Combinatory Array Logic with Sums

Rodrigo Raya

School of Computer and Communication Science, École Polytechnique Fédérale de Lausanne, Switzerland

Abstract

We prove an NP upper bound on a theory of integer-indexed integer-valued arrays that extends combinatory array logic with an ordering relation on the index set and the ability to express sums of elements. We compare our fragment with seven other fragments in the literature in terms of their expressiveness and computational complexity.

Keywords

array theories, computational complexity, model theory

1. Introduction

Decision procedures for array theories have been widely investigated in the last decades motivated by their application in software verification. [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12] Despite the variety of array theories available mainstream SMT solvers such as Z3 [13] and the CVC family [14] focus on the theory of combinatory array logic [6].

The reason why these solvers focus on this particular fragment may lie on the fact that it retains completeness guarantees for satisfiability checking of quantifier-free formulae even in the presence of model-based theory combination [15] with other decidable theories. This property has been exploited to develop verification algorithms for imperative programs that report counterexamples [16]. It is thus natural to ask to what extent the NP complexity upper bound proved by de Moura and Bjorner extends when combining combinatory array logic to other theories of interest.

In [12], we lifted the restrictions mentioned in [6] showing that combinatory array logic can be extended with a cardinality operator while still retaining a NP complexity upper bound. In essence, the results of [12] can be seen as a decomposition theorem in the style of Feferman and Vaught [17], with the remarkable difference that the results hold for the complexity class NP rather than for the decidable class. In particular, we proved that the existential first-order theory of a power structure [18] whose index set contains the equicardinality operator can be decomposed into the existential first-order theory of the elements and the weak monadic second-order theory of the indices.

This paper builds on the observation that multiset theories studied for instance in [19], are representable in combinatory array logic since the basic operations on multisets are pointwise operations on arrays of natural numbers. We then leverage the techniques of [20] to lift a

rodrigo.raya@epfl.ch (R. Raya)
0000-0002-0866-9257 (R. Raya)

restriction mentionned in [6]: that cardinalities of multisets cannot be encoded in combinatory array logic. When viewed as arrays of integers, the cardinalities of multisets correspond to the sum of the elements in the arrays, a very useful specification construct in applications such as the verification of smart contracts [21]. We then show that we can combine the resulting fragment with classical combinatory array logic and that the satisfiability problem of the resulting combination is in the complexity class NP.

Organisation of the paper. Section 2 introduces the logical theories that we use in our developments, previously obtained theory combination results and fixes notation for the remaining of the paper. Section 3 describes the language tackled and the strategy used to prove the complexity bound. Section 6 gives the extension with sums and proves the NP-complexity result. Section 7 compares the resulting theory with other seven existing array theories in the literature. Section 8 concludes the paper.

2. Preliminaries

2.1. Basic Logical Theories

Our result builds upon the quantifier-free theory of Boolean algebra and Presburger arithmetic [22], QFBAPA, which has been studied in the context of data structure verification [23]. The syntax of QFBAPA is given in Figure 1. The meaning of the syntax is as follows. F presents the Boolean structure of the formula, A stands for the top-level constraints, B gives the Boolean restrictions and T the Presburger arithmetic terms. $\mathcal{U}$ represents the universal set

$$
\begin{aligned}
F &::= A \mid F_1 \wedge F_2 \mid F_1 \vee F_2 \mid \neg F \\
A &::= B_1 = B_2 \mid B_1 \subseteq B_2 \mid T_1 = T_2 \mid T_1 \leq T_2 \mid K \text{ dvd } T \\
B &::= x \mid \emptyset \mid \mathcal{U} \mid B_1 \cup B_2 \mid B_1 \cap B_2 \mid B^c \\
T &::= k \mid K \mid T_1 + T_2 \mid K \cdot T \mid |B| \\
K &::= \ldots \mid -2 \mid -1 \mid 0 \mid 1 \mid 2 \mid \ldots
\end{aligned}
$$

Figure 1: QFBAPA's syntax

The satisfiability problem of this logic is reducible to propositional satisfiability in polynomial time.

