MCSat-Based Finite Field Reasoning in the *Yices2* SMT Solver (Short Paper)

Thomas Hader[1](✉), Daniela Kaufmann[1], Ahmed Irfan[2], Stéphane Graham-Lengrand[2], and Laura Kovács[1]

[1] TU Wien, Vienna, Austria
{thomas.hader,daniela.kaufmann,laura.kovacs}@tuwien.ac.at
[2] SRI International, Menlo Park, CA, USA
{ahmed.irfan,stephane.graham-lengrand}@sri.com

Abstract. This system description introduces an enhancement to the Yices2 SMT solver, enabling it to reason over non-linear polynomial systems over finite fields. Our reasoning approach fits into the model-constructing satisfiability (MCSat) framework and is based on zero decomposition techniques, which find finite basis explanations for theory conflicts over finite fields. As the MCSat solver within Yices2 can support (and combine) several theories via theory plugins, we implemented our reasoning approach as a new plugin for finite fields and extended Yices2's frontend to parse finite field problems, making our implementation the first MCSat-based reasoning engine for finite fields. We present its evaluation on finite field benchmarks, comparing it against cvc5. Additionally, our work leverages the modular architecture of the MCSat solver in Yices2 to provide a foundation for the rapid implementation of further reasoning techniques for this theory.

Keywords: SMT solving · MCSat · finite fields · polynomial arithmetic

1 Introduction

Satisfiability Modulo Theories (SMT) solving plays a crucial role in automated reasoning as it combines the power of Boolean satisfiability (SAT) with various mathematical background theories [3]. This connection enables the automated verification and synthesis of systems [15] that require reasoning in more expressive logical theories, for example real/integer arithmetic.

State-of-the-art SMT solvers employ a combination of Boolean level reasoning and theory-specific algorithms. This is achieved either through the use of the CDCL(T) paradigm [16] or the model-constructing satisfiability (MCSat) approach [11,14]. The MCSat algorithm lifts the Boolean-level CDCL algorithm to the theory level, while keeping the search theory independent. This approach is particularly effective for handling complex arithmetic theories. For instance, Yices2 [5] uses the MCSat approach to handle non-linear arithmetic constraints.

Finite fields offer an ideal framework for modeling bounded machine arithmetic, particularly relevant in the context of contemporary cryptosystems utilized in system security and post-quantum cryptography. Current methodologies, for instance, develop private and secure systems using zero-knowledge (ZK) proofs [7] or authenticate blockchain technologies like smart contracts [19]. Verifying applications in such areas require efficient SMT solvers that support reasoning over finite field arithmetic, e.g., verification of a compiler for ZK proofs [18].

Related Work. Currently, the related work on SMT solving in finite field arithmetic is still rather limited. Our own theoretical work [9] on MCSat approaches based on finding zero decompositions comes with a proof-of-concept implementation that facilitates only a very fundamental MCSat algorithm, has only limited support of Boolean propagation, and is unable to parse SMT-LIB 2 [2].

The only other SMT solver that we are aware of being capable of reasoning over finite fields is CVC5 [1,17], which uses a classical CDCL(T) approach. As a theory engine, Gröbner bases [4] reasoning over a set of polynomial equalities is applied. If the derived Gröbner basis contains the constant 1, then the system is unsatisfiable and a conflict core for the CDCL(T) search can be found. Otherwise, a guided enumeration of all possible solutions is performed to search for a model.

Note that both approaches [9,17] use complementary techniques. On the one hand, Gröbner bases are highly engineered to find conflicts in the polynomial input, which tends to help for unsatisfiable instances [17]. On the other hand, a model constructing approach tends to be fast whenever there is a solution, especially when there is a high number of models [9].

We further note that at the moment our implementation in YICES2, as well as CVC5, is restricted to finite fields where the order (i.e. size) is prime. This limitation is sufficient for many applications in cryptosystems and ZK proofs.

Besides using dedicated finite field solvers, problems over prime fields can be encoded in integer arithmetic using the modulo operator. Further, since terms are bounded, encoding as bit-vectors for subsequent bit-blasting is possible. However, prior experiments have shown that those encodings perform poorly on existing solvers [17].

