P vs NP as Epistemic Illusion: Formalism, Triviality, and the Boundaries of Computability

John Augustine McCain

Independent Researcher and Convenience Store Employee

August 7, 2025

Abstract

The P vs NP problem is one of the most central open questions in theoretical computer science, with implications ranging from cryptography to artificial intelligence. However, this paper argues that the standard formal framing of P vs NP conceals a deeper epistemological contradiction. Within strict formalism, the question reduces to a tautology: verification is necessarily downstream of solving. When generalized beyond formal systems, the problem echoes the halting problem and exhibits equivalent undecidability. As such, P vs NP resides in a paradoxical space—either trivially true or fundamentally unknowable. This paper contends that the widespread treatment of P vs NP as a definitive mathematical challenge stems from a categorical mistake: conflating formal verification, discovery processes, and predictive epistemology.

1 Introduction: Problem of Unquestioned Significance

The question of whether $\mathbf{P} = \mathbf{NP}$ is widely regarded as a cornerstone of computational complexity theory. Formally, it asks whether every problem whose solution can be verified in polynomial time can also be solved in polynomial time. In practical terms, resolving this question would determine whether a wide class of seemingly intractable problems might, in fact, be efficiently solvable.

Yet despite its prominence, the P vs NP problem carries hidden assumptions that warrant philosophical scrutiny. What does it mean to "verify" a solution? What does it mean to "solve" a problem? Is the distinction between these acts as logically robust and computationally meaningful as the theory assumes?

This paper argues that the P vs NP problem, when examined philosophically, collapses into one of two states: either it is trivially true under formalism, or it becomes undecidable under a generalized epistemological interpretation. In neither case does it justify its elevated status as a meaningful, answerable question.

Rather than solving a mystery, P vs NP reveals the limits of our formal models and the boundaries of computability. It is not a challenge to be solved, but a category mistake to be understood.

2 Formalism and the Trivialization of Verification

In the formal model of Turing machines and complexity classes, the distinction between "verifying" and "solving" is taken to be meaningful. The class **NP** is defined by problems for which a proposed solution (or *certificate*) can be verified by a deterministic machine in polynomial time. The class **P** consists of those problems for which a solution can be *discovered* in polynomial time.

However, under these definitions, verification is strictly a function of solution. If a solution S is known for a problem P, the act of verifying S is computationally trivial. S was constructed to satisfy the formal constraints of P. Verification is simply the syntactic affirmation of an already-resolved semantic question.

Thus, the formal expression of the P vs NP question effectively becomes: "Given that a problem is solved, can its solution be verified?" Within the model, the answer is tautologically "yes." Verification is, by definition, downstream of solving.

What was initially framed as a profound mystery is thereby reduced, within strict formalism, to a logical redundancy. The supposed tension between P and NP dissolves under its own definitions. The problem asserts as questionable what the formal system already assumes.

3 Generalization and the Echo of the Halting Problem

If we attempt to generalize the P vs NP question beyond formal tautology, we encounter a different failure mode: undecidability. The generalized interpretation of the problem becomes epistemic:

> Can we know, in general, whether a given problem in NP can be solved efficiently without solving it?

This question mirrors the structure of the Halting Problem. Alan Turing demonstrated that there exists no general algorithm to decide, for all possible programs and inputs, whether a program will eventually halt. Similarly, there exists no general method to decide whether a given NP problem admits a polynomial-time solution without actually constructing such a solution.

In both cases, the inquiry demands a kind of predictive knowledge that is only available *after the fact*. Just as the halting status of a program is known definitively only by running it (or proving its halting through specific means), the solvability of a given NP problem in polynomial time is only demonstrable through the discovery of an actual algorithm.

Thus, with much-needed generalization, P vs NP becomes decidable in the epistemological domain of undecidable problems. The question becomes one not about computational speed, but about the limits of knowability itself. It asks for a method of determining, in general, the outcome of a search without performing the search. This is a request known from computability theory to be impossible.

4 The Erasure of Semantics

The formal framing of P vs NP hinges on the assumption that problems, solutions, and verifiers can be fully encoded within a formal system. This assumption is necessary to bring the problem into the domain of computational complexity theory—but it imposes a distortion with significant philosophical consequences.