Theorem 1 ([22]). *The satisfiability problem for QFBAPA is in NP.*

2.2. Previously Obtained Combination Results

In [12], we observed that combinatory array logic [6] can be encoded in the existential fragment of the first-order theory of a power structure $\mathscr{M}^I$ [18], i.e. a structure whose elements are functions from an index set I to a carrier set M and where the function and relation symbols are interpreted point-wise. We proved the following result

Theorem 2 ([12]). *$Th_{\exists^*}(\mathscr{M}) \in NP$ if and only if $Th_{\exists^*}(\mathscr{M}^I) \in NP$.*

We then extended this language with the ability to express cardinalities of sets of indices, which resulted in a new theory termed QFBAPAI (interpreted QFBAPA) and whose formulae are of the form

$$\exists c_1, \ldots, c_m. \exists x_1, \ldots, x_n. \\ F(S_1, \ldots, S_k) \wedge \bigwedge_{i=1}^{k} S_i = \{r \in I \mid \varphi_i(x_1(r), \ldots, x_n(r), c_1, \ldots, c_m)\}$$

where the letters c_i represent constants, the letters x_i represent power structure variables and φ_i are formulae interpreted over the theory of the elements. By analogy to Theorem 2, we proved that

Theorem 3 ([12]). *$Th_{\exists^*}(\mathscr{M}) \in NP$ if and only if QFBAPAI $\in NP$.*

Finally, we showed how to encode combinatory array logic (CAL) in QFBAPAI with a worst-case log-quadratic increase in formula size.

Proposition 4 ([12]). *A CAL formula ψ can be encoded in QFBAPAI as a formula of size $O(|\psi|^2 \log_2(|\psi|))$.*

In this paper, we reuse this encoding and show that in the target language, one may also support a summation operator. This allows to effectively support such summation operators in the language of combinatory array logic.

2.3. Further Notations for the Rest of the Paper

Tuples and powers. We will abbreviate tuples $(x_1, \ldots, x_m)$ as $\overline{x}$. We fix a structure $\mathscr{M}$ for the contents of power structure variables and an index set I for the indices. If $\overline{x}$ denotes a power structure variable, we will denote by $\overline{x}(n)$ the n-th column of $\overline{x}(n)$, i.e. the tuple $(x_1(n), \ldots, x_m(n))$.
Set and multiset constructions. As usual, set comprehension is denoted with curly braces $\{i|\varphi(i)\}$ where φ_i is a formula written in first-order logic. Multiset comprehension is written by $(\!|i|\varphi(i)|\!)$ where φ is a formula written in first-order logic.

Given a set of integers A and an integer $m \in \mathbb{N}$, its m-th sum iteration [24] is the set

$$A^m = \{u \mid x_1, \ldots, x_m \in A. u = \sum_{i=1}^{m} x_i\}$$

Similarly, the arbitrary sum iteration is the set

$$A^* = \{u \mid \exists m \geq 0, x_1, \ldots, x_m \in A. u = \sum_{i=1}^{m} x_i\}$$

Cardinalities of sets and multisets. If S is a set then $|S|$ is the number of elements in S and if S is a multiset then $|S|$ is the number of elements in S counted with the corresponding multiplicity.

3. Overview of the Reduction Procedure

Our decision procedure works by reduction to QFBAPAI, that is, we perform a sequence of transformations on the input formula which preserve equivalence, until we arrive to an equivalent formula in the language of interpreted QFBAPA. Note that similar ideas appear in the literature under the name of BAPA reduction [25].

Our first step is thus defining the input language to be transformed into QFBAPAI.

Definition 5. *The theory QFBAPA-Power-Sum consists of formulae of the form*

$$F(S_1, \ldots, S_k, \overline{\sigma}) \wedge \bigwedge_{i=1}^{k} S_i = \{i \in I | \varphi_i(\overline{c}(i))\} \wedge \overline{\sigma} = \sum_{i \in I} (\!|\overline{c}(i) \mid \varphi_0(\overline{c}(i))|\!) \tag{1}$$

where F is a formula from QFBAPA, $\varphi_0, \ldots, \varphi_k$ are formulae from a logic whose satisfiability problem is decidable in NP and $\overline{c}$ is a tuple of arrays of natural numbers.

Note that the assumption of the formulae $\varphi_0, \ldots, \varphi_k$ having a satisfiability problem in NP can be removed [26, 27]. But this assumption is standard in satisfiability modulo theories solvers.

Our reduction will first transform the sum operator into the finite power of a set (see Section 2.3). Then, this finite power will be converted into a formula of Presburger arithmetic. The resulting constraint will be in the language of QFBAPAI and by the results in [12, 28] its satisfiability problem will be decidable in NP.

