative AI could be designed to output text in a format that it could be checked by a computer proof assistant — a process known as autoformalization.

With such a protocol, AI-generated outputs in certain domains can be formally verified before being put into public use.

What are computer proof assistants?

A computer proof assistant or interactive theorem prover — such as `Agda`, `HOL Light`, `Isabelle`, `Lean`, `Mizar`, or `Rocq` (née `Coq`) — is a computer program that:

- knows the rules of a logical formal system (e.g., a foundation for mathematics), which a trusted core program (the kernel) uses to check the correctness of proofs
- is programmed (via the elaborator) to interpret statements written in an expressive formal language (the vernacular) in which a user writes their definitions, theorems, and proofs.

To a human user of an interactive theorem prover, writing a formal proof feels like writing code in a programming language, but with useful real-time feedback:

- typos may be pointed out by "type-checking errors"
- the proof assistant often communicates the standing assumptions and yet-to-be proven objectives midway through a complex proof.

A formalized proof of a true theorem

To illustrate, we give a formal proof in `Lean` that symmetric matrices define a subspace.

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

open Matrix

variable {n : Type}

/-- A matrix is `symmetric` if its `i j` entry equals its `j i` entry. -/
def symmetric (A : Matrix n n ℝ) : Prop :=
  (i j : n) → A i j = A j i

/-- A proof that the subset of symmetric matrices is a subspace. -/
def SymmetricMatrixSubspace : Subspace ℝ (Matrix n n ℝ) := sorry
```

A matrix $A = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix}$ is symmetric if $A_{12} = A_{21}$.

More generally, an $n \times n$ matrix A is symmetric if $A_{ij} = A_{ji}$ for all indices i and j.

A formalized proof of a true theorem

Lean's Infoview keeps track of assumptions and objectives at each stage of a proof.

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

open Matrix

variable {n : Type}

/-- A matrix is `symmetric` if its `i j` entry equals its `j i` entry. -/
def symmetric (A : Matrix n n ℝ) : Prop :=
  (i j : n) → A i j = A j i

/-- A proof that the subset of symmetric matrices is a subspace. -/
def SymmetricMatrixSubspace : Subspace ℝ (Matrix n n ℝ) where
  carrier := sorry
  add_mem' := sorry
  smul_mem' := sorry
  zero_mem' := sorry
```

```
▼SymmetricSubspace.lean:21:0
 ▼Expected type
   n : Type
   ⊢ Subspace ℝ (Matrix n n ℝ)
▶All Messages (4)
```

Lean automatically generates the proof obligations. To complete the proof, we must replace each "sorry" with code that satisfies Lean's proof checker.

A formalized proof of a true theorem

By filling in the carrier, we tell Lean that the elements of the subspace are the symmetric matrices.

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

open Matrix

variable {n : Type}

/-- A matrix is `symmetric` if its `i j` entry equals its `j i` entry. -/
def symmetric (A : Matrix n n ℝ) : Prop :=
  (i j : n) → A i j = A j i

/-- A proof that the subset of symmetric matrices is a subspace. -/
def SymmetricMatrixSubspace : Subspace ℝ (Matrix n n ℝ) where
  carrier := symmetric -- The elements of the subspace are symmetric matrices.
  add_mem' := by
    sorry
  smul_mem' := sorry
  zero_mem' := sorry
```

```
▼SymmetricSubspace.lean:16:4
 ▼Tactic state
  1 goal
   n : Type
   ⊢ ∀ {a b : Matrix n n ℝ}, a ∈ symmetric → b ∈ symmetric → a + b ∈ symmetric
▶All Messages (3)
```

Lean tells us that to prove closure under addition,
we must show that if A and B are symmetric, then $A + B$ is symmetric.

A formalized proof of a true theorem

Lean tells us that to prove closure under addition,

we must show that if A and B are symmetric, then $A + B$ is symmetric.

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

open Matrix

variable {n : Type}

/-- A matrix is `symmetric` if its `i j` entry equals its `j i` entry. -/
def symmetric (A : Matrix n n ℝ) : Prop :=
  (i j : n) → A i j = A j i

/-- A proof that the subset of symmetric matrices is a subspace. -/
def SymmetricMatrixSubspace : Subspace ℝ (Matrix n n ℝ) where
  carrier := symmetric -- The elements of the subspace are symmetric matrices.
  add_mem' := by -- A proof that if `A` and `B` are symmetric matrices, so is `A + B`.
    intro A B Asym Bsym i j
    sorry
  smul_mem' := sorry
  zero_mem' := sorry
```

```
▼SymmetricSubspace.lean:17:4
 ▼Tactic state
  1 goal
  n : Type
  A B : Matrix n n ℝ
  Asym : A ∈ symmetric
  Bsym : B ∈ symmetric
  i j : n
  ⊢ (A + B) i j = (A + B) j i
▶All Messages (3)
```

Thus for symmetric matrices A and B and indices i and j,

we must show that $(A + B)_{ij} = (A + B)_{ji}$.

A formalized proof of a true theorem

By definition of matrix addition, $(A+B)_{ij} = A_{ij} + B_{ij}$ and $(A+B)_{ji} = A_{ji} + B_{ji}$.