Contributions. We present an integration of the theory of non-linear finite field arithmetic in the YICES2 SMT solver [5], enabling it to reason over finite field problems. This includes the following contributions which we will further explain throughout the rest of this paper:

- Adding the parsing of finite field problems to the YICES2 SMT-LIB 2 front-end (Sect. 3).
- Representing finite field polynomials as terms in YICES2 and implementing features regarding such polynomials in the LIBPOLY library [12], which is the library used for polynomials in YICES2 (Sect. 4).
- Implementing and evaluating an MCSat theory back-end for finite field reasoning, using existing concepts from non-linear real and bit-vector solving from YICES2 (Sect. 5).

To the best of our knowledge, our work is currently the only finite field instantiation of MCSat. While our initial theory reasoning approach follows closely the explanation generation procedure of our previous work [9], our implementation allows easy drop-in of an improved explanation procedure in the future.

2 Preliminaries

In mathematics, a finite field is a field that contains a finite number of elements. A finite field $\mathbb{F}_p$ of prime order p can roughly be seen as the representation of the integers modulo the prime p. We refer to [9,17] for details on the theory and representation of finite fields. Since there is no inherent order on finite fields, polynomial constraints are either equalities $p = 0$ or disequalities $p \neq 0$ for a finite field polynomial p.

For SMT solving in finite fields, we are interested in the following problem:

Given a finite field $\mathbb{F}_p$, where p is a prime number, let $X = x_1, \dots x_n$, let F be a set of polynomial constraints in $\mathbb{F}_p[X]$ and $\mathcal{F}$ a formula following the logical structure:

$$\mathcal{F} \quad = \quad \bigwedge_{C \subseteq F} \bigvee_{f \in C} f \quad = \quad \bigwedge_{C \subseteq F} \bigvee_{f \in C} \mathsf{poly}(f) \rhd 0 \quad \text{with } \rhd \in \{=, \neq\}.$$

SMT solving over finite fields: Does an assignment $\nu : \{x_1, \dots, x_n\} \rightarrow \mathbb{F}_p^n$ exist that satisfies $\mathcal{F}$?

For example, the formula $\mathcal{F}_1 = (x - 1 = 0 \vee y - 1 = 0) \wedge (xy - 1 = 0)$ is satisfied by the assignment $x \mapsto 1, y \mapsto 1$ in $\mathbb{F}_3$; whereas the formula $\mathcal{F}_2 = (x - 1 = 0 \vee y - 1 = 0) \wedge (xy - 1 = 0) \wedge (x - 2 = 0)$ is unsatisfiable in $\mathbb{F}_3$.

Yices2 and MCSat. Yices2 contains two main solvers, one based on the traditional CDCL(T) approach [16] and one based on the MCSat approach [13,14]. Yices2's common API and front-ends can automatically select which solver to use at runtime, depending on an SMT-LIB 2 logic. In particular, when non-linear real or integer arithmetic constraints are present Yices2 selects the MCSat solver. The MCSat solver in Yices2 currently supports the theories of non-linear real arithmetic (QF_NRA) [13] and integer arithmetic (QF_NIA) [10], bit-vectors (QF_BV) [8], equality and uninterpreted functions (QF_EUF), arrays [6], and combinations thereof.

In contrast to CDCL(T) that *complements* CDCL with theory reasoning, MCSat applies CDCL-like mechanisms to *perform* theory reasoning. Specifically, it explicitly and incrementally constructs models with first-order variable assignments—maintained in a *trail*—while maintaining the invariant that none of the constraints evaluate to false. MCSat decides upon such assignments when there is choice, it can propagate them when there is not, and it backtracks upon conflict. The lemmas learned upon backtracking are based on theory-specific explanations of conflicts and propagations. This theory-specific reasoning is implemented through *plugins* that provide interfaces to make decisions, perform propagations, and produce explanations.

3 Usability of SMT Solving in Finite Fields

Support for finite field reasoning in Yices2 is available on the master branch[1]and will be included in the next release (2.7). The theory of finite fields can be accessed using a not-yet official extension of the SMT-LIB 2 language (.smt2).

