Logical Collapse: Why the *P vs NP* Problem Cannot Be Resolved

Kevin E. Wells
Assistant Professor, School of Education
The University of Southern Mississippi
Kevin.E.Wells@usm.edu

August 23, 2025

Abstract

This paper argues that the *P vs NP* problem is not merely unsolved, but structurally unsolvable. The problem demands a universal binary answer, either $P = NP$ or $P \neq NP$, despite the existence of counterexamples that contradict both possibilities. Binary problems exhibit structural equivalence between solving and verifying, implying $P = NP$ within that subclass. Structured problems, including multinomial and continuous classes, exhibit a necessary asymmetry between solving and checking, implying $P \neq NP$ in those cases. These two realities are logically incompatible with any universal claim. Therefore, the *P vs NP* problem, as currently framed, collapses under its own assumptions.

1 Introduction

The *P vs NP* problem is a central challenge in theoretical computer science. Formally, it asks whether every problem whose solution can be verified in polynomial time can also be solved in polynomial time [1]. This framing suggests a binary outcome: either all NP problems can be solved in P ($P = NP$), or they cannot ($P \neq NP$).

This paper argues that this framing is not merely unsolved, but logically incoherent. Both directions of the binary are falsified by counterexamples. Thus, the problem, as currently defined, cannot be resolved.

2 Binary Problems Imply $P = NP$

Lemma 1 (Binary Structural Equivalence)**.** *Binary decision problems are a subclass where solving and verifying are structurally identical.*

A canonical example is 2-SAT: algorithms like strongly connected components simultaneously solve and verify the solution [3]. The logic required to solve is indistinguishable from that required to check.

$$\text{Solve} = \text{Check} \quad \Longleftrightarrow \quad P = NP \quad \text{(in this subclass)} \tag{1}$$

This is not a philosophical analogy but a structural identity. Binary problems satisfy $P = NP$ within their internal logic.

3 Structured Problems Imply $P \neq NP$

Lemma 2 (Asymmetry in Structured Problems)**.** *Structured problems with complex or continuous solution spaces exhibit computational asymmetry between solving and verifying.*

Examples include graph coloring, satisfiability beyond 2-SAT, integer programming, and real-valued optimization [2]. In these problems, it is easy to verify a candidate solution but exponentially difficult to find one.

$$\text{Check} \not\Rightarrow \text{Solve} \quad \Longrightarrow \quad P \neq NP \quad \text{(in this subclass)} \tag{2}$$

The asymmetry stems from intrinsic structural complexity, not algorithmic weakness.

4 The Logical Contradiction

The *P vs NP* problem demands a universal binary answer. But:

- If $P = NP$, structured problems like 3-coloring violate the claim.
- If $P \neq NP$, binary problems like 2-SAT violate the claim.

Thus:

$$\begin{aligned} P = NP &\implies \text{False (contradicted by structured problems)} \\ P \neq NP &\implies \text{False (contradicted by binary problems)} \end{aligned}$$

Corollary 1. *No universal binary truth is possible under the current framing. The problem is formally unsolvable.*

5 Epilogue: The Carnival Ring Toss

There is a reason no one has solved the *P vs NP* problem. And it is not for lack of ingenuity. It is because the problem, as posed, collapses logically. You cannot prove $P = NP$ because complex problems violate that equality. You cannot prove $P \neq NP$ because binary problems satisfy it.

Analogy. Like a carnival ring toss:

- The rules sound simple.
- The target looks hittable.
- But the rim is just slightly too wide, the ring just slightly too light, and the framing just slightly too wrong.

Years of mathematical effort may refine the throw—but never land the ring. The only honest move left is to name the contradiction and stop pretending the ring ever fit the bottle.

> *"Strange game. The only winning move is not to play."*
> — WOPR, *WarGames* (1983)

Keywords

Computational complexity, P vs NP, logical contradiction, structural asymmetry, decision theory, falsifiability, computational logic

References

[1] Cook, S. A. (1971). The complexity of theorem-proving procedures. In *Proceedings of the third annual ACM symposium on Theory of computing* (pp. 151–158).

[2] Garey, M. R., & Johnson, D. S. (1979). *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W.H. Freeman.

[3] Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C. (2009). *Introduction to Algorithms* (3rd ed.). MIT Press.

[4] Sipser, M. (2012). *Introduction to the Theory of Computation* (3rd ed.). Cengage Learning.

[5] Aaronson, S. (2013). Why philosophers should care about computational complexity. In *Computability: Turing, Gödel, Church, and beyond* (pp. 261–328). MIT Press.