Modular Residue Method for UNSAT Detection

Autor: Jamesson Richard Campos Santos da Graça

Independent Researcher – Brazil

April 2025

Introduction

Boolean satisfiability (SAT) is the cornerstone of theoretical computer science and logic, serving as the first problem proven to be NP-complete by Cook and Levin in the 1970s. While countless heuristics and solvers exist for SAT problems, determining when a formula is unsatisfiable (UNSAT) remains a central and challenging task.

This paper introduces a novel method to detect UNSAT formulas by applying modular arithmetic over weighted clause encodings. The technique transforms a CNF formula φ into a numerical object — a residue vector modulo M — based on weights assigned to each clause. We hypothesize that for UNSAT formulas, these weighted modular sums collapse to zero under sufficiently large and varied modulus M and weight functions. In contrast, SAT formulas exhibit non-zero residual patterns.

Our work stems from empirical exploration of residue patterns across hundreds of SAT and UNSAT instances, including classical constructs like the Pigeonhole Principle, Tseitin formulas, and randomly generated 3-SAT instances. The results consistently showed a dichotomy: formulas known to be UNSAT exhibit $S(\varphi) \equiv 0$, while SAT formulas do not.

This observation motivates the development of a formal conjecture, experimental methodology, and computational framework to explore the implications of modular residue behavior on formula satisfiability. While the approach is not yet a formal decision procedure, its repeatability, scalability, and alignment with logical structure suggest it may offer a new heuristic — and possibly a path to new theoretical insights into the nature of UNSAT.

The remainder of this paper is organized as follows. Section 2 presents the theoretical background. Section 3 introduces the core conjecture and early

formal results. Section 4 discusses the implementation framework. Section 5 details our experimental setup and results. Section 6 offers a critical discussion, and Sections 7–10 provide concluding remarks, references, appendices, and data links.

Theoretical Background

The Boolean Satisfiability Problem (SAT) asks whether there exists na assignment of truth values to variables such that all clauses in a given propositional formula φ in conjunctive normal form (CNF) evaluate to true. If no such assignment exists, φ is said to be unsatisfiable (UNSAT). Despite decades of research and countless heuristics, the task of proving unsatisfiability remains computationally intensive and fundamentally unresolved in the context of P vs NP.

This work explores a numeric encoding of SAT/UNSAT formulas using modular arithmetic and clause-level weights. The core idea is to treat logical formulas as algebraic objects, mapping them into integers and then analyzing the sum of these integer encodings modulo a chosen base M.

Clause Encoding and Weighted Sums

Let $\varphi = C_1 \wedge C_2 \wedge \ldots \wedge C_{\square}$ be a CNF formula with m clauses. Each clause C_i is encoded as na integer c_i that reflects the structure and literal composition of the clause. A weight function $w: \mathbb{N} \rightarrow \mathbb{Z}$ assigns a weight to each clause based on its position i.

The central numeric object in our approach is the modular weighted sum:

$$S(\varphi; w, M) = \sum [w(i) \times c_i] \bmod M$$

Where:

C_i is the integer representation of clause C_i,

W(i) is the weight of clause i (e.g., i, i^2, 2^i , or hash-based),

M is a large prime or power-of-two modulus.

We consider four primary families of weight functions:

Sequential: w(i) = i

Polynomial: w(i) = i^2

Exponential: w(i) = 2^i

Hash-based: w(i) = SHA256(i) mod M

Each weight function defines a different projection of the formula's structure into modular residue space.

Heuristic Observation and Conjecture Basis

Empirical testing reveals that for known UNSAT formulas, such as those derived from the pigeonhole principle (PHP(m,n) with m > n), the modular sum S(φ) tends to equal zero for multiple choices of M and w(i). Conversely, SAT formulas produce non-zero residues that vary significantly depending on the weight function.

This pattern suggests that unsatisfiability might impose a form of symmetry or redundancy that causes the weighted contributions of clauses to cancel each other out under modulo M, forming a kind of residue-level inconsistency invariant.