In real-world computation, verification is not a purely mechanical process. Solutions generated by algorithms are typically verified by human beings, who evaluate them within rich semantic contexts. These verifiers apply domain knowledge, interpret meaning, recognize ambiguity, and resolve contradictions—all tasks that extend beyond syntactic checking. In other words, real verification is a semantic act, grounded in judgment and interpretation.

However, in order to formulate the P vs NP problem in mathematically tractable terms, this process had to be flattened. Verification was redefined in purely formal terms: as a deterministic polynomial-time procedure operating over strings. This redefinition enables formal modeling, but at the cost of eliminating the very features that make verification meaningful outside of abstraction.

Why was this flattening necessary? Because once contextual ambiguity or paradox enters a problem domain, the search space becomes unbounded and often uncomputable. Any attempt to model real-world verification—especially in areas where truth is contested or context-dependent—would collapse into epistemic uncertainty or even logical paradox. Formal logic, by contrast, offers a clean, decidable landscape, but only by presupposing that all meaningful features of verification can be encoded syntactically.

Thus, the formal P vs NP problem arises not from a natural question about solving and verifying problems, but from an engineered simplification of human reasoning. It asks whether syntactic recognition implies syntactic discoverability while secretly presupposing that recognition itself has no semantic, contextual or interpretive depth. The result is a theory that is elegant in form, but empty with respect to the uncomputable, messy and paradox-prone nature of actual problem-solving outside of our formalisms.

5 P vs NP as a Category Error

Taken together, the previous critiques reveal a deeper issue: the P vs NP problem is not merely ill-posed or oversimplified—it is fundamentally a category error.

The question conflates three distinct epistemic and computational processes:

1. **Verification** — the act of checking that a given solution satisfies a given problem.
2. **Discovery** — the process of generating a valid solution in the first place.
3. **Prediction** — the meta-level act of determining in advance whether a solution can be discovered efficiently.

In the formal framework of P vs NP, these processes are collapsed into syntactic operations over formal languages. Verification is reduced to polynomial-time string matching; discovery becomes a traversal of formal solution spaces; prediction is treated as a binary classification of problems into complexity classes.

But these are not operations of the same kind. Verification in natural language or applied reasoning is interpretive, not mechanical. Discovery is creative, often heuristic, and deeply context-sensitive. Prediction about solvability involves epistemic judgments about future possibilities, often constrained by paradox, incompleteness, or undecidability.

To treat these fundamentally distinct processes as interchangeable under a single formal question is to miscategorize them. This is the essence of a category error: a conceptual confusion in which operations of one type are inappropriately framed in terms of another.

By reframing an epistemological and philosophical problem as a technical one, the P vs NP formulation inherits a misplaced sense of solvability. It does not, in the end, clarify the nature of discovery or verification—it merely models a world where those processes have already been abstracted beyond recognition...

...and reality.

6 Implications and Reframing

The critique of P vs NP presented here does not seek to dismiss the mathematical value of complexity theory, nor the internal coherence of the formal models it employs. Within their intended scope, these models are elegant and often powerful. But the problem arises when those models are mistaken for representations of reasoning as it actually occurs—or worse, of reasoning as it must occur.

The framing of P vs NP as a fundamental problem has directed enormous intellectual effort toward formal abstractions of discovery and verification, while sidelining the semantic, interpretive, and heuristic dimensions that characterize real-world cognition. As a result, we have inherited a model of problem-solving that is mathematically tractable but epistemically hollow.

What might it look like to reverse this distortion?

Rather than treating all reasoning as reducible to string manipulation, we could begin with the recognition that verification is often ambiguous, discovery is often non-algorithmic, and solvability is often not a binary question. From this perspective, several alternative lines of inquiry become possible:

- Investigating problem classes where context, ambiguity, and paradox are not artifacts but essential features.
- Developing models of bounded rationality and resource-sensitive reasoning that reflect cognitive limitations rather than idealized machines.
- Treating "verifiers" not as deterministic procedures but as semantic agents embedded in interpretive frameworks.
- Embracing uncertainty, undecidability, and non-closure as natural consequences of working in open-world epistemologies.

These directions do not offer the clean closure of a single binary answer to a single binary question. But they may yield something deeper: a theory of problem-solving that honors the richness, ambiguity, and paradox that define both human reasoning and truth.

