On aggregation in parametric array theories

Rodrigo Raya[1]*

Max-Planck Institute for Software Systems, Kaiserslautern, Germany
rraya@mpi-sws.org

Abstract

We prove an NP upper bound on a theory of integer-indexed integer-valued arrays that extends combinatory array logic with the ability to express sums of elements. The decision procedure that we give is based on observations obtained from our analysis of the theory of power structures.

1 Introduction

Many applications of computer science to operations research and software engineering require some form of constraint solving technology. We focus in the satisfiability modulo theories (SMT) framework which was intensively developed in the first decade of the century, leveraging progress in the architecture of propositional satisfiability solvers [19, 21, 8].

SMT addresses the satisfiability problem of fragments of first-order theories that are quantifier-free or have a small number of quantifier alternations. In fortunate occasions, this restriction makes the satisfiability problem NP-complete. In such cases, it is possible to reduce the satisfiability problem of the fragments to the satisfiability problem of propositional logic in polynomial time. Some theories supported using such reduction include real numbers, integers, lists, arrays, bit vectors, and strings [2, 15].

This work analyses the structure of a well-known fragment of the quantifer-free theory of arrays. In the SMT framework, arrays are conceived as indexed homogeneous collections of elements from some fixed domain. This is in contrast to other data-structures, like lists, which can only be accessed with recursive operators. The popularity of arrays stems from the fact that they can be used to model many abstractions useful in applications such as programming [6, 30], databases [12, 9], model checkers [11], memory models [5] or quantum circuits [4].

Several theories of arrays in the literature express essentially the same concepts under different syntactic appearances [29, 7, 13, 1]. As a consequence, a systematic classification of these theories is becoming increasingly difficult. This results in duplicated engineering efforts. It has been argued [18, Lecture 19] that some of these redundancies could be avoided by adopting a semantic perspective on the study of SMT theories.

Our results show that the semantic approach is fruitful in the area of decision procedures for theories of arrays. We demonstrated in [26, 27] how, by fixing a model of such theories, we are able to reconstruct and extend the celebrated combinatory array logic fragment [3]. In this paper, we further show how these observations can be extended to support summation constraints. Our methodology is inspired in the model theory of power structures [20, 10], which we adapt from the first-order to the quantifier-free setting, which is the one relevant for applications to SMT.

*Research supported by the Swiss NSF Project P500PT_222338

2 First-order model theory

We start reviewing some notions from first-order model theory.

A **first-order language** is one whose logical symbols are $\neg, \wedge, \vee, \forall$ and $\exists$, whose terms are either variables, constants or function symbols applied to terms and whose formulas are either atomic (relation symbols applied to terms) or general (atomic formulas and inductively, from formulas A, B, we get new formulas $\neg A$, $A \wedge B$ and $A \vee B$ and from a formula A and a variable symbol x we get the new formulas $\exists x.A$ and $\forall x.A$).

A variable in a formula is free if there is no occurrence of a quantifier binding the variable name on the path of the syntax tree of the formula reaching the occurrence of the variable. A formula without free variables is a **sentence**. A **first-order theory** is a set of sentences written in some first-order language.

A **first-order structure** $\mathcal{A}$ over a first-order language L is a tuple with four components: a set A called the domain of A; a set of elements of A corresponding to the constant symbols of L; for each positive integer n, a set of n-ary relations on A (i.e. subsets of A^n), each of which is named by one or more n-ary relation symbols of L and for each positive integer n, a set of n-ary operations on A (i.e maps from A^n to A), each of which is named by one or more n-ary function symbols of L. The mapping assigning each first-order symbol of L to its corresponding interpretation in $\mathcal{A}$ is denoted $\cdot^{\mathcal{A}}$. This function is extended to work on terms, i.e. the application of function symbols to constants, variables or other terms, by requiring that $(f(s_1, \ldots, s_n))^{\mathcal{A}} = f^{\mathcal{A}}(s_1^{\mathcal{A}}, \ldots, s_n^{\mathcal{A}})$.

Let ϕ be a sentence in a first-order language L and let $\cdot^{\mathcal{A}}$ be an interpretation of the symbols of L in the structure $\mathcal{A}$. The **sentence ϕ is satisfied in the structure $\mathcal{A}$**, written $\mathcal{A} \models \phi$, if the following conditions apply.