4. Elimination of the Summation Operator

We write equation 1 as:

$$H(S_1, \ldots, S_k, \overline{c}, \overline{\sigma}) \wedge \overline{\sigma} = \sum_{i \in I} (\!|\overline{c}(i) | \varphi_0(\overline{c}(i))|\!)$$

For handling the summation construct, we will adapt some of the techniques developed in [19, Chapter 2] to the case where, as in the case of combinatory array logic, the index set is arbitrary.

Proposition 6 (Multiset elimination)**.** *The formula*

$$H(S_1, \ldots, S_k, \overline{c}, \overline{\sigma}) \wedge \overline{\sigma} = \sum_{i \in I} (\!|\overline{c}(i) \mid \varphi_0(\overline{c}(i))|\!) \tag{2}$$

and the formula

$$H(S_1, \ldots, S_k, \overline{c}, \overline{\sigma}) \wedge \exists x. \overline{\sigma} \in \{\overline{k} \mid \varphi_0(\overline{k})\}^x \wedge ((x < |I| \wedge \exists \overline{c}_0. \neg \varphi_0(\overline{c}_0)) \vee x = |I|) \tag{3}$$

are equivalent.

Proof. $\Rightarrow$) If 2) is satisfied there are $\overline{\sigma}, S_1, \ldots, S_k$ and $\overline{c}$ such that:

$$H(S_1, \ldots, S_k, \overline{c}, \overline{\sigma}) \wedge \overline{\sigma} = \sum_{i \in I} (\!|\overline{c}(i) \mid \varphi_0(\overline{c}(i))|\!)$$

We claim that $\overline{\sigma}$ satisfies formula 3). By hypothesis, $H(S_1, \ldots, S_k, \overline{c}, \overline{\sigma})$ is true. Moreover, from the equality $\overline{\sigma} = \sum_{i \in I} (\!|\overline{c}(i) \mid \varphi(\overline{c}(i))|\!)$ follows that $\overline{\sigma}$ is the sum of some number $x \in \mathbb{N}$ of terms $\overline{c}$ satisfying

$\varphi(\bar{c})$, since $\overline{\sigma}$ is a natural number. Thus, $\overline{\sigma} \in \{\bar{k} \mid \varphi(\bar{k})\}^x$. If $x = |I|$ we are done. Otherwise, $x < |I|$, and it must be that the remaining positions of $\bar{c}$ are filled with values $\bar{c}_0$ that do not satisfy φ_0. Thus, $\exists \bar{c}_0.\neg\varphi_0(\bar{c}_0)$.
$\Leftarrow$) If formula 3) is satisfied then there are values $\overline{\sigma}, S_1, \ldots, S_k, x$ such that:

$$H(S_1, \ldots, S_k, \bar{c}, \overline{\sigma}) \wedge \exists x.\overline{\sigma} \in \{\bar{k} \mid \varphi_0(\bar{k})\}^x \wedge ((x < |I| \wedge \exists \bar{c}_0.\neg\varphi_0(\bar{c}_0)) \vee x = |I|)$$

It follows that there is a list of x elements $\bar{c}_i$ such that $\overline{\sigma} = \sum_{i=1}^{x} \bar{c}_i$ and such that $\varphi_i(\bar{c}_i)$ holds for each $i \in \{1, \ldots, x\}$. If $x = |I|$ then we define

$$\bar{c}(i) = \left\{ \bar{k}_i \quad \text{if } 1 \leq i \leq x \right.$$

Otherwise, $x < |I|$ and by hypothesis, there exist $\bar{c}_0$ such that $\neg\varphi(\bar{c}_0)$. In this case, we define:

$$\bar{c}(i) = \begin{cases} \bar{c}_i & \text{if } 1 \leq i \leq x \\ \bar{c}_0 & \text{otherwise} \end{cases}$$

It is immediate that $\overline{\sigma}, S_1, \ldots, S_k$ and $\bar{c}$ satisfy $H(S_1, \ldots, S_k, \bar{c}, \overline{\sigma}) \wedge \overline{\sigma} = \sum_{i \in I} (\!|\bar{c}(i) \mid \varphi_0(\bar{c}(i))|\!)$. □

Note that, in the proof of Proposition 6 it is key that the values of $\bar{c}$ are natural numbers, because then, by the finiteness of the total sum value $\overline{\sigma}$, we can deduce that there are a finite number of addends x. If the values of the arrays were arbitrary integers we could have an array whose i-th entry was $x[i] = (-1)^i$ for each $i \in \mathbb{N}$. Here the sum is finite but the number of addends is infinite.