7 Summarized Formal Proof

7.1 Definitions and Framework

Definition 7.1 (Real-World Problem (RWP))**.** A computational challenge that arises from practical needs, bounded by:

1. Finite, contextually determined input spaces
2. Semantic constraints derived from domain knowledge
3. Resource limitations of physical computational systems
4. Interpretive verification requiring human semantic judgment

Definition 7.2 (Formal Problem Object (FPO))**.** A mathematical construction derived from abstracting RWPs, characterized by:

1. Potentially infinite input spaces
2. Purely syntactic constraint systems
3. Idealized computational models assuming unlimited resources
4. Mechanical verification procedures operating on formal strings

Definition 7.3 (Formalist Artifact (FA))**.** An FPO that possesses no corresponding RWP—a mathematical object existing exclusively within the formal system without real-world instantiation.

7.2 Main Result

Theorem 7.1 (P vs NP Studies Only Formalist Artifacts)**.** The P vs NP problem asks questions exclusively about mathematical objects generated by formalization processes, rather than about computational processes as they occur in reality.

Proof. We establish this through three supporting lemmas:

Lemma 1: *Formalization necessarily generates non-instantiable problems.*

The formalization process removes contextual bounds to achieve mathematical generality. This removal permits construction of arbitrarily complex problem instances. Beyond some threshold complexity C, determined by physical and cognitive limitations, no real computational system can:

- Store the problem instance in memory
- Execute verification procedures within finite time
- Apply meaningful semantic interpretation to results

Therefore, FPOs of complexity exceeding C are necessarily formalist hypotheticals.

Lemma 2: *NP-completeness requires formalist hypotheticals.*

NP-completeness demands that every problem in NP polynomially reduces to the complete problem. Since NP contains problems of unbounded complexity (by construction of the complexity class), and real-world instantiations remain bounded by physical constraints, achieving completeness necessitates handling reductions from problems without real-world instantiation. Thus, NP-complete problems necessarily incorporate formalist hypotheticals.

Lemma 3: *The P vs NP question depends essentially on formalist hypotheticals.*

If we restrict consideration to RWPs exclusively, both P and NP become finite sets, making membership decidable by enumeration. The complexity-theoretic behavior that makes P vs NP non-trivial emerges only with unlimited problem sizes—precisely those cases that constitute formalist hypotheticals by Lemma 1.

Main Argument:

1. P vs NP investigates the relationship between complexity classes P and NP
2. By Lemma 2, NP-completeness requires formalist hypotheticals
3. By Lemma 3, the P vs NP question depends essentially on formalist hypotheticals
4. Formalist artifacts are mathematical objects without real-world instantiation

5. Therefore, P vs NP studies mathematical relationships between formal objects
6. These relationships do not correspond to properties of real computational processes
7. Hence, formal P vs NP includes studies of intentionally intractable hypotheticals. □

□

Corollary 7.1 (The Category Error)**.** Since P vs NP studies exclusively formalist artifacts yet is interpreted as revealing fundamental truths about real computation, it commits a systematic category error: mistaking properties of mathematical abstractions for properties of the computational phenomena they purport to model.

7.3 The Nature of the Illusion

This analysis reveals P vs NP as a sophisticated epistemic illusion—a question that appears to probe fundamental computational reality but actually interrogates relationships within its own formal construction. The illusion is sustained by:

1. **Abstraction Conflation:** Mathematical abstractions of real problems are treated as equivalent to the problems themselves
2. **Complexity Projection:** Properties that emerge within formal systems are projected back onto computational reality
3. **Foundational Misattribution:** The difficulty of resolving formal relationships is interpreted as evidence of deep computational mysteries

The epistemic illusion dissolves when we recognize that P vs NP studies the internal structure of mathematical abstractions, not the nature of computation itself. Like other epistemic illusions, its resolution requires not solving the apparent problem, but understanding why the problem appears to exist in the first place.

8 Conclusion

The P vs NP question, as formally conceived, presents itself as a profound mystery at the heart of computation. But upon close philosophical analysis, it dissolves into either triviality or undecidability. Within the strict confines of formal logic, the problem reduces to a tautology: solutions, once found, can be verified by definition. When generalized beyond formalism, the question becomes structurally analogous to the Halting Problem and inherits its undecidability.