- If ϕ is the atomic sentence $R(s_1, \ldots, s_n)$ where $s_1, \ldots, s_n$ are terms of L then $\mathcal{A} \models \phi$ if and only if $(s_1^{\mathcal{A}}, \ldots, s_n^{\mathcal{A}}) \in R^{\mathcal{A}}$.
- $\mathcal{A} \models \neg\phi$ if and only if it is not true that $\mathcal{A} \models \phi$.
- $\mathcal{A} \models \phi_1 \wedge \phi_2$ if and only if $\mathcal{A} \models \phi_1$ and $\mathcal{A} \models \phi_2$.
- $\mathcal{A} \models \phi_1 \vee \phi_2$ if and only if $\mathcal{A} \models \phi_1$ or $\mathcal{A} \models \phi_2$.
- If ϕ is the sentence $\forall y.\psi(y)$ then $\mathcal{A} \models \phi$ if and only if for all elements b of A, $\mathcal{A} \models \psi(b)$.
- If ϕ is the sentence $\exists y.\psi(y)$ then $\mathcal{A} \models \phi$ if and only if there is at least one element b of A such that $\mathcal{A} \models \psi(b)$.

Let Ax be a set of first-order sentences. We define the relation $Ax \models \phi$ which holds if and only if for every structure $\mathcal{A}$, if $\mathcal{A} \models ax$ for each sentence $ax \in Ax$ then $\mathcal{A} \models \phi$.

The **axiomatic theory** defined by a set of axioms Ax is $Th(Ax) = \{\phi | Ax \models \phi\}$.

The **semantic theory** of a structure $\mathcal{A}$ is the set $Th(\mathcal{A}) := \{\phi \,|\, \mathcal{A} \models \phi\}$.

When studying the sets $Th(\mathcal{A})$ and $Th(Ax)$ we may assume the sentences are in prenex normal form. A **prenex normal form** of a first-order formula F is a first-order formula consisting of a string of quantifiers (called the prefix of the formula) followed by a quantifier-free formula (known as the matrix of the formula) which is equivalent to F. It is well-known that there is a polynomial time algorithm transforming sentences of a first-order theory into to equivalent sentences in prenex normal form.

The **existential fragment of the first-order theory** T, denoted $Th_{\exists^*}(T)$,is the subset of sentences in T whose prefix in prenex normal form is purely existential. We write $Th_{\exists^*}(Ax)$ if T is axiomatically specified and $Th_{\exists^*}(\mathcal{A})$ if T is semantically specified.

3 Array theories

The theory of arrays T_A is defined as a first-order theory with three sorts: A for arrays, I for indices and E for elements of arrays. It has one "read" function symbol $\cdot[\cdot] : A \times I \to E$, one "write" function symbol $\cdot\langle\cdot\triangleleft\cdot\rangle : A \times I \times E \to A$ and includes the equality relation symbol $\cdot = \cdot$ for indices and elements. The theory is described axiomatically as the sets of sentences satisfying axioms Ax of the following form [2]. $=$ is axiomatised as a reflexive, symmetric and transitive relation. Array read is assumed to be a congruence relation, i.e. $\forall a, i, j.i = j \to a[i] = a[j]$. Finally, there are axioms relating the read and write operations $\forall a, v, i, j.i = j \to a\langle i \triangleleft v\rangle[j] = v$ and $\forall a, v, i, j.i \neq j \to a\langle i \triangleleft v\rangle[j] = a[j]$.

The quantifier-free fragment of T_A is the set of formulas that can be written without any use of quantifiers. Our goal is to decide which quantifier-free formulas are satisfiable. The satisfiable quantifier-free formulas correspond precisely to set of formulas in $Th_{\exists^*}(Ax)$.

Proposition 1. *The existential closure of the satisfiable formulas in the quantifier-free fragment of T_A is the set $Th_{\exists^*}(Ax)$. Conversely, if we drop the existential prefixes in $Th_{\exists^*}(Ax)$, we obtain the satisfiable formulas of the quantifier-free fragment of T_A.*

Proof. The existential closure of a satisfiable formula in the quantifier-free fragment of T_A is, by definition, in $Th_{\exists^*}(Ax)$. A formula in $Th_{\exists^*}(Ax)$ is true by definition. Converting it to prenex normal form and dropping the existential quantifier prefix leaves a formula of the quantifier-free fragment of T_A. □

Many works, starting with [29], consider an extension of the theory T_A with axioms of the form $R(a_1, \ldots, a_n) \leftrightarrow \forall i.R(a_1(i), \ldots, a_n(i))$ which says that some relation holds on a tuple of array variables $a_1, \ldots, a_n$ if and only it holds at each component. One example is the extensionality axiom $\forall a, b.a = b \leftrightarrow (\forall i.a[i] = b[i])$. In [26], we observed that several fragments extending combinatory array logic [7] can be described semantically as the theory of a power structure [20]. More precisely, we showed the following results.