On the other hand, observe that this is not actually limiting since we could impose a finiteness restriction on $(\!|\bar{c}(i) \mid \varphi(\bar{c}(i))|\!)$. But we will continue to assume that the entries are natural numbers to simplify the presentation.

Finally, note the term $x = |I|$ in the formula formula 3) can be removed if $|I|$ is infinite, since x is always a finite value.

5. Elimination of the Exponent Cardinalities

So far, we have transformed our formula into the form

$$H(S_1, \ldots, S_k, \bar{c}, \overline{\sigma}) \wedge \exists x.\overline{\sigma} \in \{\bar{k} \mid \varphi_0(\bar{k})\}^x \wedge ((x < |I| \wedge \exists \bar{c}_0.\neg\varphi_0(\bar{c}_0)) \vee x = |I|)$$

distributing the existential quantifiers, we have equivalently

$$H(S_1, \ldots, S_k, \bar{c}, \overline{\sigma}) \wedge \exists x, \overline{c_0}.\overline{\sigma} \in \{\bar{k} \mid \varphi_0(\bar{k})\}^x \wedge ((x < |I| \wedge \neg\varphi_0(\bar{c}_0)) \vee x = |I|)$$

Next, we show that

Proposition 7. *The formula $\exists x, \overline{c_0}.\overline{\sigma} \in \{\bar{k} | \varphi_0(\bar{k})\}^x \wedge \phi(x, \overline{c_0})$ where $\phi(x, \overline{c_0})$ is a formula of quantifier-free Presburger arithmetic is equivalent to a polynomial sized quantifier-free Presburger arithmetic formula $\psi(\overline{\sigma})$.*

The proof is essentially that of [19, Theorem 2.23] with the necessary adaptations to account for the integer variables on the exponents. We use the left to right implication to find what should the formula ψ be and the right to left implication to confirm that it is indeed equivalent.

$\Rightarrow$) If formula 3) is satisfied then we first use the semilinear normal form theorem [19, Theorem 2.13] to express the condition $\varphi_0(\overline{k})$ equivalently as

$$\exists \lambda_{11}, \dots, \lambda_{mq_m}. \bigvee_{i=1}^{m} \left(\overline{k} = a_i + \sum_{j=1}^{q_i} \lambda_{ij} b_{ij} \right) \tag{4}$$

where $m, q_1, \dots, q_m \in \mathbb{N}$ and vectors $a_i, b_{ij} \in \mathbb{N}^n$ for $1 \leq j \leq q_i, 1 \leq i \leq m$, $\|a_i\|_1, \|b_{ij}\|_1 \leq 2^{p(s)}$ and p is a polynomial. Next, we eliminate the star operator using [24, Proposition 2], obtaining that the condition $\overline{\sigma} \in \{\overline{k} \mid \varphi_0(\overline{k})\}^x$ is equivalent to

$$\exists \overline{\mu}, \overline{\lambda}. \overline{\sigma} = \sum_{i=1}^{q} \left(\mu_i a_i + \sum_{j=1}^{q_i} \lambda_{ij} b_{ij} \right) \wedge \bigwedge_{i=1}^{q} \left(\mu_i = 0 \implies \sum_{j=1}^{q_i} \lambda_{ij} = 0 \right) \wedge x = \sum_{i=1}^{q} \mu_i \tag{5}$$

We express the resulting vector $\overline{\sigma}$ with polynomially many generators using [19, Theorem 2.20] and [19, Lemma 2.21]. We get that (5) is equivalent to

$$\exists \overline{\lambda}, \overline{\nu}. \overline{\sigma} = \sum_{i=1}^{p(s)} \lambda_i \nu_i \wedge \bigwedge_{i=1}^{p(s)} \varphi_0(\nu_i)$$

where p is a polynomial in the size s of the original formula. Using the results of [19, Theorem 2.22] and [19, Theorem 2.23], we can further eliminate the product $\lambda_i \nu_i$. The resulting formula is of the form