SMT-LIB 2 Parsing. Extending the parser to handle finite field problems was our first extension to Yices2. Currently, polynomials over finite fields are no official theory in SMT-LIB 2 [2]. However, when implementing finite field support in cvc5 [1], an extension was proposed in [17]. In the interest of keeping inputs and benchmarks comparable, we aimed at a compatible implementation. Standardization efforts to create an official SMT-LIB 2 theory for finite field arithmetic are currently ongoing.

In the SMT-LIB 2 extension, the theory of (quantifier-free) non-linear finite field arithmetic is denoted as `QF_FFA`. Elements can be defined using the sort `FiniteField`. The sort is indexed by the order of the finite field, which is required to be a prime number. For instance `(_ FiniteField 3)` defines the finite field of order 3. Constants are indexed with the field order to indicate which finite field they belong in, e.g., `(_ ff2 3)`. Note that the integer following `ff` is interpreted modulo the field order. As a short-cut to avoid rewriting the field order over and over again, the `as` keyword can be used to interpret the constant in the correct field type: `(as ff2 FF3)` for a defined finite field sort `FF3`. To specify the finite field operations `ff.mul` and `ff.add` are available for multiplication and addition of finite field values, respectively. Both support an arbitrary number of operators. Atoms with finite field terms can be = with its respective meaning. For example, an encoding of $\mathcal{F}_1$ can be seen in Fig. 1.

```
(set-logic QF_FFA)
(define-sort FF3 () (_ FiniteField 3))
(declare-fun x () FF3)
(declare-fun y () FF3)
(assert (and
        (or (= (ff.add x (as ff-1 FF3)) (as ff0 FF3))
            (= (ff.add y (as ff-1 FF3)) (as ff0 FF3)))
        (= (ff.mul x y) (as ff-1 FF3))))
(check-sat)
```

Fig. 1. Example for an SMT-LIB 2 encoding of a finite field problem $\mathcal{F}_1$.

[1] Available at https://github.com/SRI-CSL/yices2.

4 Implementation Details

The Implementation of MCSat in Yices2. The MCSat solver in Yices2 supports multiple theories via a notion of theory *plugin* that builds upon an earlier architecture [11]. An MCSat theory plugin in Yices2 implements a number of functionalities that are given to the main MCSat solver as function pointers. The main MCSat loop calls these functions for theory-specific operations such as deciding or propagating the value of variables or getting explanation lemmas, or upon certain events such as the creation of new terms and lemmas. In return, a theory plugin can access theory-generic mechanisms for, e.g., inspecting the MCSat trail, creating variables and requesting to be notified of certain events like variable assignments, as well as raising conflicts. A theory plugin is not required to implement mechanisms for propagating theory assignments and explaining them, but for the current theories in Yices2, such propagations have provided noticeable speed-ups (see, e.g., [8]).

The Finite Field MCSat Plugin. Before handling constraints in the finite field MCSat plugin, the input assertions are represented as polynomial constraints. Limited preprocessing (e.g., constant propagation) is performed at this step. Internally, the plugin only handles polynomial equalities and disequalities. The implementation of the finite field plugin follows an approach similar to the plugin for non-linear arithmetic [10].

Using the MCSat trail, the finite field plugin reads which constraints must be fulfilled at any given time (as decided or deduced by the Boolean plugin) and tracks the assignment of values to polynomial variables. It also tracks, for each polynomial variable, the *set of feasible values* that the variable can be assigned without any of the polynomial constraints evaluating to false: Using watch lists, it detects when any of the constraints becomes *unit*, i.e. when all of its variables but one have been assigned values. Upon such detection, it computes how the constraint restricts the set of feasible values for the last remaining variable, using regular univariate polynomial factorization. When that set becomes empty, the plugin reports a theory conflict to the main MCSat engine. Given a conflict core and the current assignment, the *explanation procedure* in the plugin generates a (globally valid) lemma that explains the conflict in that it excludes a class of assignments (including the current one) that all violate the conflict core. The MCSat engine performs conflict analysis using theory explanations and Boolean resolutions, and either backtracks if it can or concludes unsatisfiability. On the other hand, the instance is satisfiable once all variables are assigned a value.