Definition 2. *The generalised power $\mathcal{P}(\mathcal{M}, I)$ of the combinatory array logic fragment is a structure whose carrier set is the set M^I of functions from the index set I to the carrier set of the structure of the array elements M and whose relations are interpreted as sets of the form*

$$\{(a_1, \ldots, a_n) \in (M^I)^n | \Phi(S_1, \ldots, S_k)\}$$

where Φ is a Boolean algebra expression over $\mathcal{P}(I)$ using the symbols $\subseteq$, $\cup$, $\cap$ or $\cdot^c$ and each set variable S is interpreted as $S = \{i \in I | \theta(a_1(i), \ldots, a_n(i))\}$ where θ is a formula in the theory of the elements.

Theorem 3. *The quantifier-free formulas of combinatory array logic can be encoded in polynomial time as sentences in the theory of the generalised power $\mathcal{P}(\mathcal{A}, I)$ in a way that preserves satisfiability of the formulas.*

Theorem 4. *The theory $Th_{\exists^*}(\mathcal{P}(\mathcal{A}, I))$ can be decided in* NP *even when the algebra of indices $\mathcal{P}(I)$ includes a cardinality operator and the language includes linear arithmetic constraints on the cardinality constraints.*

Our goal in this note is to generalise this result to summation constraints over the array (function symbols) variables. Interestingly, to preserve decidability one has to disallow constants in the element theory specifications θ.

4 Decision Procedure

Our first step is defining the input language to be decided.

Definition 5. *The theory of generalised powers with sums consists of formulae of the form*

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

where F is a formula from Boolean algebra of sets, $\varphi_0, \ldots, \varphi_k$ are formulae in the existential fragment of Presburger arithmetic and $\overline{c}$ is a tuple of arrays of natural numbers. We will refer to the first conjunct of this formula as the Boolean algebra term, to the second conjunct as the set interpretations and to the third conjunct as the multiset interpretations.

There are some differences between Definition 5 and [23, Definition 2.1]. [23, Definition 2.1] has a quantifier-free Presburger arithmetic formula instead of the Boolean algebra term F. Second, the term $\forall e.F$ corresponds to our set interpretations. Third, the term $(u_1, \ldots, u_n) = \sum_{e \in E}(t_1, \ldots, t_n)$ corresponds to our multiset interpretation. It should be noted that the indices in our setting range over the natural numbers and not over a finite set E as in [23].

An important observation is that the definition does not allow free variables to be shared between the three conjuncts. In fact, if we allowed such shared constants, the resulting fragment would have an undecidable satisfiability problem.

Corollary 6. *The satisfiability of formulas of the form*

$$F(S_1, \ldots, S_k, \overline{\sigma}, \overline{f}) \wedge \bigwedge_{i=1}^{k} S_i = \{n \in I | \varphi_i(\overline{c}(n), \overline{f})\} \wedge \overline{\sigma} = \sum (\!| \overline{c}(n) \mid \varphi_0(\overline{c}(n), \overline{f}) |\!) \tag{2}$$

is undecidable.

Proof. By reduction from Hilbert's tenth problem [17]. One can encode in this theory the addition of two natural numbers using the formula F which is in Boolean algebra of sets with cardinalities and thus includes quantifier-free Presburger arithmetic. Multiplication $z = xy$ can be encoded by imposing the array $\overline{c}$ to be equal to the constant x in each position, have length y and sum up to z. □

Let us now describe the main steps of the decision procedure for the theory in Definition 5.

Elimination of terms in Boolean algebra with Cardinalities. To eliminate these constraints, we introduce k array variables $c_1, \ldots, c_k$ and we rewrite the Boolean algebra expressions and cardinality constraints in terms of set interpretations and summation constraints. See the appendix for further details.

As a result of this phase, we obtain a formula of the form:

$$\psi(\overline{\sigma}) \wedge \bigwedge_{i=1}^{k} I = \{n \in I | \varphi_i(\overline{c}(n))\} \wedge \overline{\sigma} = \sum (\!| \overline{c}(n) \mid \varphi_0(\overline{c}(n)) |\!) \tag{3}$$

where ψ is a quantifier-free Presburger arithmetic formula and all the Boolean algebra and cardinality constraints has been translated into set interpretations and summation constraints.

