Gröbner Bases for Boolean Function Minimization

Nicolas Faroß[1], Simon Schwarz[2,3]

[1]Saarland University, Saarbrücken, Germany
[2]Max Planck Institute for Informatics, Saarland Informatics Campus, Saarbrücken, Germany
[3]Graduate School of Computer Science, Saarbrücken, Germany

Abstract

Boolean function minimization techniques try to find, for a given formula, a smaller equivalent formula. In this work, we present a novel technique for heuristic boolean function minimization. By using an algebraic encoding, we embed the minimization problem into an algebraic domain, where algorithms for computing Gröbner bases are applicable. A Gröbner basis usually forms a compact representation of our encoded function. From the Gröbner basis, we then reconstruct an equivalent, more compact boolean formula. Our approach is the first to use Gröbner bases for function minimization. Combined with advances of algebraic Gröbner bases in satisfiability checking, this motivates further research on applications of Gröbner bases in the context of boolean logic.

Keywords

Multi-level Logic Optimization, Boolean Function Synthesis, Gröbner Bases

1. Introduction

Boolean function minimization algorithms find, given a boolean formula, a small equivalent boolean formula. This, for example, is a central problem in circuit synthesis [1, 2].

In general, the exact boolean minimization problem is NP-hard, it even is Σ_2^P-complete [3]. Hence, exact approaches quickly become infeasible. Thus, approaches in this area often have trade-offs between result quality and runtime. Previous work often focused on minimizing *disjunctive normal forms* (DNF) or used syntactical heuristics to simplify functions (Section 1.1).

Our contribution is a novel heuristic approach for multi-level boolean function minimization. Given an input formula in DNF, our approach encodes it as a system of equations over $\mathbb{F}_2$, using a modification of previous encodings [4]. This representation allows computing a Gröbner basis with common algorithms [5]. The resulting Gröbner basis, then, can again be interpreted as an empirically smaller, equivalent boolean formula. This process is repeated recursively to further minimize the formula. Contrary to other minimization algorithms, our encoding over $\mathbb{F}_2$ can efficiently represent the exclusive-or ($\oplus$) operator, allowing exponentially more compact representations compared to DNF. Thus, our approach works particularly well for formulas which contain many exclusive-or operations. In practice, such formulas occur, for example, in

8th International Workshop on Satisfiability Checking and Symbolic Computation, July 28, 2023, Tromsø, Norway, Collocated with ISSAC 2023
faross@math.uni-sb.de (N. Faroß); sschwarz@mpi-inf.mpg.de (S. Schwarz)
https://people.mpi-inf.mpg.de/~sschwarz/ (S. Schwarz)
0009-0001-4419-2337 (N. Faroß); 0000-0002-0064-2922 (S. Schwarz)
© 2023 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).
CEUR Workshop Proceedings (CEUR-WS.org)

CEUR-WS.org/Vol-3455/short4.pdf

microarchitectural hash functions [6]. Some proprietary hash functions, e.g. used for cache slice indexing in modern microprocessors have been reversed (i.e. a compact representation has been found) by using our approach (work under submission). In practice, we observe more compact representations than a minimized DNF even on random formulas, Section 3. Furthermore, for some classes of formulas, nearly optimal results are provable (Theorem 2).

In addition to previous applications of Gröbner bases in logic [4, 7] that focus on preprocessing for satisfiability checking, our approach is a novel, complementary application of Gröbner bases for boolean logic. This motivates further research in this direction.

1.1. Related Work

Boolean Minimization Usually, a boolean logic minimizer is given a formula in DNF, which can be easily obtained from a truth table. Then, it produces a small, equivalent formula. Existing tools can be classified by the shape of the output formula:

Two-level logic optimization tools produce again a formula in DNF. Classically, Quine and McCluskey [8, 9] focus on optimal two-level minimization. However, their optimal approach quickly becomes infeasible [1]. Thus, later approaches such as ESPRESSO [1] make use of a heuristic search for producing a compact DNF. Still, representing a formula in DNF can have exponential overhead. For example, any DNF of $f(x_1, \ldots, x_n) = x_1 \oplus \cdots \oplus x_n$ is exponentially larger in n than f. This motivates more general minimization approaches.