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

open Matrix

variable {n : Type}

/-- A matrix is `symmetric` if its `i j` entry equals its `j i` entry. -/
def symmetric (A : Matrix n n ℝ) : Prop :=
  (i j : n) → A i j = A j i

/-- A proof that the subset of symmetric matrices is a subspace. -/
def SymmetricMatrixSubspace : Subspace ℝ (Matrix n n ℝ) where
  carrier := symmetric -- The elements of the subspace are symmetric matrices.
  add_mem' := by -- A proof that if `A` and `B` are symmetric matrices, so is `A + B`.
    intro A B Asym Bsym i j
    simp only [add_apply]
    sorry
  smul_mem' := sorry
  zero_mem' := sorry
```

```
▼SymmetricSubspace.lean:18:4
 ▼Tactic state
  1 goal
  n : Type
  A B : Matrix n n ℝ
  Asym : A ∈ symmetric
  Bsym : B ∈ symmetric
  i j : n
  ⊢ A i j + B i j = A j i + B j i
▶All Messages (3)
```

Thus for symmetric matrices A and B and indices i and j,

we must show that $A_{ij} + B_{ij} = A_{ji} + B_{ji}$.

A formalized proof of a true theorem

Since A and B are symmetric, $A_{ij} = A_{ji}$ and $B_{ij} = B_{ji}$ so this equation holds:

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

open Matrix

variable {n : Type}

/-- A matrix is `symmetric` if its `i j` entry equals its `j i` entry. -/
def symmetric (A : Matrix n n ℝ) : Prop :=
  (i j : n) → A i j = A j i

/-- A proof that the subset of symmetric matrices is a subspace. -/
def SymmetricMatrixSubspace : Subspace ℝ (Matrix n n ℝ) where
  carrier := symmetric -- The elements of the subspace are symmetric matrices.
  add_mem' := by -- A proof that if `A` and `B` are symmetric matrices, so is `A + B`.
    intro A B Asym Bsym i j
    simp only [add_apply]
    rw [Asym, Bsym]
  smul_mem' := sorry
  zero_mem' := sorry
```

```
▼SymmetricSubspace.lean:18:20
 ▼Tactic state
  No goals
 ▼Expected type
  n : Type
  ⊢ Subspace ℝ (Matrix n n ℝ)
▶All Messages (2)
```

Now Lean tells us that there are no goals!

So we may move on to the remaining proof obligations …

A formalized proof of a true theorem

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

open Matrix

variable {n : Type}

/-- A matrix is `symmetric` if its `i j` entry equals its `j i` entry. -/
def symmetric (A : Matrix n n ℝ) : Prop :=
  (i j : n) → A i j = A j i

/-- A proof that the subset of symmetric matrices is a subspace. -/
def SymmetricMatrixSubspace : Subspace ℝ (Matrix n n ℝ) where
  carrier := symmetric -- The elements of the subspace are symmetric matrices.
  add_mem' := by -- A proof that if `A` and `B` are symmetric matrices, so is `A + B`.
    intro A B Asym Bsym i j
    simp only [add_apply]
    rw [Asym, Bsym]
  smul_mem' := by -- A proof that if `k ∈ ℝ` and `A` is a symmetric matrix, so is `k • A`.
    intro k A Asym i j
    simp only [smul_apply]
    rw [Asym]
  zero_mem' := by -- A proof that the zero matrix is symmetric.
    intro i j
    simp only [zero_apply]
```

▼ SymmetricSubspace.lean:26:0

▼ Tactic state

No goals

▼ Expected type

```
n : Type
⊢ Subspace ℝ (Matrix n n ℝ)
```

▶ All Messages (0)

Recent advances in autoformalization

While archives of formal proofs are much smaller than usual training data sets, there are recent advancements in autoformalization using generative AI:

- Autoformalization with Large Language Models: in 2022, a team from Google, Stanford, and Cambridge demonstrate that LLMs do reasonably well in translating natural language mathematics to formal theorem statements in `Isabelle/HOL`.
- Baldur: Whole-Proof Generation and Repair with Large Language Models: in 2023, a team from UMass Amherst, UIUC, and Google build a prototype capable of whole proof generation and proof repair in `Isabelle/HOL`.
- Solving olympiad geometry without human demonstrations: in 2024, Deep Mind debuted AlphaGeometry, a neuro-symbolic system made up of a neural language model and a symbolic deduction engine. In a benchmarking test of 30 International Mathematical Olympiad geometry problems, AlphaGeometry solved 25 within the standard Olympiad time limit.

Despite these advances, it will likely take some time before AIs can discover their own theorems that are mathematically interesting or relevant to the real world.

Applications on the horizon?

If fully automated theorem proving is a long way off, human/computer collaborations in machine-assisted proof are on the horizon.

I will close with a few predictions about near future:

- Generative AI will dramatically improve the experience of interactive theorem proving in a computer proof assistant, revolutionizing mathematical practice.
- Humans will drive advances in formal methods, by developing domain-specific formal systems that can be used to specify desired behavior.
- Generative AI will then accelerate the process of writing formally verifiable software and proof certificates that it matches the specification.

For example, in Formally Verified Software in the Real World, a team from UNSW and Melbourne explain how formally verified software can be used to design autonomous flight systems that are highly robust against cyber attacks.

What other applications are possible when machines can be trusted to "think" logically?