Elimination of the Set Interpretations. The next step in the decision procedure is to eliminate the set interpretation term. However, in the form of Formula 3, this is particularly simple. Formula 3 is equivalent to:

$$\psi(\overline{\sigma}) \wedge \overline{\sigma} = \sum (\!|\overline{c}(n) \mid \bigwedge_{i=0}^{k} \varphi_i(\overline{c}(n))|\!) \tag{4}$$

It thus remains to remove the summation operator.

Elimination of the Summation Operator. The next step is to rewrite sums to a star operator introduced in [24]. Given a set A, the set A^* is defined as:

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

Proposition 7 (Multiset elimination)**.** *The formula*

$$\exists \overline{\sigma}, \overline{c}. \psi(\overline{\sigma}) \wedge \overline{\sigma} = \sum_{n \in \mathbb{N}} (\!|\overline{c}(n) \mid \varphi(\overline{c}(n))|\!) \tag{5}$$

and the formula

$$\exists \overline{\sigma}. \psi(\overline{\sigma}) \wedge \overline{\sigma} \in \{\overline{k} \mid \varphi(\overline{k})\}^* \tag{6}$$

are equivalent.

The argument needs to be adapted from Theorem 2.4 of [25] since both our index and element set are infinite. The details are given in the appendix.

The next step is to eliminate the star operator introduced in Proposition 7. To do so, one could use [25, Theorem 2.23] which shows that if Formula 6 is satisfiable then it also has a solution that can be written with a polynomial number of bits. We adapt this result to the case where we consider explicit integer exponents in the sets. That is we consider given a set A and an integer $m \in \mathbb{N}$, the set A^m defined as $A^m = \{u \mid x_1, \ldots, x_m \in A. u = \sum_{i=1}^{m} x_i\}$. The reason to do this is that when mixing summation and other kinds constraints such as in [27], we need to *synchronise* the cardinality constraints of the combined theory with the cardinality constraints arising from the number of addends used in the sums.

Definition 8. *LIA with sum cardinalities, denoted LIA^{card}, is the theory consisting of formulas of the form $F_0 \wedge \bigwedge_{i=1}^{n} u \in \{x \mid F_i(x)\}^{x_i}$ where F_0 and F are quantifier-free Presburger arithmetic formulae.*

Proposition 9. *LIA^{card} is in* NP.

A detailed proof is given in the appendix.

5 Conclusion

Despite the numerous works that are dedicated to the theory of arrays and its variations, it remains a challenge to provide a comprehensive classification of array theories according to the computational complexity of their satisfiability problem and their expressive power. This paper shows that even classical theories such as the combinatory array logic fragment can be optimised with respect to both metrics. An interesting extension that we leave open is to support combinatory array logic with sums and different element sorts.

Acknowledgements.

The author wishes to express his gratitude to Viktor Kunčak for useful suggestions on the presentation of the results of this paper.

References

[1] F. Alberti, S. Ghilardi, and E. Pagani. Cardinality constraints for arrays (decidability results and applications). *Formal Methods in System Design*, 51(3):545–574, December 2017. doi:10.1007/s10703-017-0279-6.

[2] Aaron Bradley and Zohar Manna. *Calculus of computation: decision procedures with applications to verification*. Springer, Berlin, 2007. OCLC: ocn190764844.

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

[4] Yu-Fang Chen, Philipp Rümmer, and Wei-Lun Tsai. A Theory of Cartesian Arrays (with Applications in Quantum Circuit Verification). In Brigitte Pientka and Cesare Tinelli, editors, *Automated Deduction – CADE 29*, pages 170–189, Cham, 2023. Springer Nature Switzerland. doi:10.1007/978-3-031-38499-8_10.

[5] Sylvain Conchon, David Declerck, and Fatiha Zaïdi. Parameterized Model Checking on the TSO Weak Memory Model. *Journal of Automated Reasoning*, 64(7):1307, 2020. URL: https://inria.hal.science/hal-03149332, doi:10.1007/s10817-020-09565-w.

[6] Przemysław Daca, Thomas A. Henzinger, and Andrey Kupriyanov. Array Folds Logic. In *Computer Aided Verification*, Lecture Notes in Computer Science, pages 230–248, Cham, 2016. Springer International Publishing. doi:10.1007/978-3-319-41540-6_13.

[7] Leonardo de Moura and Nikolaj Bjorner. Generalized, efficient array decision procedures. In *2009 Formal Methods in Computer-Aided Design*, pages 45–52, Austin, TX, November 2009. IEEE. doi:10.1109/FMCAD.2009.5351142.