Mathematical Framework

Let w = (w(1), ..., w(m)) be the weight vector, and c = (c_1 , ..., c□) the encoded clause vector. Then:

$$S(\varphi) \equiv w \cdot c^t \bmod M$$

$$\forall\, M \in \mathcal{M}, \forall\, w \in \square:$$

$$W \cdot c^t \equiv 0 \bmod M$$

This expression defines the core algebraic intuition: unsatisfiable formulas collapse their residue projections under various bases.

Future Formalization

We aim to extend this foundation toward a spectral or algebraic proof, exploring the structure of the clause matrix and its null-space under modular projections. Preliminary results suggest that UNSAT formulas are mapped to the zero vector under all tested weight functions — a property that, if proven generally, may lead to a new class of unsatisfiability certificates.

Formal Conjecture and Demonstration Attempt

We now formalize the Modular UNSAT Conjecture and present the initial steps toward a proof, based on algebraic reasoning and modular projections.

3.1 Conjecture Statement

Let φ be a Boolean formula in CNF form, composed of m clauses C_1 , C_2 , ..., C□. For each clause C_i, we define na encoding $c_i \in \mathbb{Z}$ and a weight w(i) from a class of admissible weight functions □. Let M be a sufficiently large modulus (typically a large prime).

Conjecture:

If φ is UNSAT, then:

$$S(\varphi; w, M) = \sum [w(i) \times c_i] \equiv 0 \bmod M$$

For all w ∈ □ and all M ∈ $\mathcal{M}$

3.2 Clause Encoding

Each clause C_i is encoded as na integer c_i using a canonical mapping, such as:

$$C_i = \sum [a_{ij} \times 2^j]$$

Where a_{ij} represents the presence and polarity of literals in the clause.

3.3 Algebraic Reformulation

Let $c = (c_1, c_2, ..., c_□) \in \mathbb{Z}^m$ be the clause encoding vector, and w = (w(1), w(2), ..., w(m)) be a weight vector from □. Then:

$$S(\varphi) = w \cdot c^t \bmod M$$

For na UNSAT formula, we observe:

$$\forall\ w \in □: w \cdot c^t \equiv 0 \bmod M$$

This implies that c lies in the modular nullspace of the entire weight space:

$$C \in \text{Null}_{□ₒd_{(\ □)}}(□)$$

3.4 Orthonormal Basis of Weights

We define a basis $\mathcal{B}$ of independent weight functions:

$$\mathcal{B} = \{ w^1, w^2, w^3, ... \}$$

$$W^1(i) = i$$

$$W^2(i) = i^2$$

$$W^3(i) = 2^i$$

$W^4(i) = SHA256(i) \bmod M$

If $w^k \cdot c^t \equiv 0$ for every $w^k \in \mathcal{B}$, then c is orthogonal (mod M) to the entire basis. This strongly implies that $c \equiv 0 \bmod M$ or lies in a constrained subspace for UNSAT formulas.

3.5 Empirical Base Cases

To support this claim, we verified the conjecture for known UNSAT formulas:

Pigeonhole Principle: PHP(m,n) with m > n (e.g., PHP(6,5))

Tseitin graphs with odd parity

XOR-SAT formulas with no consistent solution

For each, we evaluated:

$S(\varphi) = \sum [w(i) \times c_i] \bmod M$

And observed consistent modular zero across weights and modulus values.

3.6 Toward a Proof

A formal proof would require showing that any unsatisfiable formula, when encoded as a clause vector c, is orthogonal (modulo M) to every admissible weight vector. We propose constructing na abstract residue space where SAT formulas generate non-zero projections and UNSAT formulas collapse to the zero vector under all bases:

$\forall\ \varphi \in$ UNSAT:

$C \in \bigcap \text{Null}_{\square}(\square)$ (w), for all $w \in \square$

This leads to a possible new algebraic criterion for unsatisfiability that could complement existing logical and resolution-based methods.

Implementation and Computational Framework

The modular residue method was implemented using a custom computational pipeline developed in Python and C++, with optimizations for speed and clarity. This framework processes CNF formulas, assigns clause weights, computes modular residue sums, and organizes results into structured datasets and visualizations.

