An SMT Approach for Solving Polynomials over Finite Fields

Thomas Hader[1], Laura Kovács[1]

[1]TU Wien, Institut für Logic and Computation, Favoritenstraße 9-11, 1040 Wien, Austria

Abstract

In this extended abstract we present our work on solving non-linear polynomial systems over finite fields. Given a formula over (in-)equality constraints of polynomials over finite fields, we developed an automated search procedure that checks satisfiability of the polynomial system, that is checking the existence of an assignment of the polynomial variables to values from the finite field such that the constraints are satisfied. We have designed a Model Constructing Satisfiability (MCSat) style search procedure with two different approaches for explanation functions. We have implemented our procedure and compared its performance to state-of-the-art approaches.

Keywords

SMT, finite fields, non-linear reasoning

1. Introduction

When reasoning in emerging applications of system security [1, 2, 3, 4], computer cryptography, especially post-quantum cryptography [5, 6, 7, 8, 9, 10], or computational biology [11, 12], one is often faced with the challenge of solving non-linear arithmetic equations modelling functional behaviour of the respective application instance. In this extended abstract we propose our ongoing efforts towards developing an automated reasoning procedure for deciding the satisfiability of a system on non-linear equations over finite fields. In the area of blockchains, so-called ZK-rollups are based on polynomials over (a very large) finite field [13]. Proving properties over those can be helpful, while breaking them could have serious consequences.

There have been many approaches presented for solving systems of polynomials over finite fields. Earlier procedures are based on decomposing the system into multiple systems with specific properties. They are called triangular sets [14, 15] or characteristic sets [16, 17]. In the recent years most solutions are based on Gröbner bases which are well known for solving polynomial systems in general. In the case of finite field polynomials, specialized approaches have been developed, most notably the algorithms F4 [18] and F5 [19]. A further procedures is the widely known XL algorithm [20]. One common aspect of all these works is that they are fully solving the system - they are describing the full solution space even if just a satisfiability answer is required. To the best of our knowledge, no SMT style approach has been so far proposed for solving the satisfiability of polynomial equations over finite fields.

SMT 2022: Satisfiability Modulo Theories, August 11–12, 2022, Haifa, Israel

thomas.hader@tuwien.ac.at (T. Hader); laura.kovacs@tuwien.ac.at (L. Kovács)

2. Our SMT Problem

In algebra, a *field* is a set of elements that is closed with regard to the operations sum, difference, product and inverse finding in their usual definitions. Well known examples for fields (of infinite size) are the rational, real, and complex numbers. When the number of elements in a field is finite, it is denoted as a *finite field.* The *order* of a finite field is defined as the number of the field's elements.

Given a number $q = p^n$, where p is prime, and $n \geq 1$, we can construct a finite field of order q. As all finite fields of order q are isomorphic, we denote the field as $\mathbb{F}_q$.

In case $n = 1$ the integers modulo p describe a finite field with elements $\{0, 1, \dots, p - 1\}$. Addition, subtraction and multiplication can be performed in the integer domain followed by the modulo operation. To get the multiplicative inverse, one computes the extended Euclidean algorithm

For $n > 1$, the elements of the field are polynomials over $\mathbb{F}_p$ modulo an irreducible polynomial over $\mathbb{F}_q$ with degree n. The operations are the corresponding polynomial operations over the quotient ring. A polynomial is *irreducible* when it cannot be factored into polynomials of smaller degree, i.e. it cannot be represented as a product of polynomials of smaller degree.

Example 1. *A finite field with 5 elements is* $\mathbb{F}_5 = \{0, 1, 2, 3, 4\}$*. The term* $(2 \cdot 3) + 4$ *evaluates to* 0 *in* $\mathbb{F}_5$*. A field of order 4 using the (only) irreducible polynomial over* $\mathbb{F}_2$*:* $a^2 + a + 1$ *consists of* $\mathbb{F}_{2^2} = \{0, 1, a, 1 + a\}$*. The term* $((a) + (1)) \cdot (a + 1)$ *evaluates to* a *under* $\mathbb{F}_{2^2}$*.*

