Where Exactly Does NP Emerge?
A Structural Re-Reading of Cook's Original Formulation

Alexey A. Nekludoff

AstraVerge Research

E-mail: an@astraverge.org
ORCID: 0009-0002-7724-5762

May 31, 2026

Abstract

This paper re-examines Cook's original formulation of the P versus NP problem. The central question is not whether $P = NP$ or $P \neq NP$, but where, in the original construction, NP emerges as a genuinely new computational class.

Cook's formulation proceeds from deterministic polynomial-time computation to a polynomial-time decidable relation $R(x, y)$, and then to the existentially extended form

$$L = \{x \mid \exists y\, R(x, y)\}.$$

We argue that this transition introduces an additional finite parameter and an existential quantifier, but does not explicitly introduce a new computational mechanism.

The paper further observes that finite power expressions such as n^x and x^n, when both parameters are finite and given, remain polynomial objects. Thus the appearance of an allegedly new class requires additional justification beyond the mere introduction of a certificate parameter.

The purpose of this paper is therefore not to solve the P versus NP problem, but to question whether NP appears as a distinct class inside Cook's original definition at all.

Keywords: P versus NP; NP_{Cook}; Cook's formulation; polynomial relations; finite structures; existential verification; certificates; parameter symmetry; representation dependence; interpretative complexity; foundations of computation

Contents

1 Introduction

The P versus NP problem is usually presented as a question about the relation between efficient search and efficient verification. Informally, one asks whether every problem whose solution can be checked efficiently can also be solved efficiently.

This formulation is suggestive, but it already contains an interpretative step. It treats verification and search as belonging to different computational regimes. The present paper asks whether this distinction is already present in Cook's original formulation.

We focus on the structural sequence

$$P \longrightarrow R(x, y) \longrightarrow \exists y\, R(x, y).$$

The question is simple:

At what exact point in this sequence does NP emerge as a new computational class?

The thesis defended here is that no such point is explicit in the original construction. What appears is not a new computational mechanism, but a deterministic polynomial relation with an additional finite parameter.

2 Cook's Original Construction

Cook defines the problem using languages over finite strings and Turing machines. Informally, P consists of decision problems solvable by an algorithm within a number of steps bounded by a fixed polynomial in the length of the input.

The NP side is introduced through the existence of a checking relation. In structural form, the definition has the shape

$$L = \{x \mid \exists y\, R(x, y)\},$$

where $R(x, y)$ is decidable in polynomial time.

Several facts are immediate:

1. x is finite.
2. y is finite.
3. $R(x, y)$ is deterministic.
4. Verification is polynomial.

Thus the construction consists entirely of finite deterministic objects.

3 The Missing Moment of Emergence

The transition

$$R(x, y) \longrightarrow \exists y\, R(x, y)$$

adds an existential quantifier over a finite certificate.

However, this does not by itself introduce a new computational mechanism. The checking procedure remains deterministic. The objects remain finite. The relation remains polynomially decidable.

Therefore, if NP is to be regarded as a fundamentally new class, the moment of its emergence must be identified.

The original construction gives us:

$$P \longrightarrow R(x, y) \longrightarrow P',$$

where P' denotes a polynomially decidable relation extended by an additional finite parameter.

It does not obviously give us:

$$P \longrightarrow NP$$

as a transition to a new computational ontology.

4 Finite Parameters and Finite Power Structures

Cook's original formulation is constructed entirely from finite objects.

Inputs are finite strings, certificates are finite strings, and the verification relation is evaluated by a finite deterministic computation.

Let n and x be finite parameters.

Consider the expressions

$$n^x$$

and

$$x^n.$$

Within a finite setting, both expressions are finite power structures.

More generally, for finite parameters a and b,

$$a^b$$

is defined by a finite sequence of multiplications

$$a^b = \underbrace{a \cdot a \cdot \ldots \cdot a}_{b \text{ times}}.$$

Thus the construction of a^b requires only:

- finite parameters;
- a finite number of operations;
- deterministic evaluation.

No infinite process appears in the construction itself.

Consequently, the expressions

$$n^x$$

and

$$x^n$$

share the same finite algebraic form

$$a^b.$$

The distinction between "polynomial" and "exponential" behaviour cannot be inferred from the finite algebraic form alone.