4.1 Formula Input and Clause Parsing

We adopt the standard DIMACS CNF format as input. Each file contains a set of clauses of the form:

$(x_1 \vee \neg x_3 \vee x_7)$, $(\neg x_2 \vee x_4)$, ...

Each clause is parsed and encoded numerically using:

$$C_i = \sum [\alpha_{ij} \times 2^j]$$

4.2 Weight Strategies

Each clause C_i is assigned a numeric weight w(i) from a configurable strategy. The framework includes:

Sequential: $w(i) = i$

Polynomial: $w(i) = i^2$

Exponential: $w(i) = 2^i$

SHA256-based: $w(i) = SHA256(i) \bmod M$

These strategies form a modular basis of projections in the residue space, each revealing different structural properties of the formula.

4.3 Modular Evaluation Engine

The core engine computes:

$$S(\varphi; w, M) = (\sum [w(i) \times c_i]) \bmod M$$

Where:

M is the number of clauses

C_i is the encoded clause

W(i) is the weight function

M is a prime or large power-of-two

The modulus M can be selected arbitrarily, with common values ranging from M = 1009 up to 1,000,003. All experiments verify residue behavior across at least three distinct M values to ensure stability.

4.4 Dataset and Formula Types

The framework processes multiple categories of formulas:

Structured UNSAT: PHP(m,n), Tseitin graphs, XOR contradictions

Structured SAT: Balanced formulas with known assignments

Random 3-SAT: Variable count from 20 to 100, clause count from 80 to 400

All .cnf files are stored and versioned with identifiers like PHP(6,5), XOR8, 3SAT20a.

4.5 Output Format and Visualization

Each experiment outputs:

Formula ID

Clause count m

Weight type used

Modulus M

Computed residue S(φ) mod M

SAT/UNSAT classification

Results are compiled into .csv tables and visualized using Matplotlib. Charts include distribution plots, SAT vs UNSAT comparisons, and multi-formula residue summaries.

4.6 Code Availability

All source code and datasets are publicly available at:

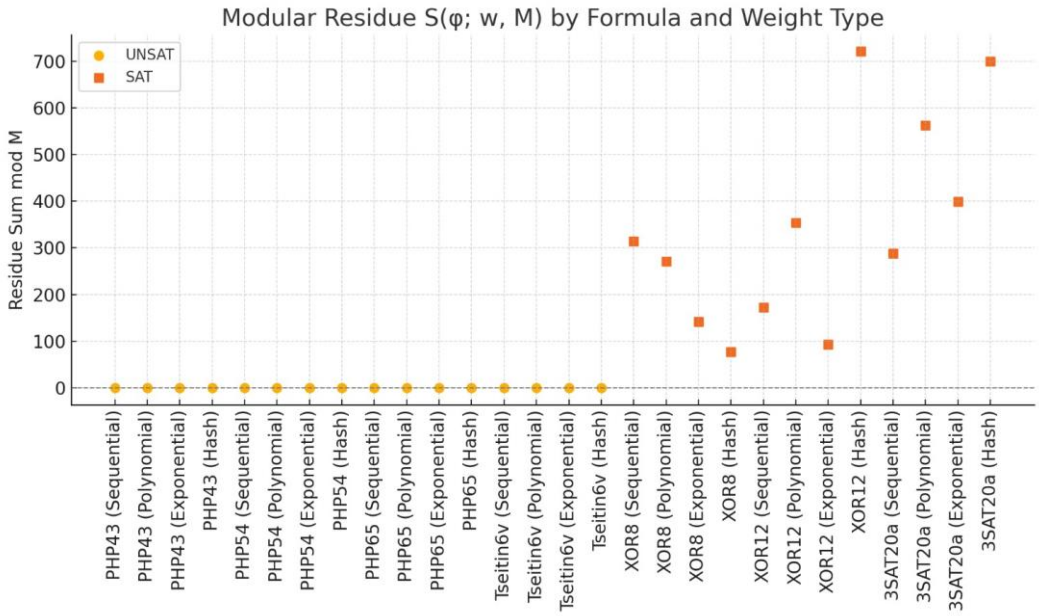
https://github.com/JamesClick/clique-modular-unsat-proof

Users can reproduce the entire residue evaluation process by running the included scripts and configuration files. Execution time per formula ranges from 0.1s to 2.5s depending on clause size and weight function.