[8] Leonardo de Moura and Nikolaj 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*, pages 337–340, Berlin, Heidelberg, 2008. Springer Berlin Heidelberg. doi:10.1007/978-3-540-78800-3_24.

[9] Haoran Ding, Zhaoguo Wang, Yicun Yang, Dexin Zhang, Zhenglin Xu, Haibo Chen, Ruzica Piskac, and Jinyang Li. Proving Query Equivalence Using Linear Integer Arithmetic. *Proceedings of the ACM on Management of Data*, 1(4):1–26, December 2023. URL: https://dl.acm.org/doi/10.1145/3626768, doi:10.1145/3626768.

[10] S. Feferman and R. Vaught. The first order properties of products of algebraic systems. *Fundamenta Mathematicae*, 47(1):57–103, 1959.

[11] Silvio Ghilardi and Silvio Ranise. MCMT: A Model Checker Modulo Theories. In Jürgen Giesl and Reiner Hähnle, editors, *Automated Reasoning*, pages 22–29, Berlin, Heidelberg, 2010. Springer. doi:10.1007/978-3-642-14203-1_3.

[12] Alessandro Gianola. *Verification of Data-Aware Processes via Satisfiability Modulo Theories*, volume 470 of *Lecture Notes in Business Information Processing*. Springer Nature Switzerland, Cham, 2023. URL: https://link.springer.com/10.1007/978-3-031-42746-6, doi:10.1007/978-3-031-42746-6.

[13] Klaus v. Gleissenthall, Nikolaj Bjørner, and Andrey 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, pages 599–613, New York, NY, USA, June 2016. Association for Computing Machinery. doi:10.1145/2908080.2908129.

[14] Wilfrid Hodges. *Model Theory*. Encyclopedia of Mathematics and its Applications. Cambridge University Press, Cambridge, 1993.

[15] Daniel Kroening and Ofer Strichman. *Decision Procedures*. Texts in Theoretical Computer Science. An EATCS Series. Springer Berlin Heidelberg, Berlin, Heidelberg, 2 edition, 2016. doi:10.1007/978-3-662-50497-0.

[16] Denis Lugiez and Silvano Dal Zilio. Multitrees Automata, Presburger's Constraints and Tree Logics. Technical Report 08-2002, Laboratoire d'Informatique Fondamentale de Marseille, 2002.

[17] Yuri Matiyasevich. *Hilbert's 10th Problem*. Foundations of Computing. MIT Press, 1993. URL: https://mitpress.mit.edu/9780262132954/hilberts-10th-problem/.

[18] José Meseguer. Lecture Notes on Topics in Automated Reasoning, 2017. URL: https://courses.grainger.illinois.edu/cs576/sp2017/.

[19] M.W. Moskewicz, C.F. Madigan, Y. Zhao, L. Zhang, and S. Malik. Chaff: engineering an efficient SAT solver. In *Proceedings of the 38th Design Automation Conference (IEEE Cat. No.01CH37232)*, pages 530–535, June 2001. ISSN: 0738-100X. URL: https://ieeexplore.ieee.org/document/935565, doi:10.1145/378239.379017.

[20] Andrzej Mostowski. On direct products of theories. *The Journal of Symbolic Logic*, 17(1):1–31, March 1952. Publisher: Cambridge University Press. URL: https://www.cambridge.org/core/journals/journal-of-symbolic-logic/article/on-direct-products-of-theories/CDB043DECF7B4195A7F2DB79493409CE, doi:10.2307/2267454.

[21] Robert Nieuwenhuis, Albert Oliveras, and Cesare Tinelli. Solving SAT and SAT Modulo Theories: From an abstract Davis–Putnam–Logemann–Loveland procedure to DPLL(T). *Journal of the ACM*, 53(6):937–977, November 2006. URL: https://dl.acm.org/doi/10.1145/1217856.1217859, doi:10.1145/1217856.1217859.

[22] Christos H. Papadimitriou. On the complexity of integer programming. *Journal of the ACM*, 28(4):765–768, October 1981. doi:10.1145/322276.322287.

[23] Ruzica Piskac. *Decision Procedures for Program Synthesis and Verification*. PhD thesis, EPFL, Lausanne, 2011.

[24] Ruzica Piskac and Viktor Kunčak. Linear Arithmetic with Stars. In *Computer Aided Verification*, Lecture Notes in Computer Science, pages 268–280, Berlin, Heidelberg, 2008. Springer. doi:10.1007/978-3-540-70545-1_25.