Rather, it appears only after an additional interpretative step is introduced.

One parameter is designated as the variable whose growth is to be studied, while another parameter is designated as fixed.

Only after this assignment does the familiar distinction between "polynomial growth" and "exponential growth" emerge.

However, Cook's original formulation contains neither infinite inputs, nor infinite certificates, nor infinite computations.

Within the finite structure of the definition, all parameters remain finite objects of the same general kind.

Therefore the qualitative separation between polynomial verification and non-polynomial search cannot be read directly from finite power structures themselves. It requires an additional interpretative layer concerning parameter roles and growth.

5 Search, Verification, and the Certificate Parameter

This paper does not claim that search and verification are operationally identical.

The claim is narrower.

If a certificate y is introduced, then the formal structure of the problem changes from a one-argument relation to a two-argument relation:

$$R(x) \longrightarrow R(x, y).$$

But the introduction of a second finite argument does not automatically create a new computational class.

It creates a relation over pairs.

The burden is therefore on the interpretation that treats

$$\exists y\, R(x, y)$$

as the birth of a fundamentally new class rather than as an existential extension of deterministic polynomial verification.

6 Representation, Size, and Interpretation

In Cook's formulation the size of an instance is measured by the length of its encoding.

However, the length of an encoding is not an intrinsic property of the underlying object. It depends on the chosen representation.

Consider the same natural number represented in several encoding schemes.

Representation	Example	Length
Binary	1111101000	10
Decimal	1000	4
Ternary	1101001	7
Roman	M	1
Unary	111...111	1000

The underlying object is identical, yet the measured size differs substantially.

Consequently, the parameter

$$n = |enc(x)|$$

depends on the encoding function.

Therefore complexity classes defined through polynomial bounds in n are not properties of the object alone. They depend on the chosen representation.

6.1 Normalized Representation Size

The standard notion of input size identifies the size of an object with the number of symbols in its encoding.

However, this choice is not unique. The same object may be represented using encoding schemes whose symbols carry different amounts of information.

For example:

- a binary symbol carries one bit of information;
- a ternary symbol carries $\log_2 3$ bits;
- a byte-sized symbol carries 8 bits;
- compressed encodings may represent multiple logical symbols by a single physical symbol.

This suggests introducing a representation-independent measure based on information content rather than symbol count.

Definition 1 (Normalized Representation Size)**.** *Let*

$$I(x)$$

denote the information content of the encoded object x, measured in bits.
Let

$$I_0$$

denote the information capacity of a chosen reference symbol.
Define

$$\rho(x) = \frac{I(x)}{I_0}.$$

Unlike ordinary string length, the quantity $\rho(x)$ is not restricted to integer values.

Consequently, three qualitatively distinct regimes appear.

Sub-unit regime

$$0 < \rho < 1.$$

The object contains less information than a single reference symbol.

For example, if the reference symbol stores eight bits and a certificate contains only three bits, then

$$\rho = \frac{3}{8}.$$

Unit regime

$$\rho = 1.$$

The object contains exactly one unit of reference information.

Super-unit regime

$$\rho > 1.$$

The object contains more information than a single reference symbol.

Traditional complexity analysis usually begins in this regime because input size is identified with the number of encoded symbols.

The existence of the sub-unit regime demonstrates that the notion of "size" depends on the chosen informational granularity.

Therefore the parameter traditionally denoted by n is not a purely structural property of the object itself. It is a consequence of a particular choice of representation and measurement scale.

This observation suggests that complexity classifications based solely on symbol count may conceal representation-dependent effects that become visible once size is treated as a normalized informational quantity.

7 From Polynomial Computation to NP_{Cook}

The purpose of this section is to reconstruct Cook's original formulation as a sequence of increasingly expressive constructions and to identify the exact point at which NP is claimed to emerge.

7.1 Polynomial Computation

Let

$$P$$

denote the class of languages decidable by a deterministic Turing machine within a polynomial number of steps.

The defining properties are:

- finite inputs;
- deterministic computation;
- polynomial running time.

7.2 Polynomial Relations

Definition 2 (Polynomial Relation)**.** *Let*

$$R \subseteq \Sigma^* \times \Sigma^*.$$

We write

$$R \in P_{rel}$$

if there exists a deterministic Turing machine and a polynomial p such that

$$R(x, y)$$

can be decided in at most

$$p(|x| + |y|)$$

steps.