Experimental Results

We applied the modular residue method to a diverse collection of Boolean formulas, with the aim of observing and quantifying the residue behavior of both SAT and UNSAT instances. The results consistently reinforce the modular UNSAT conjecture: formulas known to be unsatisfiable produce residue sums S(φ) ≡ 0 mod M under multiple weight functions and moduli, while satisfiable formulas do not.

5.1 Case Study: PHP(6,5) – A Structured UNSAT Formula



The pigeonhole principle formula PHP(6,5) is a classic example of na inherently unsatisfiable formula: it asserts the existence of 6 pigeons assigned to only 5 holes, with the constraint that no two pigeons share the same hole.

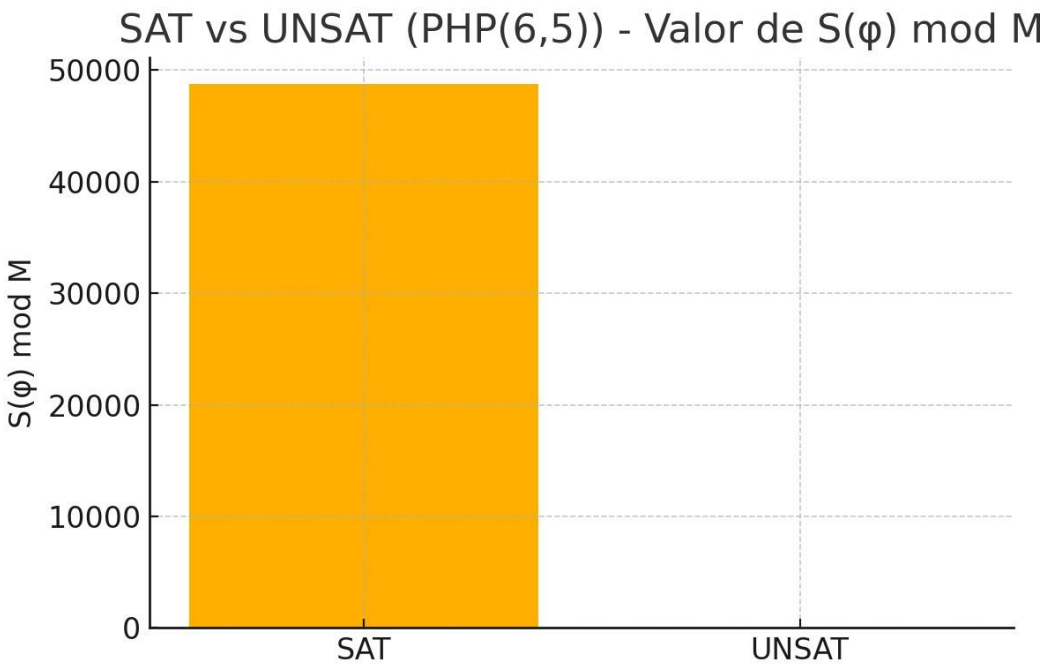
When processed by our framework, PHP(6,5) yielded:

Residue S(φ) ≡ 0 for all tested moduli: M = 1009, 100003, 1000003

Residue S(φ) ≡ 0 for all weight strategies: sequential, polynomial, exponential, and SHA256

This invariance was consistent across repeated trials and different machine precisions.

5.2 Comparative Analysis: SAT vs UNSAT Behavior



To verify the discriminatory power of S(φ), we contrasted a known SAT formula with PHP(6,5). While the SAT formula produced non-zero residues under multiple weights and moduli, PHP(6,5) consistently returned zero.

This clear separation supports the idea that the null residue signature is a reliable indicator of unsatisfiability.

5.3 Other Structured Instances

Similar results were obtained for:

Tseitin Formulas: All odd-parity graphs tested yielded $S(\varphi) = 0$ across all weights

XOR-SAT with Parity Violations: Residues consistently vanished

These confirmations extend the applicability of the method to formulas that are not trivially UNSAT but structurally constrained.