[25] Ruzica Piskac and Thomas Wies. Decision Procedures for Automating Termination Proofs. In *Verification, Model Checking, and Abstract Interpretation*, volume 6538, pages 371–386, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg. Series Title: Lecture Notes in Computer Science. URL: http://link.springer.com/10.1007/978-3-642-18275-4_26, doi:10.1007/978-3-642-18275-4_26.

[26] Rodrigo Raya and Viktor Kunčak. NP Satisfiability for Arrays as Powers. In *Verification, Model Checking, and Abstract Interpretation*, Lecture Notes in Computer Science, pages 301–318, Cham, 2022. Springer International Publishing. doi:10.1007/978-3-030-94583-1_15.

[27] Rodrigo Raya and Viktor Kunčak. On Algebraic Array Theories. *Journal of Logical and Algebraic Methods in Programming*, 136:100906, January 2024. doi:10.1016/j.jlamp.2023.100906.

[28] Georg Stefan Schmid and Viktor Kunčak. Generalized Arrays for Stainless Frames. In *Verification, Model Checking, and Abstract Interpretation*, Lecture Notes in Computer Science, pages 332–354, Cham, 2022. Springer International Publishing. doi:10.1007/978-3-030-94583-1_17.

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

[30] Qinshi Wang and Andrew W. Appel. A Solver for Arrays with Concatenation. *Journal of Automated Reasoning*, 67(1), January 2023. doi:10.1007/s10817-022-09654-y.

Appendix

A Elimination of Boolean algebra with Cardinalities terms

- For every newly introduced array variable c_j, we introduce the set interpretation:

$$I = \{n \in I | c_j(n) = 0 \vee c_j(n) = 1\}$$

this constraint is added to the set interpretation term.

- For every newly introduced array variable c_j, we rewrite the set variable S_j into the following set interpretation:

$$S_j := \{i \in I | c_j(n) = 1\}$$

which says that the indices in S_j corresponds to the positions where c_j is equal to one.

- We then substitute each Boolean algebra expression appearing in the Boolean algebra with cardinalities term of Formula 1, by repeatedly applying the following rewrite rules:

$$\begin{aligned} S_j^c &:= \{n \in I | c_j(n) = 0\} \\ S_j \cup S_k &:= \{n \in I | c_j(n) = 1 \vee c_k(n) = 1\} \\ S_j \cap S_k &:= \{n \in I | c_j(n) = 1 \wedge c_k(n) = 1\} \end{aligned}$$

- By the above rewriting process, each cardinality constraint $|S| = k$ is rewritten as

$$|\{n \in I | \varphi(\overline{c}(n))\}| = k$$

where the variable $\overline{c}$ lists the newly introduced array variables $c_1, \ldots, c_k$. We then introduce a new array variable $x = ite(\varphi(\overline{c}), 1, 0)$, that is, $x(n)$ is equal to one if $\varphi(\overline{c}(n))$ holds and it is equal to 0 otherwise. This can be encoded with the set interpretation

$$I = \{n \in I | x(n) = ite(\varphi(\overline{c}(n)), 1, 0)\}$$

One then rewrites the expression $|S| = k$ into $k = \sum_{i \in I} x(n)$.[1]

B Elimination of the multiset comprehension

Proof. $\Rightarrow$) If (1) is satisfied, there are $\overline{\sigma}, \overline{c}$ such that $\psi(\overline{\sigma}) \wedge \overline{\sigma} = \sum_{n \in \mathbb{N}} (\!|\overline{c}(n) \mid \varphi(\overline{c}(n))|\!)$. We claim that the same $\overline{\sigma}$ satisfies $\psi(\overline{\sigma}) \wedge \bigwedge_{i=1}^{k} \overline{\sigma} \in \{\overline{k} \mid \varphi(\overline{k})\}^*$. By hypothesis, $\psi(\overline{\sigma})$ is true. Moreover, $\overline{\sigma} = (\!|\overline{c}(n) \mid \varphi(\overline{c}(n))|\!)$ and either

- $(\!|\overline{c}(n) \mid \varphi(\overline{c}(n))|\!)$ is finite in which case $\overline{\sigma} \in \{\overline{k} \mid \varphi(\overline{k})\}^*$.
- or $(\!|\overline{c}(n) \mid \varphi(\overline{c}(n))|\!)$ is infinite, in which case $\overline{c}(n)$ is equal to $\overline{0}$ in all but a finite set of indices I since by hypothesis the sum $\overline{\sigma}$ is finite. Then $\overline{\sigma} = \sum_{n \in I} \overline{c}(n)$ and $\overline{\sigma} \in \{\overline{k} \mid \varphi(\overline{k})\}^*$.

