ON THE GALOIS EXTENSIONS OF METABOOLEAN FIELDS

Abstract. We introduce a field-theoretic framework for studying Boolean functions through pointwise operations over GF(2). These structures, which we term metaboolean fields, have unusually degenerate field properties. However, these properties, when interpreted in terms of partial recursive function computation, provide an unexpected wealth of deep connections. We establish that polynomial-time computable Boolean functions form a metaboolean field F_B under these operations, containing a proper subfield F_S of polynomial-time computable symmetric functions. The main result demonstrates that F_B/F_S is a Galois extension with Galois group isomorphic to S_n for n-variable functions. We prove that division in F_B is computationally equivalent to Boolean satisfiability (SAT), while all field operations in F_S, including division, are P-complete. We this work end by speculating as to the implications of our results, which could be potentially an exciting new approach to unsolved problems if our results stand up to wider scrutiny.

1. Introduction

The relationship between algebraic structure and computational complexity has been a rich source of insights in theoretical computer science. A seminal example is Valiant's work [1] showing that polynomial evaluation captures fundamental aspects of computation through projections. Building on this foundation, Skyum and Valiant [2] developed a complexity theory based on Boolean algebra that established tight connections between natural computational problems and algebraic reducibility.

The importance of symmetric Boolean functions in complexity theory was highlighted by Geréb-Graus et al. [4], who proved completeness properties of symmetric Boolean functions under certain reductions. The collapse of algebraic independence in Boolean symmetric polynomials was studied by Zhegalkin[5], and are now known as Zhegalkin polynomials.

Our work extends the basic manipulations of Boolean functions in these works in a novel new direction. Specifically, we examine a deceptively simple structure: the field of Boolean functions under pointwise operations over GF(2). While this field structure appears elementary—the elementwise AND and XOR operations seem to offer little useful information—we show that its division operation provides a pure algebraic abstraction of a variety of unbounded search problems.

The key insight is that in this field, a function has a multiplicative inverse if and only if it maps at least one input to 1. This division operation is so degenerate that its domain check is the only interesting feature, perfectly isolating the fundamental structure of existential search problems. A function's divisibility in a metaboolean

2020 *Mathematics Subject Classification.* Primary 12F10, 68Q15; Secondary 20B30, 06E30, 12E20, 13B05, 68Q17, 05A10, 11T06.

Key words and phrases. Galois theory, Boolean functions, computational complexity, finite fields, symmetric groups.

field becomes equivalent to the existence of a satisfying input, providing a natural bridge between field theory and search problems in their most general form.

When restricted to polynomial-time computable functions, this correspondence specializes to show that field division is equivalent to CNF Boolean satisfiability (SAT). This feels like a natural setting as well with fields generalize arithmetic, which is composed of polynomial time operations. However, the underlying algebraic structure is even more fundamental: it captures existential search independent of computational complexity considerations. This suggests that our field structure could serve as the foundation for an algebraic theory of generalized inversion, where the degenerate nature of division actually serves to isolate the essential character of existence-of-inverse checks.

Our main contributions are:

1. We note the existence of a peculiar field structure we call the metaboolean field.
2. We show that polynomial-time computable Boolean functions form a field F_B under pointwise operations, with polynomial-time computable symmetric functions forming a subfield F_S.
3. We establish that division in F_B is computationally equivalent to Boolean satisfiability (SAT), providing a novel connection between field operations and NP-complete problems.
4. We prove that F_B/F_S is a Galois extension with Galois group isomorphic to S_n for n-variable functions, revealing deep structural properties of Boolean computation.
5. Using this Galois correspondence, we conjecture about the relationship between computational complexity and the algebraic structure of Boolean functions.

These results bridge classical field theory with computational complexity in a way that suggests new approaches to fundamental problems in both areas. The field-theoretic perspective provides a an exciting new method for analyzing partial functions algebraically.

Our approach draws inspiration from the constructive methods used in finite field theory [3], particularly in the analysis of field extensions of GF(2). However, unlike traditional finite field theory where GF(2^n) is studied as an extension of GF(2), we consider the entire space of Boolean functions as our field, leading to qualitatively different structural properties.

2. Structure of Metaboolean Fields

2.1. **Boolean Functions over the Two-Element Finite Field.** Let B_n denote the set of all functions $f : \{0,1\}^n \to \{0,1\}$, and let $B = \bigcup_{n\geq 1} B_n$. We begin by establishing the field structure of B under natural pointwise operations.