Given a finite field $\mathbb{F}_q$, a multivariate polynomial $f \in \mathbb{F}_q[x_1, \dots, x_n]$ is of the form

$$f(x_1, \dots, x_n) = a_m \cdot x_n^{d_m} + a_{m-1} \cdot x_n^{d_{m-1}} + \cdots + a_1 \cdot x_n^{d_1} + a_0$$

where $0 < d_1 < \cdots < d_m$ are non-negative integers and the coefficients a_i are in $\mathbb{F}_q[x_1, \dots, x_{n-1}]$ with $a_m \neq 0$. A *(polynomial) constraint* is of the form $f \rhd 0$ where $\rhd \in \{=, \neq\}$. With an *assignment function* $\nu : X \to \mathbb{F}_q$ the constraint can be evaluated by replacing the variables in f accordingly and evaluating the (dis-)equality. As usual, we denote a set of constraints a *clause* and a set of clauses a *formula.* We refer to a constraint that is part of a clause as a *literal.* A formula is *satisfied* by ν if and only if each clause contains at least one constraint that evaluates to true under ν. A formula is *satisfiable* if such an assignment function exists.

SMT Problem Statement. Consider a finite field $\mathbb{F}_q$ with $q = p^n$, where p is prime and $n \geq 1$. Let $\mathcal{F}$ be a formula as defined above. Then the literals in the clauses of $\mathcal{F}$ are constraints over polynomials in $\mathbb{F}_q[x_1, \dots, x_n]$. Our SMT problem as the following satisfiability query: *Does there exist an assignment function* $\nu : \{x_1, \dots, x_n\} \to \mathbb{F}_q$ *that satisfies* $\mathcal{F}$*?*

Note that the problem of solving a polynomial system, i.e. finding a common zero for a set of given polynomials, can trivially be formulated as a conjunction of unit clauses and, therefore, is in the scope of our SMT problem.

Example 2. *Given the formula* $\mathcal{F}$ *with polynomials over* $\mathbb{F}_3[x, y]$*.*

$$\mathcal{F} = \{ \underbrace{\{x^2y - 1 = 0\}}_{C_1}, \underbrace{\{y = 0,\ 2y + 1 = 0\}}_{C_2} \}$$

The formula $\mathcal{F}$ *is satisfiable with the assignment function* $\nu : \{x \mapsto 1, y \mapsto 1\}$*.*

3. Non-Linear SMT Reasoning over Finite Fields

Our approach for solving the SMT problem of Section 2 is based on the Model Constructing Satisfiability (MCSat) calculus presented in [21]. This calculus was successfully applied in various areas, most notably for solving linear integer constraints [22] and for solving non-linear constraints over reals [23]. In our work, we ajdusted the MCSat approach towards finding solutions over finite fields (Section 3.1), while resolving propositional conflicts (Section 3.2).

In this extended abstract, we only give a high-level description of the procedure and show the basic ideas. We refer to [24] for further details, including formal definitions and proofs.

3.1. Search Procedure

The MCSat procedure combines Conflict-Driven Clause Learning (CDCL) with theory reasoning [21]. During the search for satisfying assignments, we are deciding on literals as well as values for the polynomial's variables. This is reflected in the search procedure's trail which, in addition to tracking polynomial constraints that are decided or required by propagation to be true, contains assignments α_i to polynomial variables x_i. A trail M has the following structure:

$$M = [\![\dots, F_0, \dots, F_m, x_{k-1} \mapsto \alpha_{k-1}, G_1, G_2, \dots, G_l]\!],$$

where the constraints F_i contain polynomials over $\mathbb{F}_q[x_1, \dots, x_{k-1}]$, and the constraints G_i are over $\mathbb{F}_q[x_1, \dots, x_k]$. A polynomials constraint G_i is added to the trail if and only if all its theory variables but x_k (the highest according to some predefined variable order) are assigned a value. Once all such polynomials are added, we determine a suitable value for x_k. If there is a value α_k that fulfils all constraints G_i, we add $x_k \leftarrow \alpha_k$ and continue the search with x_{k+1}. If no such value exists, we need to derive a new clause E (called *explanation clause*) to represent this fact. E is constructed in a way that a conflict occurs and a regular CDCL style conflict analysis can start right away. This generates a conflict clause which is then used to backtrack and lead the search in a new direction. Generating explanations is the core strength of a MCSat style procedure, as it translates theory knowledge (e.g., the fact that there is no suitable value for a variable) into a clause that can be utilised by CDCL.