Remark 1. *The class P_{rel} contains only finite objects, finite parameters, deterministic computation, and polynomially bounded verification.*

No non-deterministic computational mechanism appears at this stage.

7.3 Cook's Existential Extension

Definition 3 (Cook Extension)**.** *For every relation*

$$R \in P_{\text{rel}},$$

define

$$L_R = \{x \in \Sigma^* \mid \exists y\, R(x, y)\}.$$

Define

$$NP_{\text{Cook}} = \{L_R \mid R \in P_{\text{rel}}\}.$$

Proposition 1.

$$P \subseteq NP_{\text{Cook}}.$$

Proof. Let

$$L \in P.$$

Then there exists a deterministic polynomial-time decision procedure for L. Define the relation

$$R_L(x, y) \iff x \in L.$$

The relation R_L ignores the second argument and is therefore in P_{rel}.
Hence

$$L = \{x \in \Sigma^* \mid \exists y\, R_L(x, y)\}.$$

Therefore

$$L \in NP_{\text{Cook}}.$$

Thus

$$P \subseteq NP_{\text{Cook}}.$$

□

7.4 The Emergence Question

The transition

$$P \longrightarrow P_{rel} \longrightarrow NP_{Cook}$$

is entirely constructed from:

- finite strings;
- finite certificates;
- deterministic computation;

- polynomially decidable relations.

This raises the central question of the paper:

Where exactly does NP emerge as a new computational class?

The transition

$$R(x, y) \longrightarrow \exists y\, R(x, y)$$

introduces an existential quantifier over a finite parameter.

However, no new machine model, no new computational operation, and no infinite object are explicitly introduced.

The remainder of this paper investigates whether the emergence of NP_{Cook} is a structural property of the construction itself or an interpretative consequence of later assumptions about parameter roles, encoding, and growth.

8 Parameter Symmetry

The relation

$$R(x, y)$$

is defined over

$$\Sigma^* \times \Sigma^*.$$

Thus both arguments belong to the same underlying domain of finite strings.

The formula

$$\exists y\, R(x, y)$$

distinguishes the positions of the arguments, but does not by itself assign semantic roles to them.

In particular, the interpretation

$$x = \text{instance}, \qquad y = \text{certificate}$$

does not follow from domain membership alone.

Rather, it constitutes an additional semantic interpretation imposed upon the relation.

9 Interpretative Dependence of NP_{Cook}

The preceding analysis motivates the following interpretative thesis.

Interpretative Dependence Thesis.

Complexity classes defined through polynomial bounds in

$$n = |enc(x)|$$

are not purely structural properties of the underlying finite objects.

Their extension depends on:

1. the choice of encoding function *enc*;

2. the choice of size measure;

3. the interpretation of one parameter as the growth variable;

4. the interpretation of another parameter as a certificate.

In Cook's construction the transition

$$R(x, y) \longrightarrow L = \{x \mid \exists y\, R(x, y)\}$$

introduces an existential quantifier over a finite parameter.
The construction itself does not explicitly introduce:

- a new machine model;
- a new computational step;
- a new finite object type;
- an infinite structure.

Therefore the interpretation of

$$NP_{Cook}$$

as a computational regime fundamentally distinct from deterministic polynomial verification requires assumptions that are not explicit in the construction itself.

In this sense, NP_{Cook} may be viewed not as the emergence of a new computational mechanism, but as an interpretative layer built upon deterministic polynomial verification.

Remark 2. *The distinction between instance and certificate plays a role analogous to the distinction between variable and parameter in finite power expressions.*
Neither distinction is encoded in the algebraic structure itself.
Both arise through interpretation.

10 Conclusion

This paper has not attempted to prove either

$$P = NP$$

or

$$P \neq NP.$$

Instead, it has examined the logical structure of Cook's original construction.
The analysis proceeded through the sequence

$$P \longrightarrow P_{rel} \longrightarrow NP_{Cook},$$

where P_{rel} denotes deterministic polynomial relations and

$$NP_{Cook} = \{L_R \mid R \in P_{rel}\}.$$

Throughout this construction all objects remain finite, all verification procedures remain deterministic, and all computations remain polynomially bounded.
The transition

$$R(x, y) \longrightarrow \exists y\, R(x, y)$$

introduces an existential quantifier over a finite parameter, but does not explicitly introduce a new computational mechanism.