Definition 2.1. For $f, g \in B_n$, define:

$$(f+g)(x) = f(x) \oplus g(x)$$
$$(f \cdot g)(x) = f(x) \wedge g(x)$$

where $x \in \{0,1\}^n$, $\oplus$ denotes XOR (addition in GF(2)), and $\wedge$ denotes AND (multiplication in GF(2)).

Theorem 2.2. $(B, +, \cdot)$ *forms a field.*

Proof. We verify each field axiom:

(1) $(B, +)$ is an abelian group:
- Closure: For $f, g \in B_n$, $f+g \in B_n$ since GF(2) is closed under addition
- Associativity: $((f+g)+h)(x) = f(x) \oplus g(x) \oplus h(x) = (f+(g+h))(x)$
- Identity: The constant-0 function
- Inverse: Every function is its own additive inverse since $a \oplus a = 0$ in GF(2)
- Commutativity: $f(x) \oplus g(x) = g(x) \oplus f(x)$

(2) $(B \setminus \{0\}, \cdot)$ is an abelian group:
- Closure: For non-zero $f, g \in B_n$, $f \cdot g \in B_n$ since GF(2) is closed under multiplication
- Associativity: $((f \cdot g) \cdot h)(x) = f(x) \wedge g(x) \wedge h(x) = (f \cdot (g \cdot h))(x)$
- Identity: All non-zero elements are the identity
- Commutativity: $f(x) \wedge g(x) = g(x) \wedge f(x)$
- Inverse: Every non-zero element is its own inverse

(3) Distributivity: For all $f, g, h \in B$,

$$(f \cdot (g+h))(x) = f(x) \wedge (g(x) \oplus h(x)) = (f(x) \wedge g(x)) \oplus (f(x) \wedge h(x)) = ((f \cdot g)+(f \cdot h))(x)$$

□

2.2. **Metaboolean Fields.** Together, $(B, +, \cdot)$ induce a field structure we call a ***metaboolean field***, so named as it partitions Boolean functions into the two field elements:

- *Invertible functions*: Those that map at least one input to 1
- *Non-invertible functions*: Those that map all inputs to 0

The ring operations in GF(2) can be thought of as arithmetic on the parity of numbers. Thus there is a natural mapping of the zero-constant Boolean functions onto *even* and every other function onto *odd*. This has interesting parallels to the Feit-Thompson theorem, but exploration of that can be left for a later date.

Definition 2.3. A *metaboolean field* is a field of Boolean functions over GF(2) as described above.

What is truly remarkable about this field is perhaps how useless it seems at first glance. Yes, we can map non-zero Boolean functions to 1 and zero Boolean functions to 0 and do basic arithmetic on these in $\mathbb{Z}_2$, but what does that really tell us? XOR tells us if we have exactly one of each, and AND tells us if they are both nonzero, but this seems to be rather uninteresting.

We should pause, for a moment, to reflect on the fact that this structure is even a field at all. This arises entirely from the self-inverse property of polynomials over GF(2).

Theorem 2.4. *Let $R_n = \mathrm{GF}(2)[x_1, \ldots, x_n]/\langle x_1^2 - x_1, \ldots, x_n^2 - x_n\rangle$ and let B_n be the field of n-variable Boolean functions under pointwise operations. Then:*

(1) *The evaluation map* $\mathrm{ev}\colon R_n \to B_n$ *defined by* $\mathrm{ev}(p)(a) = p(a)$ *for* $a \in \{0,1\}^n$ *is a field isomorphism.*

(2) *For any field $F \neq \mathrm{GF}(2)$, $F[x_1, \ldots, x_n]/\langle x_1^2 - x_1, \ldots, x_n^2 - x_n\rangle$ is not a field.*

Proof. For (1), ev preserves field operations and is bijective as every Boolean function has a unique multilinear representation.

For (2), when $|F| > 2$, the polynomial $x_1 + 1$ has no multiplicative inverse under pointwise evaluation. □

Corollary 2.5. *When an element in B has a multiplicative inverse, it is equal to itself (in B).*

2.3. **Division Properties.** Given the existence of this unique field structure for this particular family of polynomials (and, interchangeably, functions) we might be inclined to investigate the division operation, as it is perhaps most arithmetically expressed in fields. When we do so, however, we see that division is defined for any non-zero Boolean function, and...it sends that function to itself. Thus, the boring identity operation. Again, it does not seem that our field operations do much of anything. If even the rare operation of division cannot save our metaboolean field from ignominy, we might conclude, as others have hitherto, that there is nothing here worth investigating.