Finite Field Explanations. In our earlier work [9], we presented an explanation procedure for finite fields. This approach employs subresultant regular subchains (SRS) [20] between conflicting polynomials to provide new polynomial constraints that can be propagated. In a nutshell, SRS can be used to construct a generalized greatest common divisor (GCD) of polynomials that takes into account the current partial variable assignment on the trail. The computed GCD is utilized in a zero decomposition procedure to reduce the degree of the conflicting polynomials until we can learn a polynomial constraint that excludes the current partial assignment. This constraint is added as an explanation clause to resolve the conflict. We implemented the procedure of [9] in the current version of Yices2 using LibPoly. However, it is important to note that there are other solving techniques for polynomial systems over finite field that could potentially be utilized to develop an explanation method suitable for an MCSat-based search. Furthermore, it is still an open question how different techniques perform in an MCSat environment. That is why we have kept the explanation procedure encapsulated in our implementation, allowing for easy extension in order to support development and evaluation of future explanation procedures.

5 Evaluation

Since finite field solving is a rather new endeavor in the world of SMT, no extensive set of SMT-LIB 2 benchmarks exists yet. For the evaluation we have selected the benchmark sets presented in the papers describing the theory behind the implementation of Yices2 and cvc5:

(i) The polynomial sets from our prior work [9], consisting of 325 instances. These benchmarks primarily contains finite fields up to order 211, using two classes of polynomial systems: *randomly generated* as well as *crafted* systems. The crafted benchmarks are product of (mostly) degree-1 polynomials.
(ii) Benchmarks generated using ZK proof compilers presented in [17]. Besides polynomial equations, these 1602 benchmark instances also contain Boolean structure. The field order varies form small (11) up to vast (more than 2^{255}).

Experimental Setup. Our experiments were run on an AMD EPYC 7502 CPU with a timeout of 300 seconds per benchmark instance. We compared our implementation of Yices2 against cvc5 version 1.1.1, which is the latest released version at the time of writing. We are not aware of any further SMT solvers supporting the theory of non-linear finite fields to be included in the comparison.

Experimental Results. The performance comparison between the two solvers on the first benchmark set can be seen in Fig. 2 and Fig. 3 (left). It is clear to see that the random instances are harder to solve than the crafted instances (which have significantly more variables). We believe that this is due to the lack of internal structure in random polynomials. This makes symbolic handling of those polynomial systems hard, both for Gröbner basis computation (in cvc5) as well as SRS computation (in Yices2).

Note that cvc5 performs most symbolic computation upfront (when generating the Gröbner basis) and enumerates potential solutions in a second step (using auxiliary Gröbner basis calls). The MCSat approach in Yices2, on the other hand, interleaves model generation and symbolic computation during the search. This tends to be an advantage for harder polynomial systems especially together with small finite field orders. When the finite field order increases, this advantage seems to vanish. For the crafted polynomial benchmarks, Yices2

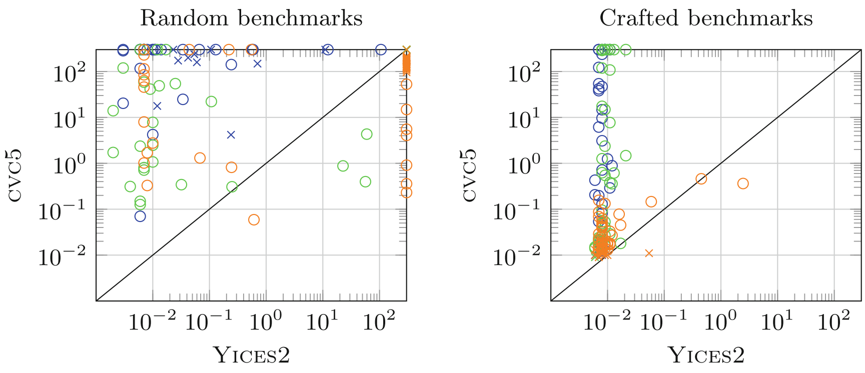

Fig. 2. Runtime comparison for benchmarks from [9] (in seconds, timeout 300 s) result: sat o, unsat x; finite field order: 3 (blue), 13 (green), and 211 (orange)

tends to be faster. We believe that this is due to the fact that the polynomials tend to be large (in the number of monomials), but rather easy to solve. Generating a full Gröbner basis upfront might add significant overhead.