Multi-level logic optimization allows minimized formulas of arbitrary shape. Still, some tools produce only formulas of a fixed structure. For example, EXORCISM [10] always produces formulas of the shape "exclusive-sum-of-products". Other tools produce formulas of arbitrary depth and structure, for example MIS [11] or LSS [12]. Most multi-level optimization tools rely on two-level optimization and apply syntactic transformations such as *subexpression recognition* and replacement to the resulting formula [13]. More recently, approaches for multi-level synthesis based on satisfiability [14, 15] have been studied. They provide optimal solutions but are only feasible for small instances. Our presented approach provides a heuristic method for multi-level logic optimization which, contrary to previous work, does not rely on syntactic recognitions.

Gröbner Bases are special generating sets for ideals and are widely used in computer algebra [16], for example for ideal membership testing of a polynomial. However, we focus on applications of Gröbner bases in logic. For example, Gröbner bases are used in the context of satisfiability checking and model counting, as well as in verification and SMT solving.

In satisfiability checking, Gröbner bases can be used for pre-processing clause sets in conjunctive normal form (CNF) [4, 7]. Concretely, it is possible to encode a (sub-)set of clauses of a CNF formula as a system of polynomials. Then, a Gröbner basis for this system is computed. The resulting system of polynomials is then, again, interpreted as a set of clauses. The resulting, usually more compact set of clauses is *equivalent* to the original set. Hence, it is possible to replace sets of clauses with more compact, equivalent sets. Satisfiability checking on the preprocessed clauses is usually faster than on the original CNF [4]. However, the cost of computing the Gröbner bases outweighs the benefits. A similar encoding can be used for model counting for boolean formulas [17]. Given a set of clauses reduced by the above technique, it becomes (computationally) easy to count satisfiable assignments. In contrast to the above approaches,

we apply Gröbner bases to formulas in disjunctive normal form (DNF), and in the context of logic minimization.

Other applications of Gröbner basis are verification of arithmetic gates [18, 19, 20] and SMT solving over finite fields [21, 22] or real numbers [23]. However, note that both applications do not directly encode boolean formulas as polynomials, but use Gröbner bases for theory solving. Still, combined with our work, this suggests that Gröbner bases have multiple applications in logic and could be worthwhile to investigate further.

2. Gröbner Bases for Logic Minimization

Given a boolean DNF $\varphi = \bigvee_i \bigwedge_j y_{ij}$ with literals $y_{ij} \in \{x_1, \dots, x_n, \neg x_1, \dots, \neg x_n\}$, we want to find an equivalent smaller formula. Our idea is to switch from boolean formulas φ to ideals I in the polynomial ring $\mathbb{F}_2[x_1, \dots, x_n]$. Then, Gröbner bases of I correspond empirically to smaller equivalent formulas of φ.

2.1. Encoding Boolean Formulas as Polynomials

In the following, we will identify the truth-values $\{\mathit{false}, \mathit{true}\}$ with the set $\mathbb{F}_2 = \{0, 1\}$, where false corresponds to 0 and true corresponds to 1. Note that $\mathbb{F}_2$ has the structure of a field with multiplication given by $\wedge$ and addition is given by $\oplus$. Hence, we can view a polynomial $f = \sum_i \prod_j y_{ij} \in \mathbb{F}_2[x_1, \dots, x_n]$ with $y_{ij} \in \{1, x_1, \dots, x_n\}$ as a boolean formula of the form $\bigoplus_i \bigwedge_j y_{ij}$. Conversely, we can encode any boolean formula φ as a polynomial over $\mathbb{F}_2$ by additionally using the identities $\neg x = x \oplus 1$ and $x \vee y = \neg(\neg x \wedge \neg y)$, which will be encoded as $x + 1$ and $1 + (1 + x) \cdot (1 + y) = xy + x + y$ respectively. We call this the *algebraic encoding* of φ.