2.4. **Checking for Division.** Upon closer inspection, division is not *exactly* identical to the identity operation for the nonzero element in our metaboolean field. This is because division must respect a very specific rule: we cannot divide by the zero element. We must check *beforehand* to see if the operation is defined. It is this check that gives our metaboolean its interesting properties. Notably, the identity operation, when treated as division, having been performed, serves as a witness to the non-zeroness of the element. Moreover, **the check to see if division is defined in the metaboolean field acts as a hidden operation that has incredibly useful properties relating it to partial functions and computation.** Formally:

Theorem 2.6. *For any $f \in B$, the following are equivalent:*

1. *f has a multiplicative inverse*
2. *f is not identically zero*
3. *There exists x such that $f(x) = 1$*

Proof. ($1 \Rightarrow 2$): If f has inverse g, then $f \cdot g = 1$, so f cannot be identically zero.

($2 \Rightarrow 3$): If f is not identically zero, then there must exist some x with $f(x) = 1$.

($3 \Rightarrow 1$): If $f(x) = 1$ for some x, then f is its own multiplicative inverse. This follows from the the corollary of Theorem 2.4. □

Thus, our degenerate division construction finally reveals, behind multiple layers of trivial and degenerate behavior, an enormously powerful tool that we can use to algebraically study invertibility.

2.5. **The Symmetric Subfield.** We now introduce a subfield of B that will become useful to use later. As the study of symmetry groups is a natural motivating problem for field theory, we will introduce the symmetric subfield of B. Let $S_n \subseteq B_n$ denote the subset of symmetric Boolean functions on n variables, and let $S = \bigcup_{n \geq 1} S_n$.

Theorem 2.7. *S forms a subfield of B.*

Proof. We verify closure under field operations:

(1) Let $f, g \in S$. For any permutation π of inputs and any $x = (x_1, \ldots, x_n)$:

$$\begin{aligned}(f+g)(x_1,\ldots,x_n) &= f(x_1,\ldots,x_n) \oplus g(x_1,\ldots,x_n)\\ &= f(x_{\pi(1)},\ldots,x_{\pi(n)}) \oplus g(x_{\pi(1)},\ldots,x_{\pi(n)})\\ &= (f+g)(x_{\pi(1)},\ldots,x_{\pi(n)})\end{aligned}$$

(2) Similarly for multiplication, using $\wedge$ instead of $\oplus$.
(3) The constant functions 0 is symmetric.
(4) The property of being non-zero in the field is symmetric with respect to the inputs.
(5) If $f \in S$ is not identically zero, then its inverse f is also symmetric.

Therefore, S is a subfield of B. □

3. Division and Boolean Satisfiability

3.1. **From Division to Satisfiability.** Theorem 2.6 revealed a profound connection between field division in B and the Boolean satisfiability problem. Recall that a Boolean function f has a multiplicative inverse if and only if there exists some input x such that $f(x) = 1$. This existential condition mirrors the core structure of Boolean satisfiability problems.

3.2. **Polynomial-Time Boolean Functions.** Although we technically could look at any Boolean function, or choose alternate restrictions, such as Boolean circuits, to look at different complexity classes, restricting our functions to be those that are polynomial time is natural. For one, fields, through their generalization of arithmetic, seem like a match for polynomial time operations. Secondly, the exponential size of the domain is, in a sense, diagonal over polynomials (compared, to say, circuits). Finally and perhaps most important, this theory was developed from the analysis of the symmetries of CNF formulae, and thus we are most familiar with this subset of B.

Definition 3.1. A Boolean function family $f = \{f_n\}_{n\in\mathbb{N}}$ where $f_n : \{0,1\}^n \to \{0,1\}$ is polynomial-time computable if there exists a deterministic Turing machine M and a polynomial p such that:

(1) For every $n \in \mathbb{N}$ and $x \in \{0,1\}^n$, $M(\langle 1^n, x\rangle)$ computes $f_n(x)$
(2) M halts within $p(n)$ steps on all inputs of length n
(3) M outputs either 0 or 1

Let F_B denote the set of all polynomial-time computable Boolean function families. As noted above, this restriction to polynomial-time computation aligns naturally with the verification structure inherent in NP problems.

3.3. **Equivalence with CNFSAT.**

Theorem 3.2. *Let $f \in F_B$. The problem of determining whether f has a multiplicative inverse is polynomial-time equivalent to CNFSAT (SAT).*

Proof. Consider any CNF formula ϕ as a Boolean function f_ϕ that evaluates to 1 precisely when its input satisfies ϕ. Clearly $f_\phi \in F_B$ since evaluating a CNF formula can be done in polynomial time. Then:

$$\phi \text{ is satisfiable } \Leftrightarrow f_\phi \text{ has a multiplicative inverse}$$

Conversely, to find f^{-1}, we must first prove that there is an input x such that $f(x) = 1$. We can then safely do nothing to complete our division operation in polynomial time. The polynomial-time computability of f ensures this is an instance of SAT.