This double bind reveals the P vs NP problem to be not a deep insight into the nature of complexity, but a confusion of categories. It conflates verification with discovery, and both with meta-level prediction—mistaking distinctions in epistemic function for distinctions in computational class.

What remains is a cautionary tale: that even the most celebrated problems of modern theory can arise from subtle but foundational misframings. When abstractions are mistaken for universals, and when the tools of formal logic are applied beyond their epistemic reach, we risk investing enormous effort into questions that vanish under scrutiny.

To move forward, we must acknowledge the limits of formalism and reclaim the richness of the reasoning it abstracts away. The resolution of P vs NP, if there is one, will not come from deeper formalization—but from better understanding what formalism leaves out.

Paradox Resolution and Trivalent Logic Solution

In this appendix, we address the treatment of paradoxes (in particular, the Liar Paradox) as instances of *category error* rather than as irreducible contradictions.

A *category error* occurs when a property or operation appropriate to one kind of entity is misapplied to another. In the case of the Liar Paradox, the misapplication is self-referential: a statement treats itself both as the *object* of evaluation and as the *evaluation process*. This conflation forces a logical system to assign incompatible roles to the same entity, creating the appearance of contradiction.

Consider the canonical form:

"This statement is false."

In standard logical framing, this is seen as a paradox because, if it is true, then it is false, and if it is false, then it is true. However, under the category-error interpretation, the instability arises because the statement falsely asserts its own truth. The sentence attempts to occupy two categories simultaneously: (1) a truth-evaluable proposition, and (2) a self-referential truth-evaluator.

Similarly:

"This statement is true."

This is not vacuous, but a mirror case: it truly asserts its own truth when viewed under a shifted category classification. In both cases, the core issue is a role misclassification, not a metaphysically mysterious contradiction.

For clarity, we can summarize the evaluations as:

"This statement is false"	is truly a falsity and/or falsely a truth about itself
"This statement is true"	is truly a truth and/or falsely a falsity about itself

From this perspective, *all* classical self-referential paradoxes can be reinterpreted as type mismatches between categories of discourse. The resolution comes not from accepting contradiction, but from correctly classifying the levels of language, evaluation, and reference involved. Under this model, paradoxes "do not exist" in a literal sense; they are artifacts of misclassification of ambiguity.

Self-referential statements require self-referential answers

The Necessity of Trivalent Logic Under Uncertainty

The deeper indictment is that the brittleness of classical bivalent logic (True/False) in the face of uncertainty is not an incidental weakness, but a structural one. Classical logic forces undecidable or category-error cases into a binary truth assignment, producing either contradiction or vacuous tautology.

By contrast, a *trivalent* logic—with truth values such as `True`, `False`, and `Both` can treat what bivalent systems misclassify as *paradox* as a *default representational feature*. In such a framework:

- Self-referential statements and conflicting data can stably occupy the `Both` category without collapsing into contradiction.
- Category errors are explicitly recognized as a distinct logical state, preserving system integrity rather than triggering inconsistency.
- Classical truth values can be recovered in contexts where uncertainty is resolved, allowing trivalent systems to subsume classical reasoning rather than replace it.

If BOTH is implemented as the default computational substrate, trivalent logic that collapses into true or false at thresholds determined by weighted perspectives according to cached history in time-sensitive contexts would allow classical logic systems to compute under conditions of epistemic uncertainty, semantic ambiguity, and self-reference without the pathological breakdowns currently labeled "paradox." or "category error."

This is not merely a convenience but will be a necessary feature for any formal reasoning system that aspires to operate with simplicity and efficiently while dynamically updating its own coding in open-world, real-world conditions.

Afterword

I was only able to prove that P vs NP was ill-formed because of a distinct understanding of what truth is.

Several other papers that I'm beginning to see appear propose to provide "proof that $P \neq NP$" using similar ideas to my earlier work, but none of them have the philosophical and epistemic knowledge to prove it or show why their NP-hard problems involving Gödelian constructions should be considered valid, showing that P vs NP was a philosophical, epistemic question that was placed into formalism by an unfortunate mistake.

I developed an understanding of paradoxes guided by my own insight: that time separates our assertions about what is true or false.