The previous relationship between boolean formulas and polynomials can be further extended to a correspondence between equivalent formulas and ideals in $\mathbb{F}_2[x_1, \dots, x_n]$. Recall that an *ideal* I in a polynomial ring R is a subset $I \subseteq R$ which can be written as $I = \{\sum_{i=1}^m g_i f_i \mid g_1, \dots, g_m \in R\}$ for some polynomials $f_1, \dots, f_m \in R$. Such polynomials are called generators of I, and we write $I = (f_1, \dots, f_m)$.

Theorem 1. *There is a bijection between equivalence classes of boolean formulas $[\varphi]$ and ideals $I \subseteq \mathbb{F}_2[x_1, \dots, x_n]$ containing $x_1^2 + x_1, \dots, x_n^2 + x_n$, such that $\varphi(x) = 0 \iff \forall f \in I, f(x) = 0$.*

Proof. Let $[\varphi]$ be an equivalence class of boolean formulas in n variables. Then it is uniquely determined by its zero set $\{x \in \mathbb{F}_2^n \mid \varphi(x) = 0\}$. Conversely, any set $Y \subseteq \mathbb{F}_2^n$ is the zero set of the formula $\bigwedge_{y \in Y} \bigvee_{i=1}^n (x_i \oplus y_i)$. Hence, there is a bijection between equivalent formulas and subsets of $\mathbb{F}_2^n$. Similarly, any ideal containing $x_1^2 + x_1, \dots, x_n^2 + x_n$ is uniquely determined by its zero set $\{x \in \mathbb{F}_2^n \mid \forall f \in I, f(x) = 0\}$ by Hilbert's Nullstellensatz for finite fields [24]. Additionally, every subset $Y \subseteq \mathbb{F}_2^n$ is finite and hence a zero set of an ideal by elementary results from algebraic geometry. Further, this ideal can always be assumed to contain $x_1^2 + x_1$, $\dots$, $x_n^2 + x_n$, such that we have a bijection between ideals containing $x_1^2 + x_1, \dots, x_n^2 + x_n$ and subsets $Y \subseteq \mathbb{F}_2^n$. By combining this bijection which the first one, we obtain the statement of the theorem. □

Importantly, we can explicitly convert a boolean formula to generators of the corresponding ideal and vice versa. Let $\varphi = \bigvee_i \bigwedge_j y_{ij}$ be a formula in DNF, then its corresponding ideal I is generated by $x_1^2 + x_1, \ldots, x_n^2 + x_n$ and $f_i = \prod_j y_{ij}$, where we identify a literal y_{ij} with a polynomial via our algebraic encoding. This can be verified by using Theorem 1 and observing that $\varphi(x) = 0$ if and only if $f_i(x) = 0$ for all f_i, which is equivalent to $f(x) = 0$ for all $f \in I$. Similarly, an ideal $I = (f_1, \ldots, f_m, x_1^2 + x_1, \ldots, x_n^2 + x_n)$ corresponds to the boolean formula $\bigvee_i f_i$, where the polynomials f_i are again identified with a boolean formula.

Note that the polynomials $x_i^2 + x_i$ represent the idempotency law $x_i \wedge x_i = x_i$ and allow the elimination of all higher powers of x_i during the Gröbner basis computation. Thus, there exist only 2^n different leading monomials besides x_i^2, which implies that the size of a reduced Gröbner basis is bounded by $2^n + n$. Further, it is possible to compute this Gröbner basis in time $2^{O(n)}$ using Buchberger's algorithm, see Proposition 4.1.1 in [17]. This provides a significant improvement compared to the usual double exponential bound for Gröbner bases.

2.2. Gröbner Basis Minimization

Empirically, Gröbner bases are relatively small if the ideal is not too complex. Hence, we simplify a boolean formula by converting it to an ideal I. Then, we compute a Gröbner basis of I to obtain a set of generators G. G is then converted back to a formula of the form $\bigvee_{g \in G} g \cong \bigvee_{g \in G} \neg\neg g$. We can apply this approach recursively to the terms $\neg g$ to further reduce the size and obtain a formula of the form $\bigvee_i \neg \bigvee_j \neg \bigvee_k \ldots \cong \bigvee_i \bigwedge_j \bigvee_k \ldots$, as depicted in the following algorithm:

```
1: function Minimize(φ)
2:     G ← GröbnerBasis(Ideal(φ))
3:     for g ∈ G do
4:         if g is linear or recursion limit is reached then
5:             φ_g ← Formula(g)
6:         else
7:             φ_g ← ¬Minimize(DNF(¬Formula(g)))
8:     return ⋁_{g∈G} φ_g
```

Note that it is not guaranteed that our algorithm produces a smaller formula and by the previous bound the Gröbner bases can have a size exponential in the number of variables. However, empirically we observed very compact formulas. Furthermore, for special classes of formulas, better bounds can be proven.