This correspondence is bijective and preserves polynomial-time reductions in both directions. □

3.4. Algebraic Statement of NP-completeness.

This equivalence reveals that our division algorithm algebraically exactly captures the notions of NP-completeness.

Theorem 3.3. *For $f \in F_B$:*

1. *If f has an inverse, computing f^{-1} takes polynomial time (since $f^{-1} = f$ and $f \in F_B$)*
2. *Determining whether f^{-1} exists is NP-complete*

Proof. (1) By Corollary 2.5, when f has a multiplicative inverse, $f^{-1} = f$. Since $f \in F_B$, by Definition 3.1 there exists a polynomial-time Turing machine M that computes f. Therefore, computing f^{-1} requires exactly one call to M, which takes polynomial time.

(2) For NP-hardness: By Theorem 3.3, determining existence of f^{-1} is polynomial-time equivalent to SAT.

For membership in NP: Given $f \in F_B$, we can verify that f has an inverse using a certificate x where $f(x) = 1$. Such an x exists if and only if f has an inverse by Theorem 2.6. Verification takes polynomial time since $f \in F_B$. □

4. Division in the Symmetric Subfield

4.1. Structure of Symmetric Functions.

Lemma 4.1. *Let $f\colon \{0,1\}^n \to \{0,1\}$ be symmetric. Then for any $x, y \in \{0,1\}^n$ with $|x|_1 = |y|_1$, $f(x) = f(y)$.*

Proof. Since f is symmetric, for any permutation $\pi \in S_n$,
$f(x_1, \ldots, x_n) = f(x_{\pi(1)}, \ldots, x_{\pi(n)})$. For any x, y with $|x|_1 = |y|_1$, there exists a permutation π such that $y = (x_{\pi(1)}, \ldots, x_{\pi(n)})$. □

Lemma 4.2. *For $f \in F_S$ on n variables, f has a multiplicative inverse if and only if there exists $k \in \{0, 1, \ldots, n\}$ such that f outputs 1 on all inputs of Hamming weight k.*

Proof. (⇒) Let f have a multiplicative inverse. By Theorem 2.6, there exists x such that $f(x) = 1$. Let $k = |x|_1$. By Lemma 4.1, $f(y) = 1$ for all y with $|y|_1 = k$.

(⇐) If f outputs 1 on all inputs of weight k, then there exists x (any input of weight k) such that $f(x) = 1$. By Theorem 2.6, f has a multiplicative inverse. □

4.2. Zhegalkin-SAT.

Just as division in F_B corresponds to SAT, division in F_S corresponds to what we call ZSAT (Zhegalkin SAT): determining whether a symmetric Boolean function, represented as a Zhegalkin polynomial over GF(2), takes the value 1 for some input weight.

Definition 4.3. The ZSAT problem is:

- Input: Description of a polynomial-time computable symmetric Boolean function f

- Output: Whether there exists $k \in \{0, 1, \ldots, n\}$ such that f outputs 1 on inputs of weight k

This version of SAT can be very easily understood as CNF SAT with an additional restriction, that the function must also be symmetric. This can simply be added on as an additional constraint to CNFSAT. This constraint is equivalent to evaluating normal CNF clauses at the n+1 values logically possibly assignable to the elementary symmetric polynomials.

4.3. **Elementary Symmetric Polynomials.**

Definition 4.4. For n variables and $0 \leq k \leq n$, the k-th elementary symmetric polynomial σ_k over GF(2) is:

$$\sigma_k(x_1, \ldots, x_n) = \bigoplus_{1 \leq i_1 < \cdots < i_k \leq n} (x_{i_1} \wedge \cdots \wedge x_{i_k})$$

Lemma 4.5. *For input x with $|x|_1 = w$, $\sigma_k(x) = \binom{w}{k}_2$, where subscript 2 denotes reduction modulo 2.*

Proof. When computing $\sigma_k(x)$, we count subsets of size k from the w ones in x. This count is $\binom{w}{k}$, and in GF(2), only its parity matters. □

Theorem 4.6. *For n-variable symmetric functions over GF(2), the possible output patterns correspond bijectively to rows of Pascal's triangle modulo 2 of length $n+1$.*

Proof. Let f be symmetric. By Lemma 4.1, f is determined by its values on inputs of weights $0, \ldots, n$. By Lemma 4.5, these values must correspond to binomial coefficients modulo 2 from some row of Pascal's triangle. Conversely, each such row defines a valid symmetric function over GF(2). □

Thus, for ZSAT, we simply take any CNF formula from CNFSAT, and only check the n+1 values that correspond to these special assignments from Pascal's triangle.

