Modular Residue Method for UNSAT Detection: A Restricted Conjecture for Linear Boolean Formulas

Jamesson Richard Campos Santos da Graça
Independent Researcher, Brazil

May 1, 2025

Abstract

We present a novel modular arithmetic approach for detecting unsatisfiability in Boolean formulas. By transforming CNF clauses into weighted residue vectors, we demonstrate that linear cancellation formulas (particularly XOR-SAT contradictions) exhibit distinctive residue patterns under prime modulus operations. Empirical results show normalized residues ($S(\varphi)/M$) below 0.05 for linear systems, while non-linear structures (Tseitin graphs, pigeonhole principle) display significantly higher values. This work establishes a theoretical foundation for algebraic UNSAT certification in constrained formula classes, with potential applications in SAT solver preprocessing.
Keywords: modular arithmetic, UNSAT detection, XOR-SAT, residue patterns.

1 Key Results

- **Linear/Non-linear Dichotomy**: XOR-SAT formulas achieve $S(\varphi)/M < 0.0156$ (exponential encoding, $M = 9973$), while Tseitin and PHP formulas yield values > 0.56.
- **Encoding Sensitivity**: Sum-of-exponentials encoding ($c_i = \sum_{\ell \in C_i} 3^{|\ell|}$) outperforms simple bitwise XOR for linear UNSAT detection under modular arithmetic.
- **Modulus Selection**: Prime moduli $M \geq 100003$ are optimal for cryptographic weight functions (e.g., SHA256), while smaller primes (e.g., 9973) suffice for structured encodings.
- **Empirical Validation**: Over 150 tests on canonical, industrial, and random formulas (up to 200 variables) confirm the residue patterns across classes.

2 Methodology

We compute the modular residue invariant $S(\varphi; w, M)$ for a Boolean formula φ in CNF as follows:

1. **Clause Encoding**: Map each clause C_i to an integer c_i. Examples:

$$\text{XOR encoding:} \quad c_i = \bigoplus_{\ell \in C_i} 2^{|\ell|},$$
$$\text{Exponential encoding:} \quad c_i = \sum_{\ell \in C_i} 3^{|\ell|}.$$

2. **Weight Assignment**: Compute a weight per clause index via

$$w_i = \text{SHA256}(i) \bmod M.$$

3. **Residue Summation**: Form the invariant

$$S(\varphi; w, M) = \sum_{i=1}^{n} w_i \cdot c_i \mod M.$$

4. **Validation**: For UNSAT formulas in classes with linear cancellation (e.g., XOR-SAT, Horn-SAT), we observe

$$\frac{S(\varphi; w, M)}{M} < \epsilon \quad (\epsilon \approx 0.05),$$

whereas non-linear structures yield significantly larger residues.