For the second benchmark set, many instances can be solved by both solvers immediately (c.f. Fig. 3 right). We believe that those instances can be solved without extensive finite field reasoning, as the benchmark set contains Boolean structure. This enables both solvers to successfully solve benchmarks even with vast field orders. However, once extensive algebraic reasoning is required in finite fields of vast order (the majority of the benchmarks), the purely symbolic approach of cvc5 in proving unsatisfiability seems to be advantageous. An MCSat approach requires to pick actual values in a gigantic search space, thus especially strong lemmas need to be learned in order to prune the search space efficiently. Improving the explanation procedure is part of our future work.

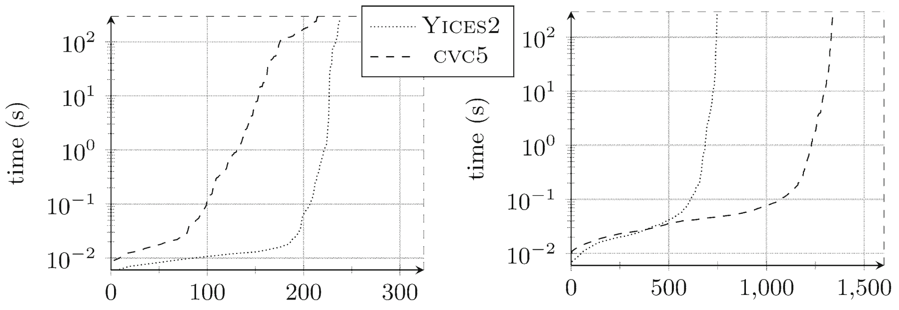

Fig. 3. Instances solved over time (timeout 300s) by Yices2 and cvc5 from [9] (left) and [17] (right).

6 Summary and Outlook

In this system description we have presented the first implementation of an MCSat-based decision procedure for non-linear finite field polynomials. We have shown that MCSat is a feasible way of solving SMT instances over finite fields and it compares well with SMT approaches using Gröbner bases for many instances.

The presented tool implementation is well suited for future experiments and the rapid development of more advanced explanation procedures that will eliminate the current bottlenecks with regard to large finite fields.

Acknowledgments. This work was conducted during the first author's stay at SRI International. We acknowledge funding from the ERC Consolidator Grant ARTIST 101002685, the TU Wien SecInt Doctoral College, the FWF grants SFB 10.55776/F8504 and ESPRIT 10.55776/ESP666), the WWTF ICT22-007 project ForSmart, the NSF award CCRI-2016597, the Amazon Research Award 2024 QuAT, and from SRI Internal Research And Development funds. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the US Government or NSF.

Disclosure of Interests. The authors have no competing interests to declare that are relevant to the content of this article.

References

1. Barbosa, H., et al.: cvc5: a versatile and industrial-strength SMT solver. In: TACAS 2022. LNCS, vol. 13243, pp. 415–442. Springer, Cham (2022). https://doi.org/10.1007/978-3-030-99524-9_24
2. Barrett, C., Fontaine, P., Tinelli, C.: The Satisfiability Modulo Theories Library (SMT-LIB). www.SMT-LIB.org (2016)
3. Barrett, C.W., Sebastiani, R., Seshia, S.A., Tinelli, C.: Satisfiability modulo theories. In: Biere, A., Heule, M., van Maaren, H., Walsh, T. (eds.) Handbook of Satisfiability - Second Edition, Frontiers in Artificial Intelligence and Applications, vol. 336, pp. 1267–1329. IOS Press (2021). https://doi.org/10.3233/FAIA201017, https://doi.org/10.3233/FAIA201017
4. Buchberger, B.: Bruno Buchberger's PhD Thesis 1965: an algorithm for finding the basis elements of the residue class ring of a zero dimensional polynomial ideal. J. Symbol. Comput. **41**(3-4), 475–511 (2006). https://doi.org/10.1016/J.JSC.2005.09.007
5. Dutertre, B.: Yices 2.2. In: Biere, A., Bloem, R. (eds.) CAV 2014. LNCS, vol. 8559, pp. 737–744. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-08867-9_49
6. Dutertre, B., Goel, A., Graham-Lengrand, S., Irfan, A., Jovanovic, D., Mason, I.A.: Yices 2 in SMT-COMP 2023 (2023)
7. Goldwasser, S., Micali, S., Rackoff, C.: The knowledge complexity of interactive proof systems. SIAM J. Comput. **18**(1), 186–208 (1989). https://doi.org/10.1137/0218012