[1] It is remarkable than being closed under the if-then-else operator (ite) also appears in modern presentations of Feferman-Vaught theorem, see for instance [14, Theorem 9.6.2], as the property of "being closed under gluing over a Boolean algebra". Such theoretical generalisations are also used in practice, see for instance [28].

$\Leftarrow$) If (2) is satisfied, then there is $\overline{\sigma}$ such that $\psi(\overline{\sigma}) \wedge \overline{\sigma} \in \{\overline{k} \mid \varphi(\overline{k})\}^*$. It follows that there is a finite list of elements $\overline{k}_i$ such that $\overline{\sigma} = \sum_{i=1}^{p} \overline{k}_i$. We define

$$\overline{c}(n) = \begin{cases} \overline{k}_n & \text{if } 1 \leq n \leq p \\ 0 & \text{otherwise} \end{cases}$$

It is immediate that $\overline{\sigma}$ and $\overline{c}$ satisfy $\psi(\overline{\sigma}) \wedge \overline{\sigma} = \sum_{n \in \mathbb{N}} (\!|\overline{c}(n) \mid \varphi(\overline{c}(n))|\!)$. □

C NP membership of LIA^{card}

Proof. Let V_{PA} be a polynomial time verifier for LIA^{card}. Figure 1 gives a verifier V for LIA^{card}. We show that $x \in \mathsf{LIA}^{card}$ if and only if there exists a polynomial-size certificate w such that V accepts $\langle x, w \rangle$.

On input $\langle x, w \rangle$:

1. Interpret w as:
   (a) lists of vectors a_l, b_s generating some element in each semilinear set given by F_i.
   (b) matrices A_i and vectors b_i such that $A_i x \leq b_i$ is in the disjunctive normal form of each F_i.
   (c) a certificate C for V_{PA} on input y.

   where

$$y = \left(\begin{array}{c|c} A_{g_1} & 0 \\ \cdots & 0 \\ A_{g_n} & 0 \\ \hline 0 & A_0 \end{array}\right) \begin{pmatrix} \cdots \\ \lambda_{ij} \\ \cdots \\ \cdots \\ \mu_i \\ \cdots \\ x \\ u \\ v \end{pmatrix} = \begin{pmatrix} b_{g_1} \\ \cdots \\ b_{g_n} \\ b_0 \end{pmatrix} \wedge \bigwedge_{il} A_i a_l \leq b_i \wedge \bigwedge_{is} A_i b_s \leq 0$$

2. Accept iff V_{PA} accepts $\langle y, C \rangle$.

Figure 1: Verifier for LIA^{card}.

$\Rightarrow$) If $x = F_0 \wedge \bigwedge_{i=1}^{n} u \in \{x \mid F_i(x)\}^{x_i} \in \mathsf{LIA}^{card}$ then we show that there is a solution that uses a polynomial number of bits in the size of F_0 and F_1.

We convert each formula F_i to semilinear normal form [25, Theorem 2.13].

Lemma 10. *Let F be a linear arithmetic formula of size s. Then there exist numbers $m, q_1, \ldots, q_m \in \mathbb{N}$ and vectors $a_i, b_{ij} \in \mathbb{N}^n$ for $1 \leq j \leq q_i, 1 \leq i \leq m$ with $\|a_i\|_1, \|b_{ij}\|_1 \leq 2^{p(s)}$*

with p polynomial such that $F(x)$ is equivalent to the formula:

$$\exists \alpha_{11}, \ldots, \alpha_{mq_m}. \bigvee_{i=1}^{m} \left(x = a_i + \sum_{j=1}^{q_i} \alpha_{ij} b_{ij} \right) \tag{7}$$

Next, we eliminate the star operator, as in [16, Proposition 2] and [25, Theorem 2.14].

Lemma 11. *Let F be a quantifier-free linear integer arithmetic formula whose semilinear normal form is formula 7. Then $u \in \{y \mid F(y)\}^x$ is equivalent to*

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

We express the resulting vector u with polynomially many generators [25, Theorem 2.20].