Theorem 2. *Let φ be a formula in DNF which is equivalent to $\bigvee_i \bigoplus_j A_{ij} x_j$ for a matrix $A \in \mathbb{F}_2^{n \times n}$. Then $|\textsc{Minimize}(\varphi)| < 2 \cdot n \cdot \mathrm{rank}(A)$, where $|\cdot|$ denotes the number of boolean operators and variables in a formula.*

Proof. Consider a boolean formula φ which is equivalent to $\bigvee_{i=1}^n \bigoplus_{j=1}^n A_{ij} x_j$ for a matrix $A \in \mathbb{F}_2^{n \times n}$. Then the corresponding ideal I from Theorem 1 is generated by $x_i^2 + x_i$ and the linear terms $\sum_{j=1}^n A_{ij} x_i$ defined by the rows of A. Denote with m the rank of A and let $A' \in \mathbb{F}_2^{m \times n}$ the be matrix given by the non-zero rows in the reduced row echelon form of

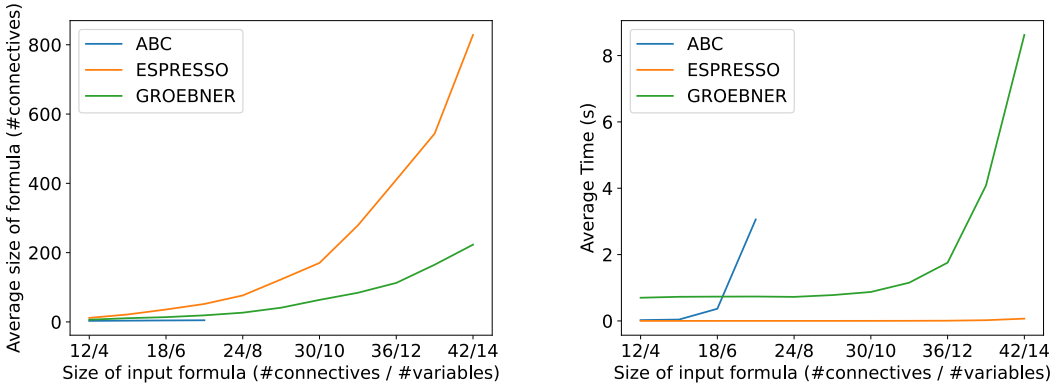

Figure 1: Comparison of output size and runtime.

A, which can be computed using Gaussian elimination. Then one can check that a reduced Gröbner basis of I is given by the terms $\sum_{j=1}^{n} A'_{ij} x_i$ corresponding to the rows of A' and the terms $x_i^2 + x_i$ for which x_i is not a leading monomial of one of the first terms. Since a reduced Gröbner basis is unique, it will also be computed by our minimization algorithm. Further, the terms of the form $x_i^2 + x_i$ are redundant and will be removed, such that a formula of the form $\bigvee_{i=1}^{m} \bigoplus_{j=1}^{n} A'_{ij} x_j$ is returned. By counting the number of operators and variables, we obtain that the size of such a formula is always less than $2nm$. □

Further Improvements In addition to our main algorithm, we implemented the following two improvements: First, minimizing the DNF with a two-level minimization algorithm during preprocessing reduces the size of the input. This does not change result quality, but can significantly speed up the algebraic computations. Second, some elements of the Gröbner basis can be redundant in the sense that a subset of the Gröbner basis already generates the corresponding ideal. Finding a minimal generating subset of the Gröbner basis can be formulated as a weighted set-cover problem, which we solve in a post-processing step to improve result quality. Eliminating these redundant terms directly during the computation of the Gröbner basis is subject to further work.