While explanation clauses are primarily generated when a conflict occurs, we can utilize this feature in other instances as well. For example, consider a constraint that is fulfilled independent of the assignment of some variables. This fact can be expressed using an explanation clause to further guide the search.

Example 3. *A trail while searching for a solution of Example 2 might be:*

$$M = [\![(y = 0), y \mapsto 0, E \rightarrow (x^2y - 1 \neq 0)]\!]$$

Where $E = \{x^2y - 1 \neq 0,\ y \neq 0\}$ is a generated explanation clause. Note that E is generated such that an immediate unit propagation is possible and the search is in a conflicting state.

3.2. Generating Explanations

Finding a procedure to generate explanations is the key challenge when applying a MCSat style procedure to a new theory domain.

The explanation clause generation procedure takes a trail M containing assignments for variables $x_1, \ldots, x_{k-1}$ as well as a polynomial constraint G over $\mathbb{F}_q[x_1, \ldots, x_k]$ that makes the trail incompatible. This means that by adding G to the M, the (previously non-empty) set of possible values for the (yet unassigned) variable x_k gets empty. Therefore, M cannot be extended with an assignment for x_k after G is added.

To generate an explanation clause, we first generate a set of constraints $\mathcal{C}$ that contains all constraints from M as well as G. It then holds that

$$\bigwedge_{c \in \mathcal{C}} c \implies \exists x_n \bigwedge_{c \in \mathcal{C}} c$$

Utilizing a quantifier elimination procedure, we can remove $\exists x_n$ and get a formula $\mathcal{C}'$ which contains only constraints with polynomials in $\mathbb{F}_q[x_1, \ldots, x_{n-1}]$ and, therefore, can be fully evaluated using the variable assignments in M. Let $\nu[M]$ be the assignment function generated from the assignments in M. Representing $\mathcal{C}'$ in conjunctive normal form (CNF), there must be at least one clause that is false under $\nu[M]$ as we have added the incompatible constraint G. We can use this clause (together with constraints from $\mathcal{C}$) to construct the explanation clause E. Note that it suffices to generate one such clause and it is not required to fully generate $\mathcal{C}'$.

Example 4. *We show how the clause E in Example 3 is generated. After arriving at a trail*

$$M = [\![(y = 0), y \mapsto 0]\!]$$

we are searching for an assignment for x. However, there is no feasible value to satisfy the unit clause $C_1 = \{x^2 y - 1 = 0\}$. Therefore,

$$\exists x. y = 0 \wedge x^2 y - 1 = 0$$

is not satisfied by the current assignment $\nu = \{y \mapsto 0\}$. When applying a quantifier elimination procedure we can derive that there is in fact no solution and we thus generate the explanation clause $E = \{x^2 y - 1 \neq 0,\ y \neq 0\}$.

There are multiple quantifier elimination procedures for polynomials over finite fields. In our work, we have constructed two alternative explanation functions based on two quantifier elimination procedures, in particular using a tailored approach for eliminating existential quantifiers [25] and using Gröbner basis computation for deriving elimination ideals of polynomials [26, 27].

Elimination theory. The first approach we present is based on [25] for solving systems of polynomials, and adjusting the solving procedure over finite fields. Given a polynomial system $S = (S_=, S_{\neq})$, with finite sets of polynomials $S_=, S_{\neq} \subset \mathbb{F}_q[x_1, \ldots, x_n]$, a solution (i.e. a zero) of S is a tuple $\alpha \in \mathbb{F}_q^n$ such that for all $s \in S_=$, $s(\alpha) = 0$ and for all $s' \in S_{\neq}$, $s'(\alpha) \neq 0$. Given a set of polynomial constraints $\mathcal{C}$, we can easily generate such a system.