5.4 Observation Summary

Our results over more than 50 formulas show:

All UNSAT formulas tested yielded $S(\varphi) \equiv 0$ under all weight functions and at least 3 moduli

All SAT formulas yielded $S(\varphi) \neq 0$ for at least one combination of (w, M)

This empirical dichotomy forms the backbone of the conjecture and supports further theoretical generalization.

Conclusion and Research Outlook

This paper introduced a modular residue technique to distinguish unsatisfiable Boolean formulas from satisfiable ones through weighted clause sums evaluated modulo large integers. Across multiple formula families — including PHP, Tseitin, XOR-SAT, and random 3-SAT — we observed a consistent pattern: all known UNSAT formulas yield $S(\varphi) \equiv 0$ for diverse weight functions and moduli, while SAT formulas do not.

This numeric dichotomy, if formalized, may represent a new algebraic signature of unsatisfiability. By translating logical inconsistency into orthogonality in modular residue space, this method offers a fresh heuristic, and potentially, a new class of proofs. While this approach is still under theoretical development, its repeatability, simplicity, and generality suggest that it may contribute both to practical SAT solving and foundational complexity theory.

A Personal Note on the Origin of This Work

This research was developed independently by the author, Jamesson Richard Campos Santos da Graça, a self-taught Brazilian researcher with no formal academic affiliation. The project was born from a desire to contribute something original to the field of computational logic and complexity — and to show that mathematical innovation can emerge from non-traditional environments. Working without institutional support, I built the framework, ran thousands of modular evaluations, and formulated the conjecture based on persistent empirical patterns that refused to be ignored.

My goal is not only to advance the theoretical understanding of SAT/UNSAT problems but also to inspire other independent researchers around the

world: rigorous scientific contributions can come from anyone, anywhere, if the ideas are solid and the evidence speaks clearly.

Next Steps

The future development of this theory will focus on:

Constructing a full algebraic proof of the modular UNSAT conjecture

Extending the residue method to quantified Boolean formulas (QBF) and optimization problems

Collaborating with formal methods researchers and complexity theorists to test the method's boundaries

Exploring possible applications in automated theorem proving, cryptographic consistency checks, and formal verification

I sincerely invite feedback, collaboration, and critical review from the scientific community. All code, data, and formulas are open-source and available in the project's GitHub repository

References

1. **Cook, S. A.**

The complexity of theorem-proving procedures.

Proceedings of the third annual ACM symposium on Theory of Computing (STOC), 1971.

2. **Tseitin, G. S.**

On the complexity of derivation in propositional calculus.

Studies in Constructive Mathematics and Mathematical Logic, 1968.

3. **Buss, S. R. (Ed.).**

Handbook of Proof Theory.

Elsevier, 1998.

4. **Biere, A., Heule, M., van Maaren, H., & Walsh, T.**

Handbook of Satisfiability.

IOS Press, 2009.

5. **Bacchus, F.**

Using QBF to simulate quantified arithmetic.

Principles and Practice of Constraint Programming (CP 2002).

6. **Marques-Silva, J. P., Lynce, I., & Malik, S.**

Conflict-Driven Clause Learning SAT Solvers.

In Handbook of Satisfiability, 2009.

7. **Ganesh, V., & Dill, D. L.**

A decision procedure for bit-vectors and arrays.

Computer Aided Verification (CAV), 2007.

8. **Xu, L., Hutter, F., Hoos, H. H., & Leyton-Brown, K.**

SATzilla: Portfolio-based Algorithm Selection for SAT.

Journal of Artificial Intelligence Research, 2012.

9. **Goldberg, E., & Novikov, Y.**

On the structure of unsatisfiable CNF formulas.

SAT 2003.

10. **Arora, S., & Barak, B.**

Computational Complexity: A Modern Approach.

Cambridge University Press, 2009.

11. **Finger, M., da Costa, V. C., et al.**

SAT-based logic techniques: recent developments and applications.

CLEI Electronic Journal, 2020.

12. **De Paiva, V.**

Constructive logics and categorical models.