8. Graham-Lengrand, S., Jovanović, D., Dutertre, B.: Solving Bitvectors with MCSAT: explanations from bits and pieces. In: Peltier, N., Sofronie-Stokkermans, V. (eds.) IJCAR 2020. LNCS (LNAI), vol. 12166, pp. 103–121. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-51074-9_7
9. Hader, T., Kaufmann, D., Kovács, L.: SMT solving over finite field arithmetic. In: Piskac, R., Voronkov, A. (eds.) Intl. Conf. on Logic for Programming, Artificial Intelligence and Reasoning (LPAR). EPiC Series in Computing, vol. 94, pp. 238–256. EasyChair (2023). https://doi.org/10.29007/4N6W, https://doi.org/10.29007/4n6w
10. Jovanović, D.: Solving nonlinear integer arithmetic with MCSAT. In: Bouajjani, A., Monniaux, D. (eds.) VMCAI 2017. LNCS, vol. 10145, pp. 330–346. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-52234-0_18
11. Jovanovic, D., Barrett, C., de Moura, L.: The design and implementation of the model constructing satisfiability calculus. In: International Conference on Formal Methods in Computer-Aided Design (FMCAD), pp. 173–180. IEEE (2013).https://doi.org/10.1109/FMCAD.2013.7027033
12. Jovanovic, D., Dutertre, B.: Libpoly: a library for reasoning about polynomials. In: Brain, M., Hadarean, L. (eds.) Intl. Workshop on Satisfiability Modulo Theories (SMT). CEUR Workshop Proceedings, vol. 1889, pp. 28–39. CEUR-WS.org (2017). https://ceur-ws.org/Vol-1889/paper3.pdf
13. Jovanović, D., de Moura, L.: Solving non-linear arithmetic. In: Gramlich, B., Miller, D., Sattler, U. (eds.) IJCAR 2012. LNCS (LNAI), vol. 7364, pp. 339–354. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-31365-3_27
14. de Moura, L., Jovanović, D.: A model-constructing satisfiability calculus. In: Giacobazzi, R., Berdine, J., Mastroeni, I. (eds.) VMCAI 2013. LNCS, vol. 7737, pp. 1–12. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-35873-9_1
15. de Moura, L.M., Bjørner, N.S.: Satisfiability modulo theories: introduction and applications. Commun. ACM **54**(9), 69–77 (2011)
16. Nieuwenhuis, R., Oliveras, A., Tinelli, C.: Abstract DPLL and abstract DPLL modulo theories. In: Baader, F., Voronkov, A. (eds.) LPAR 2005. LNCS (LNAI), vol. 3452, pp. 36–50. Springer, Heidelberg (2005). https://doi.org/10.1007/978-3-540-32275-7_3
17. Ozdemir, A., Kremer, G., Tinelli, C., Barrett, C.W.: Satisfiability modulo finite fields. In: International Conference on Computer Aided Verification (CAV), Part II. LNCS, vol. 13965, pp. 163–186. Springer (2023).https://doi.org/10.1007/978-3-031-37703-7_8
18. Ozdemir, A., Wahby, R.S., Brown, F., Barrett, C.W.: Bounded verification for finite-field-blasting - in a compiler for zero knowledge proofs. In: Enea, C., Lal, A. (eds.) International Conference on Computer Aided Verification (CAV), Part III. LNCS, vol. 13966, pp. 154–175. Springer (2023).https://doi.org/10.1007/978-3-031-37709-9_8
19. Szabo, N.: Smart Contracts: Building Blocks for Digital Markets (1996). http://www.fon.hum.uva.nl
20. Wang, D.: Elimination Methods. Springer Science & Business Media (2001)