Following the ideas in [25], in [24] we present a set of algorithms that decomposes the system S into multiple systems such that

$$\mathsf{Proj}^{x_1, \ldots, x_{n-1}} \mathsf{zero}(S) = \bigcup_{s \in \mathcal{S}} \mathsf{zero}(s)$$

and the systems in $\mathcal{S}$ only contain polynomials from $\mathbb{F}_q[x_1, \ldots, x_{n-1}]$. The projection operator $\mathsf{Proj}^{x_1,\ldots,x_{n-1}}$ translates the zeros from the $\mathbb{F}_q^n$ space into the $\mathbb{F}_q^{n-1}$ space by simply removing the last element of the n-tuple. Note that the decomposition process ensures that every solution $(\alpha_1, \ldots, \alpha_{n-1}) \in \bigcup_{s \in \mathcal{S}} \mathsf{zero}(s)$ can be extended to a solution for S, i.e. there exists an $\beta \in \mathbb{F}_q$ such that $(\alpha_1, \ldots, \alpha_{n-1}, \beta) \in \mathsf{zero}(S)$. Therefore, this is the desired existential quantification elimination procedure we need to generate an explanation clause. We generate the explanation clause E by adding one polynomial constraint from each decomposed system $s \in \mathcal{S}$ such that the current assignment $\nu[M]$ does not satisfy the constraint.

As mentioned above, we are not required to calculate the full decomposition. We can abort the procedure once the assignment function for the current trail is excluded and thus keep the size of $\mathcal{S}$ reasonably small compared to fully decomposing the system for direct solving.

Gröbner basis computation. Gröbner bases yield a finite representation of polynomial ideals. The algorithmic computation of Gröbner bases implements a saturation/completion method in order to derive generating sets (i.e. bases) for polynomial ideals in a polynomial ring. Roughly speaking, the ideal described by a given finite basis contains all polynomials that have the same zero set. Although an ideal is, in general, infinite in size, there always exists finite bases, in particular a Gröbner basis w.r.t. to a given variable ordering (extended to a monomial ordering).

When a basis G is Gröbner with a lexicographical monomial ordering, it admits existential quantifier elimination. This means that

$$\mathsf{Proj}^{x_1,\ldots,x_{n-1}} \mathsf{zero}(G) = \mathsf{zero}(G \cap \mathbb{F}_q[x_1, \ldots, x_{n-1}])$$

holds and we can generate an explanation clause by removing all polynomials that contain x_k from G. Quantifier elimination for polynomials over finite fields using Gröbner bases has been presented in [28].

Note that G describes a basis, i.e. a set of polynomials, instead of a polynomial system as defined above. Thus, we need to convert dis-equalities to equalities beforehand. This can be done by introducing a new variable for each converted constraint. These additional variables are later removed in the same way as x_n. There are many Gröbner basis algorithms specially tailored for finite fields (e.g. [20, 18, 19]), however, incorporating the current trail's assignment into a Gröbner basis based explanation procedure remains an open problem. Nevertheless, by reducing the input to the procedure - the core of a conflict instead of the whole problem - a performance benefit can often be observed.

4. Implementation and Results

A first prototype of our procedure was implemented in Python using Sage [29]. For performance evaluation, we generated random inputs. We compared our approach to solving an input instance with Sage's built-in capabilities utilizing lexicographical Gröbner bases to create an elimination ideal for the whole problem at once which, to the best of our knowledge, is the state-of-the-art. We compared four solving approaches: GFSAT(ELIM) and GFSAT(GB) are our approaches with elimination theory and Gröbner bases for explanation generation, respectively;

Table 1
Performance comparison for $\mathbb{F}_3$. Number of instances solved within 30 seconds out of 25 instances for each approach and category. A category is described by the number of polynomials (#poly) and the number of variables (#var) of its instance.

#var	8	8	10	10
#poly	8	12	8	12
total	25	25	25	25
GFSAT(GB)	4	1	1	1
GFSAT(ELIM)	25	25	19	16
GB	25	25	7	13
BF	25	2	0	0

Suite I

#var	32	64	64	128
#poly	32	64	128	256
total	25	25	25	25
GFSAT(GB)	25	25	25	25
GFSAT(ELIM)	25	25	25	25
GB	16	19	16	0
BF	0	0	0	0

Suite R

GB utilizes Sage's built-in Gröbner bases capabilities to solve the whole instance using one ideal; and BF is a brute force approach. For the calculation of elimination ideals in GFSAT(GB) and GB, Sage utilizes the `eliminate` command of the Singular library. Note that our implementation has a significant engineering disadvantage compared to Sage's internal capabilities. While Sage utilizes highly engineered routines written in C++ and compiled to machine code, our prototype is using interpreted Python for most of its runtime. A native implementation of our procedure would certainly lead to a significant performance increase.