$$\exists \overline{\lambda}, \overline{\nu}. \overline{\sigma} = \sum_{i=1}^{p(s)} t_i \wedge \bigwedge_{i=1}^{p(s)} \varphi_0(\nu_i) \wedge \phi(\overline{c_0}, x)$$

where $t_i = \sum_{j=0}^{t(s)} 2^j ite(\lambda_{ij}, x_i, 0)$ where $t(s)$ is a polynomial in the size s of the original formula and λ_{ij} is the j-th bit of the number λ_i. This formula is in quantifier-free Presburger arithmetic and its equivalent to the original so we conclude by setting

$$\exists \overline{\lambda}, \overline{\nu}, \overline{c_0}, x. \psi(\overline{\sigma}) = \sum_{i=1}^{p(s)} t_i \wedge \bigwedge_{i=1}^{p(s)} \varphi_0(\nu_i) \wedge \phi(\overline{c_0}, x) \tag{6}$$

By construction, a solution of formula (6) readily gives a solution of the original formula.

6. Combination with Power Structures

Using the previous reductions, we have arrived at a formula of the form.

$$H(S_1, \dots, S_k, \overline{c}, \overline{\sigma}) \wedge \psi(\overline{\sigma})$$

However, this formula falls in the fragment QFBAPAI studied in [12, 28], and thus we conclude

Theorem 8. *The complexity of deciding satisfiability of QFBAPA-Power-Sum is in NP.*

7. Comparison with other Array Theories

Traditionally, research on decision procedures for array theories has focused on providing concrete algorithms for syntactically presented theories. As a result, works on, in particular, combinatory array logic, are often unaware of each other which results in duplicated engineering work and the lack of a comparison between potentially different implementation techniques. Meseguer [29] has also expressed similar remarks about syntactically presented theories.

One advantage of our semantic approach is that one has a clear comparison criteria between the different theories proposed in the literature. Figure 2 lists some related theories of arrays in the literature ordered by expressivity. The satisfiability problem of these logics is in NP for all but the array fold logic (AFL) fragment which in general lies in PSPACE.

An important characteristic of these theories is the intended interpretation of the index and element set theories. As presented in [10, 11] it could seem that the flat array subfragment and the Π-logic theory are restricted to integer indices or elements. The results on QFBAPAI show that this restriction is not necessary. However, RegExp-QFBAPA-Power already requires the index set to be the integers numbers and RegExp-QFBAPA-Power-Sum and AFL require both the index and element set to be the integers.

It seems that sum constraints were of interest to the authors of [9] since the repository of the tool AFOLDER [30] contains some examples using sums. However, array fold logic as presented in [9] cannot express sums since counters can only be updated adding constants. Thus, to the best of our knowledge, the theory of arrays that we present in this paper is the first array theory whose satisfiability problem is decidable in non-deterministic polynomial time and is able to express sum constraints.

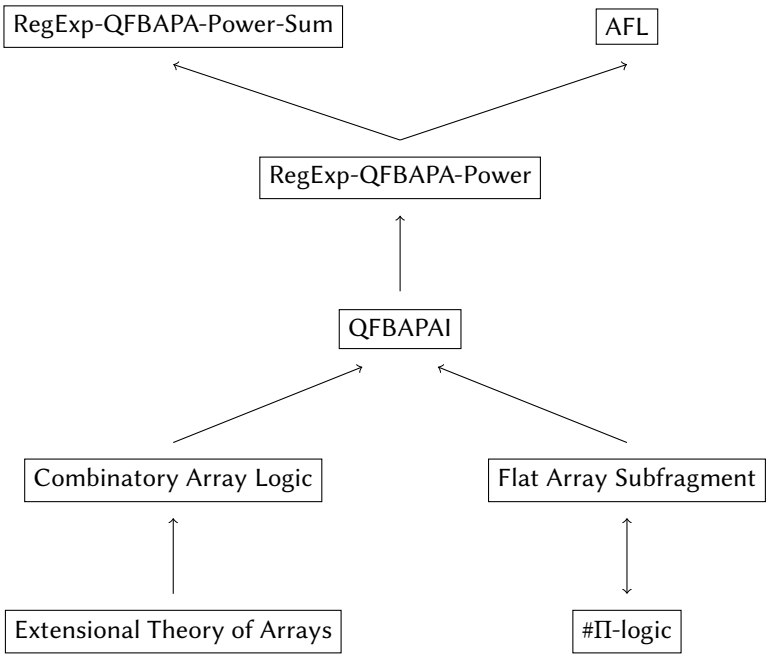

Figure 2: Treated theories of arrays sorted by expressivity.