3. Evaluation

A first prototype of this algorithm has been implemented in SageMath [25], using SINGULAR [26] for computing Gröbner bases, as well as ESPRESSO [1] for DNF preprocessing and GLPK [27] for set-cover computations. Note that Gröbner bases are computed with respect to the degree reverse lexicographic order. A basic benchmark in Figure 1 shows the average size of a reconstructed formula, and the runtime of the minimization process. This is compared to a minimized formula in DNF from ESPRESSO [1] and a minimal formula produced with ABC's 'exact' command [28, 14]. The benchmark ran on 200 random formulas for each size. The

formulas were sampled uniformly at random on a syntactic level, i.e. one syntax tree is sampled uniformly at random out of all possible syntax trees for a specific size. Note that exact, optimal synthesis with ABC is only possible for up to 7–8 variables, while our approach can deal with up to 20 variables. Our resulting formulas are always more compact than a DNF produced by ESPRESSO. However, our approach has a significantly greater runtime. We are currently working on reverse-engineering microarchitectural hash functions used e.g. for cache indexing in processors with our algorithm (to appear in [29]).

4. Conclusion

We have presented a novel technique for heuristic multi-level formula minimization based on an algebraic encoding of formulas and Gröbner basis computations. Currently, our algorithm can handle functions with up to 20 input bits, whereas exact synthesis can only handle up to 8 input bits. Empirically, our approach produces smaller formulas than two-level minimization algorithms such as ESPRESSO. In particular, our idea works well for formulas with many exclusive-or operations. Our algorithm shows a new application of Gröbner basis computations in logic minimization. In addition to previous work on Gröbner bases in satisfiability [4, 7], this result suggests that the use of our algebraic encoding, as well as Gröbner bases, brings benefits to boolean logics and could be worthwhile to investigate further.

Acknowledgements: We thank our anonymous reviewers for their constructive comments.

References

[1] R. K. Brayton, G. D. Hachtel, C. McMullen, A. Sangiovanni-Vincentelli, Logic minimization algorithms for VLSI synthesis, The Springer International Series in Engineering and Computer Science, Springer, 1984. doi:10.1007/978-1-4613-2821-6.

[2] R. L. Rudell, Multiple-valued logic minimization for PLA synthesis, Technical Report, EECS Department, University of California, Berkeley, 1986. URL: http://www2.eecs.berkeley.edu/Pubs/TechRpts/1986/734.html.

[3] D. Buchfuhrer, C. Umans, The complexity of boolean formula minimization, in: Automata, Languages and Programming, Springer, 2008, pp. 24–35. doi:10.1007/978-3-540-70575-8_3.

[4] C. Condrat, P. Kalla, A Gröbner basis approach to CNF-formulae preprocessing, in: Tools and Algorithms for the Construction and Analysis of Systems, Springer, 2007, pp. 618–631. doi:10.1007/978-3-540-71209-1_48.

[5] B. Buchberger, A theoretical basis for the reduction of polynomials to canonical forms, ACM SIGSAM Bulletin 10 (1976) 19–29. doi:10.1145/1088216.1088219.

[6] J. D. McCalpin, Mapping addresses to L3/CHA slices in Intel processors, Technical Report, TACC, The University of Texas at Austin, 2021. doi:10.26153/tsw/14539.

[7] T. Nguyen, Combinations of boolean Gröbner bases and SAT solvers, Ph.D. thesis, University of Kaiserslautern, 2014. doi:10.13140/RG.2.2.25392.92167.

[8] W. V. Quine, The problem of simplifying truth functions, The American mathematical monthly 59 (1952) 521–531. doi:10.1080/00029890.1952.11988183.

[9] E. J. McCluskey, Minimization of boolean functions, The Bell System Technical Journal 35 (1956) 1417–1444. doi:10.1002/j.1538-7305.1956.tb03835.x.

[10] A. Mishchenko, M. Perkowski, Fast heuristic minimization of exclusive-sums-of-products, 5th International Reed-Muller Workshop (2001).

[11] R. K. Brayton, R. Rudell, A. Sangiovanni-Vincentelli, A. R. Wang, MIS: A multiple-level logic optimization system, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems 6 (1987) 1062–1081. doi:10.1109/TCAD.1987.1270347.