During development, we observed that the effort of solving polynomial systems depends to a huge extent on the amount of irreducible factors in the system's polynomials. This holds true for both, the regular Gröbner basis methods, as well as our approach. Irreducible factors do not contribute to a solution as their zeros are not in the base field. Therefore, when generating benchmarks for performance comparison, we decided to take this observation into account by generating two sets of benchmarks. The first benchmark set (suite I) consists of polynomials with mostly irreducible factors. Polynomials of the second suite (suite R) have almost exclusively zeros from the base field.

Table 1 gives an overview over the number of instances that could be solved with each approach for suite I and R. All our experiments were conducted on an Intel Core i5-8365U CPU @ 1.60GHz with 16GB of RAM running Linux and Sage release 9.5. Experiments clearly show that irreducible factors in the input polynomials have a performance impact. For GFSAT(GB) this has a worse impact than for GFSAT(ELIM). We assume that this is due to the fact that explanations generated by the latter are weaker but easier to compute. Constraints generated by GFSAT(GB) tend to grow much faster in size and, thus, become intractable quicker. In suite R, the performance difference is much smaller, nevertheless, GFSAT(ELIM) tend to outperform GFSAT(GB) slightly.

While the theory holds for any finite field, our experiments show that the approach is only practically feasible when the field's size is rather limited. With our current approach, solving systems in finite fields beyond a single digit field size is unfortunately intractable. Our experiments have shown that the current bottleneck of our approach is the vastly growing size of the generated polynomials when generating explanations. This is especially the case when the size of the base field increases even slightly. It is certainly a next step to weaken the explanations in favour of polynomial length.

Unfortunately, most real world applications mentioned in the introduction often require finite fields of order well over 2^{200}, which is far beyond our current capabilities and remains a challenge for future research.

5. Summary and Future Work

In our work so far, we have concluded that an MCSat style search is suited for solving non-linear polynomials over finite fields. For that we have developed two independent approaches for explanation generation based on elimination theory as well as Gröbner bases.

So far, we could show that a MCSat search procedure performs better compared to traditional solving techniques for polynomials over finite fields. We believe that this is because calling the quantifier elimination procedures with smaller subproblems leads to an overall better performance. Furthermore, using a partial assignment allows us to stop the solving procedures early on.

In our next steps, we will work on how we can utilize the current assignment in Gröbner basis calculation and check how the growth of polynomials can be tackled. Moreover, we are interested in applying our SMT procedure on examples coming from real-world applications, potentially improving our approach towards solving problems with large number of variables and constraints (yet, with likely relatively small polynomial degrees).

Acknowledgments

We thank Nikolaj Bjørner for the fruitful discussion on this work. We also acknowledge partial support from the ERC Consolidator Grant ARTIST 101002685 and the TU Wien SecInt Doctoral College for the work leading to this extended abstract.

References

[1] C. Schneidewind, I. Grishchenko, M. Scherer, M. Maffei, eThor: Practical and Provably Sound Static Analysis of Ethereum Smart Contracts, in: CCS, 2020, pp. 621–640.

[2] N. Elad, S. Rain, N. Immerman, L. Kovács, M. Sagiv, Summing up Smart Transitions, in: CAV, 2021, pp. 317–340.

[3] E. Albert, P. Gordillo, A. Hernández-Cerezo, A. Rubio, A Max-SMT Superoptimizer for EVM handling Memory and Storage, in: TACAS, 2022, pp. 201–219.

[4] B. Tan, B. Mariano, S. K. Lahiri, I. Dillig, Y. Feng, SolType: refinement types for arithmetic overflow in solidity, POPL. (2022) 1–29.

[5] J.-C. Faugere, A. Joux, Algebraic cryptanalysis of hidden field equation (hfe) cryptosystems using gröbner bases, in: Annual International Cryptology Conference, Springer, 2003, pp. 44–60.