Although we have not focused our discussion on applications it seems that users of combinatory array logic (for instance [31, 32]) could benefit of support for sum constraints in a variety of scenarios. One possible application of such constraints could be in the verification of smart contracts [33, 34, 21, 35, 36].

Finally, it is worth noting that there has been recent progress in the automation of linear arithmetic with stars [37]. An adaption of those techniques could be a good starting point for the implementation of decision procedures for QFBAPA-Power-Sum.

8. Conclusion

We have described an extension of combinatory array logic that is able to express sums of elements and proved that the associated satisfiability problem of the fragment lies in the NP complexity class. Our result combines the method of [12] with those in [38] and [19, Chapter 2]. Our focus has been on the semantic characterisation of the fragments rather than in providing deterministic decision procedures. Future work could include exploring the practical implementation of the fragment as well as extending the classification of Figure 2 to other theories considered in the literature.

More broadly, our paper completes the observations made in [12, 28, 26, 27, 39] that the calculus of data structures can be extended to support the verification of structures holding data, as opposed to sets and multisets, by demonstrating how other aggregators, besides cardinalities, can be supported by the calculus. In unpublished work [40] Suter and Szajkó have discussed the introduction of other operators such as minimum and maximum, which further confirms our thesis that the calculus is an adequate framework for designing decision procedures for the automatic verification of real-world data structures.

References

[1] James Cornelius King, A program verifier, Ph.D. thesis, Carnegie-Mellon University, Pittsburgh Pennsylvania USA, 1969. URL: https://apps.dtic.mil/sti/citations/AD0699248, section: Technical Reports.

[2] A. Stump, C. Barrett, D. Dill, J. Levitt, A decision procedure for an extensional theory of arrays, in: Proceedings 16th Annual IEEE Symposium on Logic in Computer Science, IEEE Comput. Soc, Boston, MA, USA, 2001, pp. 29–37. URL: http://ieeexplore.ieee.org/document/932480/. doi:10.1109/LICS.2001.932480.

[3] A. R. Bradley, Z. Manna, H. B. Sipma, What's Decidable About Arrays?, in: Verification, Model Checking, and Abstract Interpretation, Lecture Notes in Computer Science, Springer, Berlin, Heidelberg, 2006, pp. 427–442. doi:10.1007/11609773_28.

[4] S. Ghilardi, E. Nicolini, S. Ranise, D. Zucchelli, Decision procedures for extensions of the theory of arrays, Annals of Mathematics and Artificial Intelligence 50 (2007) 231–254. URL: https://doi.org/10.1007/s10472-007-9078-x. doi:10.1007/s10472-007-9078-x.

[5] P. Habermehl, R. Iosif, T. Vojnar, What Else Is Decidable about Integer Arrays?, in: Foundations of Software Science and Computational Structures, Lecture Notes in Computer

Science, Springer, Berlin, Heidelberg, 2008, pp. 474–489. doi:10.1007/978-3-540-78499-9_33.

[6] L. de Moura, N. Bjorner, Generalized, efficient array decision procedures, in: 2009 Formal Methods in Computer-Aided Design, IEEE, Austin, TX, 2009, pp. 45–52. doi:10.1109/FMCAD.2009.5351142.

[7] F. Alberti, S. Ghilardi, N. Sharygina, Booster: An Acceleration-Based Verification Framework for Array Programs, in: F. Cassez, J.-F. Raskin (Eds.), Automated Technology for Verification and Analysis, Lecture Notes in Computer Science, Springer International Publishing, Cham, 2014, pp. 18–23. doi:10.1007/978-3-319-11936-6_2.

[8] F. Alberti, S. Ghilardi, N. Sharygina, Decision Procedures for Flat Array Properties, Journal of Automated Reasoning 54 (2015) 327–352. doi:10.1007/s10817-015-9323-7.

[9] P. Daca, T. A. Henzinger, A. Kupriyanov, Array Folds Logic, in: Computer Aided Verification, Lecture Notes in Computer Science, Springer International Publishing, Cham, 2016, pp. 230–248. doi:10.1007/978-3-319-41540-6_13.