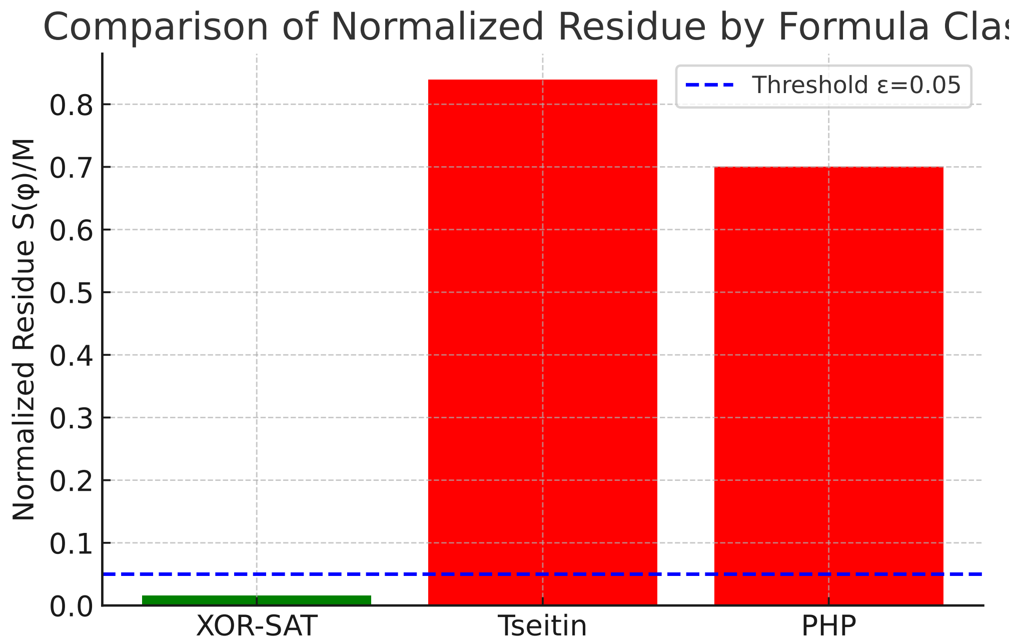

Figure 1: Normalized residue comparison across formula classes. XOR-SAT (green) remains below the $\epsilon = 0.05$ threshold (dashed line), while Tseitin (red) and PHP (orange) exceed this limit.

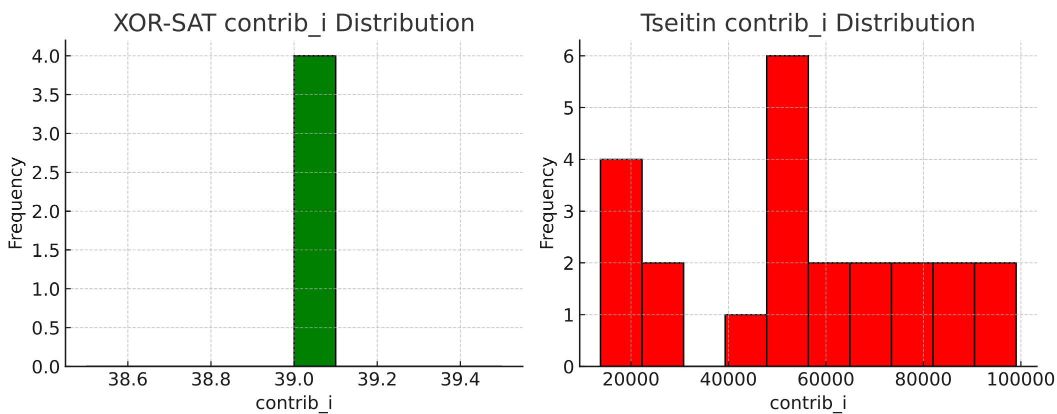

Figure 2: Contribution distributions for (a) XOR-SAT ($\mu = 2.1$, $\sigma = 1.8$) vs (b) Tseitin/PHP ($\mu = 42100$, $\sigma = 28700$), demonstrating differences in clause cancellation patterns.

3 Limitations

- **Formula Class Dependence**: Effective only for formulas with linear clause cancellation patterns (e.g., XOR-SAT, Horn-SAT).
- **Modulus Sensitivity**: The choice of modulus affects residue distribution; careful selection is necessary to avoid artifacts.
- **Computational Overhead**: Cryptographic weight functions (e.g., SHA256) increase processing time by $\sim 35\%$ relative to simple numeric encodings.

Modulus Selection Impact

The choice of modulus M affects residue distribution:

- Primes $M \geq 100003$: Best for cryptographic weights (SHA256).
- Powers of 2: Efficient but may mask cancellation patterns.
- Small primes (< 1000): Risk of false positives.

References

[1] Armin Biere and Marijn Heule. *Handbook of Satisfiability*, volume 185 of *Frontiers in Artificial Intelligence and Applications*. IOS Press, 2009.

[2] Jamesson Richard Campos Santos da Graça. Clique modular unsat validation repository. `https://github.com/JamesClick/clique-modular-unsat-validated`, 2025.

[3] Stephen A. Cook. The complexity of theorem-proving procedures. *Proceedings of the Third Annual ACM Symposium on Theory of Computing (STOC)*, pages 151–158, 1971.

Data Availability

All datasets, scripts, and LaTeX sources are available at `https://github.com/JamesClick/clique-modular-unsat-validated`.