4.4. ***P*-completeness of F_S.**

Given this correspondence to division in the symmetric Boolean function field, we might wish to now characterize how powerful our notion of polynomial-time symmetric Boolean functions might be. Do we lose any computational power by restricting our functions to only be symmetric? The P-completeness of F_S can be demonstrated with a simple reduction from the Circuit Value Problem.

Definition 4.7. F_S-EVAL is the problem:

- Input: $\langle M, 1^n, x\rangle$ where M computes a symmetric function f in time $p(n)$
- Output: $f(x)$

Lemma 4.8. *Given a circuit C and input x, we can construct in polynomial time a circuit C' such that:*

1. $C'(1^n) = C(x)$
2. $|C'| \leq p(|C|)$ *for some polynomial p*

Proof. **Construction:** Let C be a Boolean circuit with input gates $g_1, \ldots, g_n$ and let $x \in \{0, 1\}^n$.

Step 1: Initialize C' as a copy of C.

Step 2: For each input gate g_i in C':

- If $x_i = 1$: Leave the input gate unchanged

- If $x_i = 0$: Insert a NOT gate after the input gate

Size Analysis:

1. Original circuit C has size s
2. At most n new NOT gates are added
3. Total size bound: $|C'| \leq s + n \leq 2s$
4. Therefore $|C'| \leq p(|C|)$ where $p(x) = 2x$

Correctness Proof: We prove by induction on circuit depth that for any gate g in C, its corresponding gate g' in C' satisfies:

$$g'(1^n) = g(x)$$

Base case: Input gates at depth 0

- If $x_i = 1$: $g'_i(1^n) = 1 = x_i = g_i(x)$
- If $x_i = 0$: $g'_i(1^n) = \neg 1 = 0 = x_i = g_i(x)$

Inductive step: Consider a gate g at depth $d > 0$

- Let $h_1, \ldots, h_k$ be the input gates to g
- By induction: $h'_j(1^n) = h_j(x)$ for all j
- Therefore: $g'(1^n) = g(h'_1(1^n), \ldots, h'_k(1^n)) = g(h_1(x), \ldots, h_k(x)) = g(x)$

Time Analysis: The construction takes time $O(|C|)$:

1. Copying original circuit: $O(|C|)$
2. Processing input gates: $O(n)$
3. Total time: $O(|C|)$

Therefore, we have a polynomial-time construction of C' that:

1. Correctly simulates $C(x)$ on input 1^n
2. Has size linear in $|C|$

This transformation provides the foundation for our symmetric function constructions by allowing us to convert arbitrary circuit computations into equivalent computations on the all-ones input. □

Theorem 4.9. *F_S-EVAL is P-complete.*

Proof. First, we show F_S-EVAL $\in$ P. Given input $\langle M, 1^n, x\rangle$:

1. Compute $k = |x|_1$ in time $O(n)$
2. Simulate M on input $\langle 1^n, k\rangle$ for $p(n)$ steps
3. Output M's result

Total runtime is $O(n + p(n))$, which is polynomial.

For P-hardness, we reduce from the Circuit Value Problem (CVP). Given a circuit C and input x:

Construction of symmetric function f_C:

1. Let $n = |x|$
2. Apply Lemma 4.8 to obtain circuit C' where $C'(1^n) = C(x)$
3. Define $f_C : \{0,1\}^n \to \{0,1\}$ as follows:

$$f_C(y) = \begin{cases} C'(y) & \text{if } |y|_1 = n \\ 0 & \text{otherwise} \end{cases}$$

Correctness:

- f_C is symmetric: For any y_1, y_2 with $|y_1|_1 = |y_2|_1$:
  - If $|y_1|_1 < n$: $f_C(y_1) = f_C(y_2) = 0$

  - If $|y_1|_1 > n$: $f_C(y_1) = f_C(y_2) = 0$
  - If $|y_1|_1 = n$: $y_1 = y_2 = 1^n$, so $f_C(y_1) = f_C(y_2) = C'(1^n)$
- f_C is polynomial-time: Given y:
  - Computing $|y|_1$ takes $O(n)$ time
  - If $|y|_1 = n$, evaluating C' takes polynomial time
  - Otherwise, outputting 0 takes $O(1)$ time
- The reduction preserves answers:

$$\begin{aligned} C(x) = 1 &\iff C'(1^n) = 1 \text{ (by Lemma 4.8)} \\ &\iff f_C(1^n) = 1 \text{ (by construction)} \\ &\iff \langle M_{f_C}, 1^n, 1^n \rangle \text{ evaluates to } 1 \end{aligned}$$

  where M_{f_C} is the Turing machine computing f_C

Time Analysis:

(1) Converting C to C' takes polynomial time (Lemma 4.8)
(2) Constructing description of M_{f_C} takes polynomial time:
  - Counter for computing $|y|_1$: $O(n)$ size
  - Circuit C': polynomial size
  - Control logic: constant size
(3) Total reduction time is polynomial in $|C| + |x|$

Therefore, F_S-EVAL is P-complete under polynomial-time reductions. The reduction is parsimonious, preserving both the answer and the computational structure of the original circuit.

□

4.5. **ZSAT Complexity.** Given the P-completeness of F_S, it is not necessarily surprising, then that ZSAT is also P-complete. The time complexity of the division operation for the symmetric metaboolean field is then, closed, in the sense that the division operation, as polynomial time function, exists as an element of F_S. Notably, the division operation does seem to obey the Time Hierarchy, with F_S division always being a degree higher than the function that is being divided.

Theorem 4.10. *ZSAT is P-complete.*

Proof. First, we show ZSAT $\in$ P. Given input $\langle M, 1^n \rangle$:

(1) For each $k \in \{0, \ldots, n\}$:
  - Let x_k be the lexicographically first string with $|x_k|_1 = k$
  - Simulate M on $\langle 1^n, k \rangle$ for $p(n)$ steps
  - If M accepts, return YES
(2) If no value of k leads to acceptance, return NO

This algorithm:

- Makes $n + 1$ evaluations of M
- Each evaluation takes $p(n)$ time
- Total runtime is $O((n + 1)p(n))$, which is polynomial

For P-hardness, we reduce from CVP. Given circuit C and input x:

Construction Phase:

(1) Let $n = |x|$
(2) Find smallest k such that $2^k - 1 \geq n$:
  - $k = \lceil \log_2(n + 1) \rceil$

- This ensures $2^k - 1$ is the next "full" level size above n

(3) Pad C to size $2^k - 1$:
- Add at most $2^k - 1 - n$ dummy AND gates with constant inputs
- Connect them in a binary tree structure
- Note: $2^k - 1 - n < 2n$ since $k \leq \log_2(n+1) + 1$

(4) Apply Lemma 4.8 to obtain C' where $C'(1^k) = C(x)$

(5) Construct Turing machine M_C that computes:

$$f_C(y) = \begin{cases} C'(1^k) & \text{if } |y|_1 = k \\ \binom{|y|_1}{j}_2 & \text{otherwise} \end{cases}$$

where j is chosen based on $C'(1^k)$ as follows:
- If $C'(1^k) = 1$: Choose j such that $\binom{k}{j}_2 = 1$
- If $C'(1^k) = 0$: Choose j such that $\binom{k}{j}_2 = 0$

Correctness of Construction:

(1) The padding ensures:
- Circuit size is exactly $2^k - 1$ for some $k \leq \log_2(n+1) + 1$
- Original circuit behavior is preserved
- Sufficient structure for Pascal patterns

(2) By Lucas' Theorem, for prime p:

$$\binom{n}{j}_p = \prod_{i=0}^{m} \binom{n_i}{j_i}_p$$

where n_i, j_i are base-p digits

(3) For $p = 2$, this means:
- $\binom{k}{j}_2 = 1$ iff the binary expansions satisfy
- $(k_i)_2 \geq (j_i)_2$ for all i
- No "carries" occur in binary subtraction $k - j$

(4) Therefore:
- When $C'(1^k) = 1$: Choose j with binary expansion ensuring $\binom{k}{j}_2 = 1$
- When $C'(1^k) = 0$: Choose j with binary expansion ensuring $\binom{k}{j}_2 = 0$
- Such j always exists by properties of Pascal's triangle mod 2

(5) The resulting function f_C:
- Is symmetric (depends only on $|y|_1$)
- Matches a row of Pascal's triangle mod 2
- Has $f_C(1^k) = C'(1^k) = C(x)$

Construction of M_C:

(1) On input $\langle 1^n, w \rangle$:
- If $w = k$: Evaluate $C'(1^k)$
- Otherwise: Compute $\binom{w}{j}_2$ using Lucas' Theorem