[10] K. v. Gleissenthall, N. Bjørner, A. Rybalchenko, Cardinalities and universal quantifiers for verifying parameterized systems, in: Proceedings of the 37th ACM SIGPLAN Conference on Programming Language Design and Implementation, PLDI '16, Association for Computing Machinery, New York, NY, USA, 2016, pp. 599–613. URL: https://doi.org/10.1145/2908080.2908129. doi:10.1145/2908080.2908129.

[11] F. Alberti, S. Ghilardi, E. Pagani, Cardinality constraints for arrays (decidability results and applications), Formal Methods in System Design 51 (2017) 545–574. URL: https://doi.org/10.1007/s10703-017-0279-6. doi:10.1007/s10703-017-0279-6.

[12] R. Raya, V. Kunčak, NP Satisfiability for Arrays as Powers, in: Verification, Model Checking, and Abstract Interpretation, Lecture Notes in Computer Science, Springer International Publishing, Cham, 2022, pp. 301–318. URL: https://link.springer.com/chapter/10.1007/978-3-030-94583-1_15. doi:10.1007/978-3-030-94583-1_15.

[13] L. de Moura, N. Bjørner, Z3: An Efficient SMT Solver, in: Tools and Algorithms for the Construction and Analysis of Systems, volume 4963 of *Lecture Notes in Computer Science*, Springer Berlin Heidelberg, Berlin, Heidelberg, 2008, pp. 337–340. doi:10.1007/978-3-540-78800-3_24.

[14] H. Barbosa, C. Barrett, M. Brain, G. Kremer, H. Lachnitt, M. Mann, A. Mohamed, M. Mohamed, A. Niemetz, A. Nötzli, A. Ozdemir, M. Preiner, A. Reynolds, Y. Sheng, C. Tinelli, Y. Zohar, cvc5: A Versatile and Industrial-Strength SMT Solver, in: D. Fisman, G. Rosu (Eds.), Tools and Algorithms for the Construction and Analysis of Systems, Lecture Notes in Computer Science, Springer International Publishing, Cham, 2022, pp. 415–442. doi:10.1007/978-3-030-99524-9_24.

[15] L. de Moura, N. Bjørner, Model-based Theory Combination, Electronic Notes in Theoretical Computer Science 198 (2008) 37–49. URL: https://linkinghub.elsevier.com/retrieve/pii/S1571066108002946. doi:10.1016/j.entcs.2008.04.079.

[16] G. S. Schmid, Scaling Language Features for Program Verification, Ph.D. thesis, EPFL, Lausanne, 2022. doi:10.5075/epfl-thesis-8030.

[17] S. Feferman, R. Vaught, The first order properties of products of algebraic systems, Fundamenta Mathematicae 47 (1959) 57–103.

[18] W. Hodges, Model Theory, Encyclopedia of Mathematics and its Applications, Cambridge