Lemma 12 (Polynomially many generators for sums)**.** *Let F be a quantifier-free linear integer arithmetic formula of size s whose semilinear normal form is formula 7. Then $u \in \{y \mid F(y)\}^x$ is equisatisfiable with*

$$\exists \lambda_{ij}, \mu_i. u = \sum_{i \in I_0} \left(a_i + \sum_{(i,j) \in J} \lambda_{ij} b_{ij} \right) + \sum_{i \in I_1} \mu_i a_i \wedge x = |I_0| + \sum_{i \in I_1} \mu_i$$

for some $I_0, I_1 \subseteq \{1, \ldots, q\}$, $J \subseteq \cup_{i=1}^{q} \{(i,1), \ldots, (i,q_i)\}$, $|I_0| \leq |J| \leq q(s)$, $|I_1| \leq q(s)$ and q is a polynomial.

We write the equation

$$u = \sum_{i \in I_0} \left(a_i + \sum_{(i,j) \in J} \lambda_{ij} b_{ij} \right) + \sum_{i \in I_1} \mu_i a_i \wedge x = |I_0| + \sum_{i \in I_1} \mu_i$$

as a system $A_g x_g = b_g$ of the form

$$\begin{pmatrix} \ldots & a_i & \ldots & \ldots & b_{ij} & \ldots & \ldots & a_i & \ldots & 0 & -I_k \\ 0 & \ldots & \ldots & \ldots & \ldots & 0 & 1 & \ldots & 1 & -1 & 0 \end{pmatrix} \begin{pmatrix} \ldots \\ \lambda_{ij} \\ \ldots \\ \ldots \\ \mu_i \\ \ldots \\ x \\ u \end{pmatrix} = \begin{pmatrix} \ldots \\ -a_i \\ \ldots \\ 0 \\ \ldots \\ 0 \\ -|I_0| \end{pmatrix}$$

Since u is also a solution of F_0 it will further satisfy some system $A_0 w = b_0$ corresponding to one of the terms of the disjunctive normal form of F_0 and where we can assume that the unknown u appears in the first rows of $w = (u, v)$. As a result, we obtain a combined system.

$$\left(\begin{array}{c|c} A_{g_1} & 0 \\ \ldots & \ldots \\ A_{g_n} & 0 \\ \hline 0 & A_0 \end{array} \right) \begin{pmatrix} \ldots \\ \lambda_{ij} \\ \ldots \\ \ldots \\ \mu_i \\ \ldots \\ x \\ u \\ v \end{pmatrix} = \begin{pmatrix} b_{g_1} \\ \ldots \\ b_{g_n} \\ b_0 \end{pmatrix}$$

This system has a polynomial number of rows, columns and uses polynomially many bits for its maximum absolute value. Thus, it is guaranteed to have a solution that uses polynomially many bits by the following theorem from [22, page 767].

Lemma 13. *Let A be an $m \times n$ integer matrix and b a m-vector, both with entries from $[-a..a]$. Then the system $Ax = b$ has a solution in $\mathbb{N}^n$ if and only if it has a solution in $[0..M]^n$ where $M = n(ma)^{2m+1}$.*

Moreover, since all the generators chosen for F_i lie in the same linear subset they satisfy $A_i a_l \leq b_i$ and $A_i b_s \leq 0$ for some system $A_i x \leq b_i$ in the disjunctive normal form of each F_i.

The resulting formula has polynomial-size in the size of F_0 and F_1, thus there exists a polynomial-size certificate C such that V_{PA} accepts $\langle y, C\rangle$. It follows that $V_{\mathsf{LIA}^{card}}$ accepts $\langle x, \langle \{a_l\}, \{b_s\}, \{A_i\}, \{b_i\}, C\rangle\rangle$.

$\Leftarrow$) If $V_{\mathsf{LIA}^{card}}$ accepts $\langle x, w\rangle$ then there exists a polynomial-size certificate C such that V_{PA} accepts $\langle y, C\rangle$ and thus y is satisfiable. This means that for some vectors a_l satisfying F_i and some vectors b_s satisfying the homogeneous part of F_i it holds that

$$u = \sum_{l \in I_0} \left(a_l + \sum_{(l,s) \in J} \lambda_{ls} b_{ls} \right) + \sum_{l \in I_1} \mu_l a_l \wedge x_i = |I_0| + \sum_{l \in I_1} \mu_l$$

and furthermore (u, v) satisfies F_0. It follows that $F_0 \wedge \bigwedge_{i=1}^{n} u \in \{x \mid F_i(x)\}^{x_i}$ is satisfied by such u and v. Thus, $x \in \mathsf{LIA}^{card}$. □