The paper therefore leaves the reader with a prior question that appears logically antecedent to the traditional P versus NP problem:

Where exactly does NP emerge?

If the emergence of NP cannot be identified within the structure of the original construction itself, then the distinction between P and NP_{Cook} may depend not only on computation, but also on representation, parameter interpretation, and the conceptual framework through which the construction is read.

The central observation of this paper is therefore not that

$$P = NP$$

or that

$$P \neq NP.$$

Rather, it is that Cook's original construction does not explicitly identify a point at which a fundamentally new computational class comes into existence.

The emergence of NP_{Cook} appears only after additional interpretative choices concerning representation, parameter roles, and growth classification have been imposed upon an otherwise deterministic polynomial framework.

A Formal Interpretation Schemes

The notion of complexity implicitly depends on a choice of representation and interpretation.

Definition 4 (Interpretation Scheme)**.** *An interpretation scheme is a tuple*

$$\mathcal{I} = (\Sigma^*, enc, \ell, \rho),$$

where

- *enc is an encoding function;*
- *ℓ is a size measure;*
- *ρ is a role assignment.*

The role assignment

$$\rho : \{1, \ldots, n\} \to \mathcal{R}$$

maps argument positions to semantic roles.

Typical roles include

$$\mathcal{R} = \{\text{instance}, \text{certificate}, \text{parameter}\}.$$

For Cook's relation

$$R(x, y),$$

the conventional interpretation is

$$\rho(1) = \text{instance}, \qquad \rho(2) = \text{certificate}.$$

However, this assignment is not part of the relation itself.
It belongs to the interpretation scheme.
Consequently, the transition

$$R(x, y) \to \exists y\, R(x, y)$$

should properly be viewed as occurring relative to

$$\mathcal{I},$$

rather than as a purely structural transformation.

B Three Ontologies of NP

The expression "NP" is used for several formally equivalent but conceptually distinct constructions.

B.1 Cook-Style NP

$$NP_{Cook} = \{L_R \mid R \in P_{rel}\}.$$

The primitive object is a deterministic polynomial relation.
NP emerges through existential extension.

B.2 Verifier-Based NP

A language belongs to NP if there exists a polynomial-time verifier

$$V(x, y).$$

The primitive object is a verification procedure.
NP emerges through certificates.

B.3 Nondeterministic-Turing-Machine NP

A language belongs to NP if it is accepted by a nondeterministic Turing machine in polynomial time.

The primitive object is a machine with nondeterministic branching.
NP emerges through machine semantics.

B.4 Observation

These formulations are known to be extensionally equivalent.

However, they introduce NP through different primitive notions:

Formulation	Primitive notion
Cook	Relation
Verifier	Verification procedure
NDTM	Machine semantics

The present paper focuses exclusively on the first construction.

C Examples of Representation Dependence

This appendix illustrates how complexity descriptions may depend on representation.

C.1 Unary and Binary Encoding

Consider a natural number N.

Unary representation requires

$$N$$

symbols.

Binary representation requires

$$\lfloor \log_2 N \rfloor + 1$$

symbols.

Thus the same object receives radically different size assignments.

C.2 Roman Numerals

The number

$$1000$$

may be represented as

$$1000$$

(decimal),

$$1111101000$$

(binary),
or

$$M$$

(Roman).

The underlying object remains unchanged, while the measured length changes substantially.

C.3 Succinct Encodings

Many finite structures admit compressed descriptions whose lengths are asymptotically smaller than explicit encodings.

In such cases, complexity classifications may depend on whether the compressed representation or the expanded representation is chosen as the input object.

C.4 Observation

These examples do not invalidate complexity theory.

They illustrate that complexity classes are defined relative to a representation scheme rather than directly on abstract objects.

References

[1] S. Cook, "The p versus np problem," in *The Millennium Prize Problems*, Clay Mathematics Institute, 2006.

[2] M. Sipser, *Introduction to the Theory of Computation*. Cengage Learning, 2012.

[3] C. H. Papadimitriou, *Computational Complexity*. Addison-Wesley, 1994.