(2) Implementation details:
- Counter for tracking position: $O(\log n)$ bits
- Binary arithmetic for Lucas' Theorem: $O(\log n)$ bits
- Circuit evaluation space: $O(|C'|)$ bits

Time Analysis:

(1) Computing k: $O(\log n)$ time

(2) Padding circuit: $O(n)$ time since we add ¡ 2n gates

(3) Converting to C': polynomial time (Lemma 4.8)
(4) Computing binomial coefficients mod 2: $O(n \log n)$ time using Lucas
(5) Total reduction time: polynomial in $|C| + |x|$

Verification of Reduction:

$$\begin{aligned} C(x) = 1 &\iff C'(1^k) = 1 \text{ (by Lemma 4.8)} \\ &\iff f_C \text{ has a 1-output at weight } k \\ &\iff \text{ZSAT accepts } \langle M_C, 1^k \rangle \end{aligned}$$

Therefore, ZSAT is P-complete under polynomial-time reductions. □

Corollary 4.11. *The field operations of F_S are P-complete.*

Thus we have characterized the time complexity of some of the field operations in the metaboolean fields. While the structures are simple, the concepts here can be confusing. The elements of the field are functions, and the operations of the field are functions, but the inverse operations of the field do not necessarily have to be elements of the field. Compare logically, here, to the square root.

5. Summary of Initial Results

5.1. **The Two Metaboolean Fields.** So far, we have constructed two metaboolean fields associated with polynomial time computations, F_S, and F_B, with the restriction motivated by the SAT-like structure. Given the existence of these fields and their relationship to each other, it is only natural to biw investigate their Galois group structure. Before we do so, we review the complete complexity of field operations in F_B and F_S.

Theorem 5.1 (Complexity of Field Operations)**.** *The computational complexity of field operations in F_B and F_S is characterized as follows:*

(1) *In F_B:*
   - *Addition and multiplication are polynomial-time computable*
   - *Testing for multiplicative inverse existence is NP-complete*
   - *Computing f^{-1} when it exists is polynomial-time ($f^{-1} = f$)*
   - *Testing equality/zero testing conjectured to be co-NP-complete, to be verified in future work*

(2) *In F_S:*
   - *All operations are polynomial-time computable*
   - *Testing for multiplicative inverse existence is P-complete*
   - *Computing f^{-1} when it exists is polynomial-time ($f^{-1} = f$)*
   - *Testing equality requires only $n + 1$ evaluations*

Proof. The complexity of operations in F_B follows from Theorems 3.2 and 3.3. For F_S, the results follow from Theorems 4.9 and 4.10, noting that symmetric functions are completely determined by their values on inputs of weights 0 through n. □

6. Main Result

6.1. **Extension Structure.** The field structure relationship F_B and F_S is simple and straightforward by construction, as their relationship generalizes the motivating problems of field theory.

Theorem 6.1. *F_B is a Galois extension of F_S.*

Proof. We need to show that F_B is a finite, normal, and separable extension of F_S.

1. F_B is a field extension of F_S:
   - $F_S \subset F_B$
   - The operations in F_B extend those in F_S
2. F_B is algebraic over F_S: Every $f \in F_B$ satisfies the equation $f^2 - f = 0$ in F_S.
3. F_B is a normal extension: For any $f \in F_B$, all roots of its minimal polynomial over F_S are in F_B.
4. F_B is a separable extension: In characteristic 2, $x^2 - x$ has distinct roots for all x.
5. Galois group: $\mathrm{Gal}(F_B/F_S) \cong S_n$ for functions on n variables, where S_n is the symmetric group.
   - The automorphisms of F_B fixing F_S correspond to permutations of input variables.
   - There are $n!$ such automorphisms for n-variable functions.

Therefore, F_B is a Galois extension of F_S. □

Corollary 6.2. *Gal*$(F_B/F_S) \cong S_n$

We have thus arrived at our main result, an algebraic description of the Galois extension F_B over F_S, which results in a rather unsurprising Galois group, given the construction. While the methods thus far have been exceedingly novel, they are also exceedingly simple due to the degenerate structure of the field. The method is generalizable to other schemes of universal partial computation and constitutes a novel structure worth investigating irrespective of this motivating example.

7. Further Conjecture

7.1. **Other Unsolved Problems.**

The general methods here were inspired by, and no doubt bear a relation to some previously solved open problems. Notably, the Abel-Ruffini Theorem, which forbids a general root formula in terms of the coefficients of a polynomial, and construction proofs from geometry. We speculate that the main result, with a little effort, leads one to the conclusion that P is not NP, which would solve one of the major current open problems.