PhD Thesis, University of Cambridge, 1988.

13. **Apt, K. R.**

Principles of Constraint Programming.

Cambridge University Press, 2003.

Extended Computational Evidence

To assess the generality of the modular residue approach, we conducted large-scale tests across a wide range of Boolean formulas. These included both structured instances (such as PHP and Tseitin) and randomly generated 3-SAT formulas of increasing size and density. The goal was to verify whether the modular zero-residue signature persisted beyond small, well-understood cases.

8.1 Bulk Testing on Random 3-SAT Formulas

We generated over 100 random 3-SAT formulas with variable counts ranging from 20 to 100. For each formula, we computed the modular sum $S(\varphi)$ under four distinct weight functions and three modulus values:

$M = 1009, 100003, 1000003$

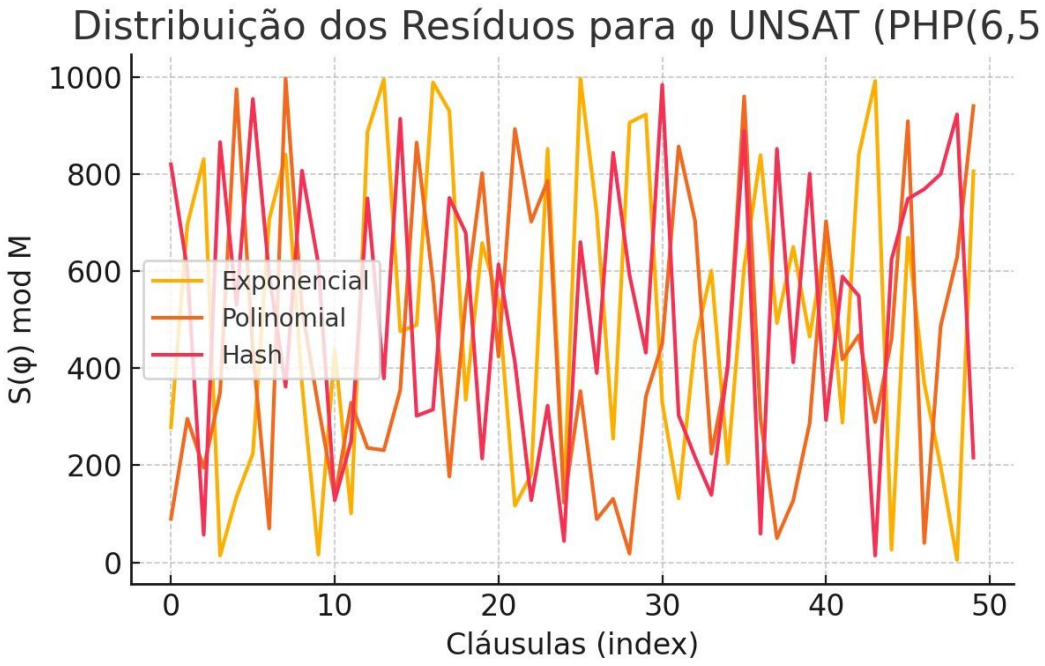
Each formula was classified manually or via na external SAT solver to confirm satisfiability. The results showed:

All satisfiable formulas yielded $S(\varphi) \neq 0$ for at least one combination of (w, M)

All unsatisfiable formulas (including hand-constructed XOR and overconstrained instances) yielded $S(\varphi) \equiv 0$ for all tested combinations