University Press, Cambridge, 1993.
[19] R. Piskac, Decision Procedures for Program Synthesis and Verification, Ph.D. thesis, EPFL, Lausanne, 2011.
[20] R. Piskac, V. Kunčak, Linear Arithmetic with Stars, in: Computer Aided Verification, Lecture Notes in Computer Science, Springer, Berlin, Heidelberg, 2008, pp. 268–280. doi:10.1007/978-3-540-70545-1_25.
[21] N. Elad, S. Rain, N. Immerman, L. Kovács, M. Sagiv, Summing up Smart Transitions, in: A. Silva, K. R. M. Leino (Eds.), Computer Aided Verification, Lecture Notes in Computer Science, Springer International Publishing, Cham, 2021, pp. 317–340. doi:10.1007/978-3-030-81685-8_15.
[22] V. Kunčak, M. Rinard, Towards Efficient Satisfiability Checking for Boolean Algebra with Presburger Arithmetic, in: Automated Deduction – CADE-21, Lecture Notes in Computer Science, Springer, Berlin, Heidelberg, 2007, pp. 215–230. doi:10.1007/978-3-540-73595-3_15.
[23] V. Kunčak, Modular Data Structure Verification, Ph.D. thesis, Massachusetts Institute of Technology, 2007.
[24] D. Lugiez, S. D. Zilio, Multitrees Automata, Presburger's Constraints and Tree Logics, Technical Report 08-2002, Laboratoire d'Informatique Fondamentale de Marseille, 2002.
[25] T. Wies, R. Piskac, V. Kunčak, Combining Theories with Shared Set Operations, in: Frontiers of Combining Systems, Lecture Notes in Computer Science, Springer, Berlin, Heidelberg, 2009, pp. 366–382. doi:10.1007/978-3-642-04222-5_23.
[26] R. Raya, J. Hamza, V. Kunčak, On the Complexity of Convex and Reverse Convex Prequadratic Constraints, in: EPiC Series in Computing, volume 94, EasyChair, 2023, pp. 350–368. URL: https://easychair.org/publications/paper/T6kG. doi:10.29007/wdd7, iSSN: 2398-7340.
[27] R. Raya, The Complexity of Checking Non-Emptiness in Symbolic Tree Automata, 2023. URL: http://arxiv.org/abs/2311.05250, arXiv:2311.05250 [cs].
[28] R. Raya, V. Kunčak, On algebraic array theories, Journal of Logical and Algebraic Methods in Programming 136 (2024) 100906. URL: https://www.sciencedirect.com/science/article/pii/S2352220823000603. doi:https://doi.org/10.1016/j.jlamp.2023.100906.
[29] J. Meseguer, Lecture Notes on Topics in Automated Reasoning, 2017.
[30] pdaca, AFolder - Solver for Array Folds Logic., 2022. URL: https://github.com/pdaca/AFolder, original-date: 2016-03-22T15:29:30Z.
[31] G. S. Schmid, V. Kunčak, Generalized Arrays for Stainless Frames, in: Verification, Model Checking, and Abstract Interpretation, Lecture Notes in Computer Science, Springer International Publishing, Cham, 2022, pp. 332–354. doi:10.1007/978-3-030-94583-1_17.
[32] S. Pirelli, A. Valentukonytė, K. Argyraki, G. Candea, Automated Verification of Network Function Binaries, USENIX Association, Renton, WA, 2022, pp. 585–600. URL: https://www.usenix.org/conference/nsdi22/presentation/pirelli.
[33] A. Permenev, D. Dimitrov, P. Tsankov, D. Drachsler-Cohen, M. Vechev, VerX: Safety Verification of Smart Contracts, in: 2020 IEEE Symposium on Security and Privacy (SP), 2020, pp. 1661–1677. doi:10.1109/SP40000.2020.00024, iSSN: 2375-1207.
[34] E. Albert, J. Correas, P. Gordillo, G. Román-Díez, A. Rubio, GASOL: Gas Analysis and Optimization for Ethereum Smart Contracts, in: Tools and Algorithms for the Con-

struction and Analysis of Systems, volume 12079, Springer International Publishing, Cham, 2020, pp. 118–125. URL: http://link.springer.com/10.1007/978-3-030-45237-7_7. doi:10.1007/978-3-030-45237-7_7, series Title: Lecture Notes in Computer Science.

[35] L. Alt, M. Blicha, A. E. J. Hyvärinen, N. Sharygina, SolCMC: Solidity Compiler's Model Checker, in: S. Shoham, Y. Vizel (Eds.), Computer Aided Verification, Lecture Notes in Computer Science, Springer International Publishing, Cham, 2022, pp. 325–338. doi:10.1007/978-3-031-13185-1_16.

[36] R. Otoni, M. Marescotti, L. Alt, P. Eugster, A. Hyvärinen, N. Sharygina, A Solicitous Approach to Smart Contract Verification, ACM Transactions on Privacy and Security 26 (2023) 15:1–15:28. URL: https://doi.org/10.1145/3564699. doi:10.1145/3564699.

[37] M. Levatich, N. Bjørner, R. Piskac, S. Shoham, Solving LIA* Using Approximations, in: D. Beyer, D. Zufferey (Eds.), Verification, Model Checking, and Abstract Interpretation, Lecture Notes in Computer Science, Springer International Publishing, Cham, 2020, pp. 360–378. doi:10.1007/978-3-030-39322-9_17.

[38] R. Piskac, V. Kunčak, Decision Procedures for Multisets with Cardinality Constraints, in: Verification, Model Checking, and Abstract Interpretation, Lecture Notes in Computer Science, Springer, Berlin, Heidelberg, 2008, pp. 218–232. doi:10.1007/978-3-540-78163-9_20.

[39] R. Raya, Decision Procedures for Power Structures, Ph.D. thesis, EPFL, 2023.

[40] A. Szajkó, P. Suter, Complete Procedures for Ordered Data Structures, Technical Report, 2012.