[6] J.-C. Faugere, A. Otmani, L. Perret, J.-P. Tillich, Algebraic cryptanalysis of mceliece variants with compact keys, in: Annual International Conference on the Theory and Applications of Cryptographic Techniques, Springer, 2010, pp. 279–298.

[7] N. T. Courtois, J. Pieprzyk, Cryptanalysis of block ciphers with overdefined systems of equations, in: International conference on the theory and application of cryptology and information security, Springer, 2002, pp. 267–287.

[8] I. Dinur, A. Shamir, Cube attacks on tweakable black box polynomials, in: Annual international conference on the theory and applications of cryptographic techniques, Springer, 2009, pp. 278–299.

[9] C. Cid, S. Murphy, M. Robshaw, Algebraic aspects of the advanced encryption standard, Springer Science & Business Media, 2006.

[10] N. T. Courtois, G. V. Bard, Algebraic cryptanalysis of the data encryption standard, in: IMA International Conference on Cryptography and Coding, Springer, 2007, pp. 152–169.

[11] A. Jarrah, H. Vastani, K. Duca, R. Laubenbacher, An optimal control problem for in vitro virus competition, in: 2004 43rd IEEE Conference on Decision and Control (CDC)(IEEE Cat. No. 04CH37601), volume 1, IEEE, 2004, pp. 579–584.

[12] L. Pachter, B. Sturmfels, Algebraic statistics for computational biology, volume 13, Cambridge university press, 2005.

[13] The Ethereum Community, Zero-knowledge rollups, 2022. URL: https://ethereum.org/en/developers/docs/scaling/zk-rollups/.

[14] P. Aubry, D. Lazard, M. M. Maza, On the theories of triangular sets, Journal of Symbolic Computation 28 (1999) 105–124.

[15] P. Aubry, M. M. Maza, Triangular sets for solving polynomial systems: a comparative implementation of four methods, Journal of Symbolic Computation 28 (1999) 125–154.

[16] Z. Huang, Parametric equation solving and quantifier elimination in finite fields with the characteristic set method, Journal of Systems Science and Complexity 25 (2012) 778–791.

[17] X.-S. Gao, Z. Huang, Characteristic set algorithms for equation solving in finite fields, Journal of Symbolic Computation 47 (2012) 655–679.

[18] J. C. Faugere, A new efficient algorithm for computing gröbner bases (F4), Journal of pure and applied algebra 139 (1999) 61–88.

[19] J. C. Faugere, A new efficient algorithm for computing gröbner bases without reduction to zero (F5), in: Proceedings of the 2002 international symposium on Symbolic and algebraic computation, 2002, pp. 75–83.

[20] N. Courtois, A. Klimov, J. Patarin, A. Shamir, Efficient algorithms for solving overdefined systems of multivariate polynomial equations, in: International Conference on the Theory and Applications of Cryptographic Techniques, Springer, 2000, pp. 392–407.

[21] L. De Moura, D. Jovanović, A model-constructing satisfiability calculus, in: International Workshop on Verification, Model Checking, and Abstract Interpretation, Springer, 2013, pp. 1–12.

[22] D. Jovanović, L. De Moura, Cutting to the chase solving linear integer arithmetic, in: International Conference on Automated Deduction, Springer, 2011, pp. 338–353.

[23] D. Jovanović, L. De Moura, Solving non-linear arithmetic, in: International Joint Conference on Automated Reasoning, Springer, 2012, pp. 339–354.

[24] T. Hader, Non-Linear SMT-Reasoning over Finite Fields, Master's thesis, TU Wien, Vienna, 2022.

[25] D. Wang, Elimination methods, Springer Science & Business Media, 2001.

[26] B. Buchberger, A Theoretical Basis for the Reduction of Polynomials to Canonical Forms,

ACM SIGSAM Bulletin 10 (1976) 19–29.
[27] D. Cox, J. Little, D. OShea, Ideals, varieties, and algorithms: an introduction to computational algebraic geometry and commutative algebra, Springer Science & Business Media, 2013.
[28] S. Gao, A. Platzer, E. M. Clarke, Quantifier elimination over finite fields using gröbner bases, in: International Conference on Algebraic Informatics, Springer, 2011, pp. 140–157.
[29] The Sage Developers, SageMath, the Sage Mathematics Software System (Version 9.5), 2022. `https://www.sagemath.org`.