[12] J. A. Darringer, D. Brand, J. V. Gerbi, W. H. Joyner, L. Trevillyan, LSS: A system for production logic synthesis, IBM Journal of Research and Development 28 (1984) 537–545. doi:10.1147/rd.285.0537.

[13] M. Perkowski, Multi-level logic minimization, http://web.cecs.pdx.edu/~mperkows/=FSM/finite-sm/node17.html, Accessed: 28.03.2023.

[14] M. Soeken, W. Haaswijk, E. Testa, A. Mishchenko, L. G. Amarù, R. K. Brayton, G. D. Micheli, Practical exact synthesis, in: 2018 Design, Automation & Test in Europe Conference & Exhibition (DATE), IEEE, 2018, pp. 309–314. doi:10.23919/DATE.2018.8342027.

[15] W. Haaswijk, M. Soeken, A. Mishchenko, G. D. Micheli, SAT-based exact synthesis: Encodings, topology families, and parallelism, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems 39 (2020) 871–884. doi:10.1109/TCAD.2019.2897703.

[16] T. Becker, V. Weispfenning, Gröbner bases, Graduate Texts in Mathematics, Springer, 1993. doi:10.1007/978-1-4612-0913-3.

[17] S. Gao, Counting zeros over finite fields with Gröbner bases, Master's thesis, Carnegie Mellon University, 2009. URL: https://www.cs.cmu.edu/~sicung/papers/MS_thesis.pdf.

[18] D. Ritirc, A. Biere, M. Kauers, Improving and extending the algebraic approach for verifying gate-level multipliers, in: 2018 Design, Automation & Test in Europe Conference & Exhibition (DATE), IEEE, 2018, pp. 1556–1561. doi:10.23919/DATE.2018.8342263.

[19] D. Kaufmann, A. Biere, M. Kauers, Verifying large multipliers by combining SAT and computer algebra, in: 2019 Formal Methods in Computer Aided Design (FMCAD), IEEE, 2019, pp. 28–36. doi:10.23919/FMCAD.2019.8894250.

[20] D. Kaufmann, A. Biere, M. Kauers, Incremental column-wise verification of arithmetic circuits using computer algebra, Formal Methods in System Design 56 (2020) 22–54. doi:10.1007/s10703-018-00329-2.

[21] T. Hader, D. Kaufmann, L. Kovacs, SMT solving over finite field arithmetic, in: Proceedings of 24th International Conference on Logic for Programming, Artificial Intelligence and Reasoning, EasyChair, 2023, pp. 238–256. doi:10.29007/4n6w.

[22] A. Ozdemir, G. Kremer, C. Tinelli, C. W. Barrett, Satisfiability modulo finite fields, volume 185 of *Frontiers in Artificial Intelligence and Applications*, IOS Press, 2009, pp. 825–885. doi:10.3233/978-1-58603-929-5-825.

[23] S. Junges, U. Loup, F. Corzilius, E. Ábrahám, On Gröbner bases in the context of satisfiability-modulo-theories solving over the real numbers, in: Algebraic Informatics, Springer, 2013, pp. 186–198. doi:10.1007/978-3-642-40663-8_18.

[24] S. R. Ghorpade, A note on Nullstellensatz over finite fields, Contemporary Mathematics

(2018).
[25] The Sage Developers, Sage Mathematics Software, Version 9.0, http://www.sagemath.org, 2023.
[26] W. Decker, G.-M. Greuel, G. Pfister, H. Schönemann, Singular — A computer algebra system for polynomial computations, Version 4.3.0, http://www.singular.uni-kl.de, 2022.
[27] GNU Linear Programming Kit, Version 4.32, http://www.gnu.org/software/glpk/glpk.html, 2008.
[28] R. Brayton, A. Mishchenko, ABC: An academic industrial-strength verification tool, in: Computer Aided Verification, Springer, 2010, pp. 24–40. doi:10.1007/978-3-642-14295-6_5.
[29] L. Gerlach, S. Schwarz, N. Faroß, M. Schwarz, Efficient and generic microarchitectural hash-function recovery, 2024. To appear in the IEEE Symposium on Security & Privacy.