Conjecture. *The main result has major implications for $P \neq NP$.*

We envision a proof looking something like the following:

Proof. (Sketch.) The sketch proceeds by analyzing the computational complexity of determining multiplicative inverse existence in F_B through its Galois correspondence with permutation groups.

Lemma 7.1 (Structural Correspondence)**.** *For n-variable functions in F_B, there exists a Galois correspondence such that:*

1. *Gal*$(F_B/F_S) \cong S_n$
2. *The fixed field of S_n is precisely F_S*
3. *Invertible functions correspond to odd permutations*
4. *Non-invertible functions correspond to even permutations*

For $n > 4$, the alternating group A_n has the following properties:

1. A_n is simple

(2) $S_n/A_n \cong \mathbb{Z}_2$
(3) $|A_n| = n!/2$

Let A be any algorithm that determines invertibility in F_B. By the Galois correspondence, A must effectively determine whether an element of S_n is odd or even through computations in the ground field F_S.

Lemma 7.2 (Computational Lower Bound)**.** *Any algorithm determining parity in S_n via operations in F_S must evaluate at least $n!/2 + 1$ field operations.*

Proof. We proceed by contradiction. Suppose algorithm A uses fewer than $n!/2+1$ operations. Let G be the subgroup of S_n generated by the sequence of operations performed by A. Consider:

(1) By the simplicity of A_n for $n > 4$, any proper subgroup of A_n must have index at least $n!/2$ in S_n
(2) The sequence of elements encountered by A forms a chain:

$$g_1, g_1g_2, g_1g_2g_3, \ldots, g_1g_2\cdots g_k$$

where $k < n!/2 + 1$ and each g_i corresponds to a symmetric function evaluation
(3) By the pigeonhole principle, either:
- The chain fails to reach all elements of A_n, or
- The chain must contain a repeated element

(4) In either case:
- If the chain doesn't reach all elements of A_n, it cannot distinguish odd from even permutations
- If the chain repeats, subsequent operations cycle within a proper subgroup, again failing to distinguish parity

Therefore, at least $n!/2 + 1$ operations are required. □

We now establish complexity bounds:

(1) Each symmetric function evaluation requires polynomial time $p(n)$
(2) The minimum number of required evaluations is $n!/2 + 1$
(3) Total required time is at least $p(n)(n!/2 + 1)$
(4) For any polynomial p, this is superpolynomial in n

By Theorem 3.3, division in F_B is polynomial-time equivalent to SAT. Therefore:

(1) SAT cannot be solved in polynomial time
(2) Hence P $\neq$ NP

□

8. Conclusions

Our methods in this work are exceedingly novel and show some promise for providing deep answers to longstanding questions in mathematics. The claims above will surely draw intense criticism from the wider mathematical community, and rightly so. The connections between computation and algebra could be made more rigorous, and will need to be thoroughly checked for validity. The algebra should stand on its own and represents a novel contribution in its own right, but it is the interpretation with regards to Turing complete computation that should be scrutinized most. Yet, the simplicity and elegance of the math seem nearly too beautiful to ignore in the applied context, which was the motivating concern to begin with.

Should the methods herein withstand scrutiny and gain wider acceptance, then this method represents a new and extremely powerful tool in the mathematician's toolbox. We contend that the mathematics in this work is not necessarily beyond that which would be found in a (a variety of) first-year graduate texts, and thus hopefully should be easily verified and refined by the wider mathematical community. Our main contribution has merely been to notice a very simple, but very obscure structure that has been hiding in plain sight.

References

[1] L. G. Valiant, *Completeness classes in algebra*, Conference Record of the Eleventh Annual ACM Symposium on Theory of Computing (Atlanta, Ga., 1979), ACM, New York, 1979, pp. 249–261.

[2] S. Skyum and L. G. Valiant, *A complexity theory based on Boolean algebra*, J. Assoc. Comput. Mach. **32** (1985), no. 2, 484–502.

[3] R. Lidl and H. Niederreiter, *Finite fields*, Encyclopedia of Mathematics and its Applications, vol. 20, Cambridge University Press, Cambridge, 1997.

[4] M. Geréb-Graus, T. H. Kuo, and D. W. Welsh, *Symmetric Boolean functions*, Theoret. Comput. Sci. **108** (1993), no. 2, 371–390.

[5] I. I. Zhegalkin, *O tekhnike vychisleniy predlozheniy v simvolicheskoy logike* [On the technique of calculating propositions in symbolic logic], Mat. Sbornik **43** (1927), 9–28.