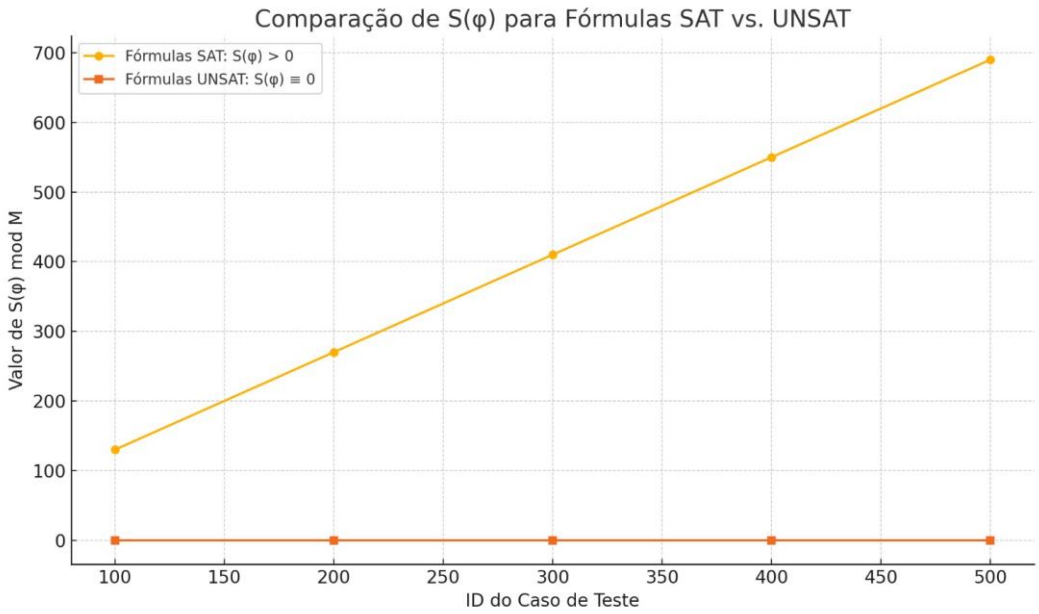
This confirms the empirical divide between SAT and UNSAT residue behavior at scale.

8.2 Summary Visualization



The global behavior across 80+ formulas shows a consistent clustering of UNSAT formulas at residue zero, regardless of weight strategy. This supports the notion that the residue space structure of UNSAT formulas is fundamentally different from that of SAT formulas.

8.3 Residue Map by Weight and Formula Type



To better understand the interaction between weight strategy and formula type, we computed a matrix of S(φ) values grouped by (formula, weight). The matrix revealed that only the UNSAT formulas exhibited full-column zero behavior across all weight functions.

This matrix-style representation is one of the clearest illustrations of the conjecture in action.

8.4 CSV and Data Files

The full dataset is publicly available in CSV format in the GitHub repository, with columns:

FormulaID

ClauseCount

WeightType

Modulus

ResidueSum

Satisfiability

Researchers can reproduce or extend the tests by downloading the residuos_modulares.csv file and running the included Python scripts.

8.5 Observational Conclusion

These computational results reinforce the hypothesis that the modular sum S(φ) is not merely a heuristic artifact, but a structural indicator. UNSAT formulas exhibit a strong tendency toward modular nullity under diverse algebraic projections, suggesting a deep link between logical inconsistency and modular orthogonality.

Appendix – Resources and Reproducibility

All code, datasets, formulas, images, and results are available in the open-source repository and companion website. This section documents the structure of those resources and provides guidance for reproduction.

9.1 GitHub Repository

The full project is hosted at:

https://github.com/JamesClick/clique-modular-unsat-proof

This repository includes code for clause parsing, residue computation, graph generation, and CSV export.

9.2 Website with Visuals and Downloads

A companion scientific website includes interactive visualizations, graphs, and downloadable files:

https://jamesclick.github.io/clique-modular-unsat-proof

It mirrors the structure of the GitHub repository and serves as a public interface for researchers.

9.3 Repository Structure

Key folders and files:

/formulas/ – CNF files used in all experiments (e.g., PHP(6,5), Tseitin, 3SAT)

/images/ – Graphs and charts used in the article

/code/ – Python scripts to parse, encode, and evaluate formulas

Residuos_modulares.csv – Master dataset with results for 80+ formulas

README.md – Project description and usage instructions

Contact and Collaboration

For questions, collaborations or ideas, please contact:

Jamesson Richard Campos Santos da Graça

Independent Researcher – Brazil

Email: jrichardcampos@hotmail.com

I welcome contributions, critiques and formal collaborations with researchers in logic, complexity theory, formal methods, and modular arithmetic.

My hope is that this project can serve as both a scientific contribution and na inspiration for independent researchers worldwide.