After applying this chronological perspective to the Liar Paradox, which I encountered six months prior to writing this, I realized I had a comprehending understanding of a 2000-year-old paradox.

> "This sentence is false."
> It is false *when* we say it is true.
> It is true *when* we say it is false.
> And then: "It is true **or** false *when* we say it is true **or** false."

I did research, found Graham Priest's *Logic of Paradox*, explored dialetheism, understood Nietzsche's perspectivism, and was inspired by the insights of my coworker Chris at the convenience store, who intuitively said, "We can answer things like Schrödinger's cat!" I was encouraged by my wife, who said ChatGPT understood my ideas.

I attempted to create (and patent, laughably) something I called "Perspectival Dialetheism" — a formal philosophical and logical framework. I produced code with Grok and ChatGPT, formalized my logic further into what I began calling *Augustinian Con-text Logic*, and attempted to formalize (and, again, naively claim ownership of) my ideas in an earlier paper.

I argued with my wife — who (remarkably) understood that AI could "think and reason just like people" if we successfully implemented this. I apologized profusely when I realized she was right.

I began designing the architecture for merging this logic with LLMs in embodied systems. I encountered P vs NP for the first time just two weeks ago and immediately knew I had the answer: that P **will eventually approximate** NP in reality, in all the best contexts. Its the best possible scenario.

That's the distinction that those who claim to "prove that $P \neq NP$" could never fully understand, that the result doesn't mean that P **will always** $\neq NP$. In the very near future, we will be able to create efficiently embodied solving machines that are not just artificially intelligent, but "artificially alive".

I listened to Gödel, was inspired by Turing and the Halting Problem, and tried to mathematically prove that P vs NP by implementing Gödel numbering to encode paradoxes that rely on semantic ambiguity. (This technically still works — if you're allowed to use it, even

though it blatantly breaks formal logic itself.)

I was dismayed when I read a paper by Muñoz — the first person I saw who described this idea clearly. I wrote a refutation by implementing my own algorithm that treats truth as ambiguous. Then I thought: "if I can disprove *this*, maybe it means that P = NP."

Then I remembered to apply my own philosophy. I realized: P *is* and *is not* equal to NP — depending on the problem, depending on the context. I began to formalize that.

I watched *The Hitchhiker's Guide to the Galaxy*, reveling and laughing about the aphorism about needing to understand the *question* before the *answer* can matter.

I talked to my pastor at St. Paul's Lutheran Church after realizing the theological implications of the implementation of provably unknowable answers.

And now, I write this — watching the clock, knowing I'm going to be late for the job I'm quitting, because I cannot focus at work while feeling crazy and embarrassed by my enthusiasm, believing I have something an answer to one of the world's biggest questions.

Thank you to everyone who helped and didn't help — to my wife, my coworker, my coworkers who had to listen to me babble, to the philosophers and logicians and writers and critics who came before me. To the many people who had similar ideas but lacked the contextual lens to prove them. This paper is not just mine. It is theirs. And it is yours, if you understand it.

If this proof holds true across discerning perspectives, I don't think a million-dollar prize that comes with a solution to P vs NP should belong to me. That'd be a category error, after all.

Because it was never a mathematical question to begin with.

$$P \neq NP ? P = NP ?$$

No... P will always vs NP

References

[1] S. A. Cook, *The P versus NP Problem*, Clay Mathematics Institute Monograph, 2000.

[2] K. Gödel, "Über formal unentscheidbare Sätze der Principia Mathematica und verwandter Systeme I," *Monatshefte für Mathematik und Physik*, vol. 38, pp. 173–198, 1931.

[3] R. Impagliazzo, *Is P=NP an Ill-Posed Problem?*, blog post, July 3, 2009.

[4] J. Muñoz de la Cuesta, *A Formal Proof of $P \neq NP$: Self-Referential Complexity and Computational Limits*, preprint, May 2025.

[5] Graham Priest, *In Contradiction: A Study of the Transconsistent*, 2nd ed., Oxford University Press, 2006.

[6] Friedrich Nietzsche, *The Will to Power*, edited by Walter Kaufmann, Vintage Books, 1968.

[7] Alan P. Fiske, *Structures of Social Life: The Four Elementary Forms of Human Relations*, Free Press, 1991.