NP Satisfiability for Arrays as Powers*

Rodrigo Raya[0000−0002−0866−9257] and Viktor Kunčak[0000−0001−7044−9522]

School of Computer and Communication Science
École Polytechnique Fédérale de Lausanne (EPFL), Switzerland
{rodrigo.raya,viktor.kuncak}@epfl.ch

Abstract. We show that the satisfiability problem for the quantifier-free theory of product structures with the equicardinality relation is in NP. As an application, we extend the combinatory array logic fragment to handle cardinality constraints. The resulting fragment is independent of the base element and index set theories.

1 Introduction

Arrays are a fundamental data structure in computer science. Decision procedures for arrays are therefore of paramount importance for deductive program verification. A number of results have examined fragments that strike interesting trade-offs between expressive power and complexity [4, 5, 10, 12, 17, 21].

A particularly important fragment for formal verification is combinatory array logic (CAL) fragment [19], which is implemented in the widely used Z3 theorem prover [20]. A key to expressive power of the generalized array fragment is that it extends the extensional quantifier-free theory of arrays [21] (which supports only equality, lookup, and update operations) with point-wise functions and relations, analogous to "vector operations".

In this paper, we start by observing that the generalized array fragment signature corresponds to the signature of a product structure [13]. The decidability of product structures has been studied in the literature on model theory [7, 18]. Moreover, the results from model theory also permit formulas that constrain sets of indices using, for example, equicardinality relation [7], which provides additional expressive power. Unfortunately, the existing presentations of results from model theory typically consider *quantified* first-order theory, resulting in high complexity [8] even when instantiated to the case of no quantifier alternations. The basic source of this inefficiency is that the underlying procedure explicitly constructs exponentially many formulas.

In contrast to these claims about quantified formulas, the results on generalized arrays theories suggest (Theorem 17 in [19]) that the satisfiability problem of the quantifier-free theory of a power structure is in NP whenever the theory of the components is.

In this paper, we present a direct proof of the NP membership for satisfiability of formulas in power structures. The proof is largely independent of the theories

* Research supported in part by the Swiss NSF Project #200021_197288

of the indices and the theory of array elements. As a consequence, we obtain that the satisfiability problem of the quantifier-free fragment of Skolem arithmetic is in NP [11], which, interestingly, was previously shown by appealing to results in number theory.

As a **main contribution**, we generalize this construction to prove that the satisfiability problem of the quantifier-free fragment of BAPA [14] is in NP when set variables are interpreted with index sets defined by formulas of the language of the component theory. Whereas the quantifier-free fragment of BAPA (termed QFBAPA) was shown to be in NP [16], it was not clear that such construction carries over to the situation where index sets are *interpreted* to be positions in the arrays. In this paper we show that interpreting QFBAPA sets as sets of array indices that satisfy certain formula results in a logic whose satisfiability is still in NP. We call this new quantifier-free theory QFBAPAI. We show how to use it to encode constraints that mimic those of combinatory array logic (CAL) [19]. The result is an extension of CAL that can additionally express cardinality constraints that hold componentwise. Unlike [5], the logic is independent of the component or the index theory. Our formalism shows that QFBAPA sets can be interpreted, overcoming a limitation pointed out in [1]. The use of cardinality constraints makes our results out of scope of [19], whereas avoiding explicit construction of all Venn regions allows us to, unlike, [7], establish membership in NP.

2 NP Complexity for Power Structures

Throughout the paper, we fix a first-order language L, a non-empty set I and a structure $\mathcal{M}$ with carrier M for the components of the arrays. We model arrays as a particular kind of product structure:

Definition 1. *The power structure Π has the function space M^I as domain and interprets the symbols of the language L as follows:*

- *For each constant c and $i \in I$, $c^\Pi(i) = c^\mathcal{M}$.*
- *For each function symbol f, $i \in I$, $n \in \mathbb{N}$ and $(a_1, \ldots, a_n) \in (M^I)^n$:*

$$f^\Pi(a_1, \ldots, a_n)(i) = f^\mathcal{M}(a_1(i), \ldots, a_n(i))$$

- *For each relation symbol R, $n \in \mathbb{N}$ and $(a_1, \ldots, a_n) \in (M^I)^n$:*

$$(a_1, \ldots, a_n) \in R^\Pi \text{ if and only if for every } i \in I, (a_1(i), \ldots, a_n(i)) \in R^\mathcal{M}$$

We will write tuples $(a_1, \ldots, a_n) \in (M^I)^n$ as $\overline{a}$ and $(a_1(i), \ldots, a_n(i))$ as $\overline{a}(i)$.

Definition 2. *The quantifier-free theory of a model $\mathcal{N}$, $Th_{\exists^*}(\mathcal{N})$, is the set of existentially quantified formulas φ of L such that $\mathcal{N} \models \varphi$. A solution of the formula is a satisfying assignment to the existential variables.*

Lemma 1. *Let ψ be a first-order formula in prenex form and C a disjunct of the DNF form of its matrix. Then $|C| = O(|\psi|)$.*

Proof. The DNF conversion only affects the propositional structure of the formula. Thus, in C one may at most have the relations occurring in ψ and their negations. In the worst case, one gets at most $2|\psi|$ symbols accounting for the relations and at most $4|\psi|$ symbols accounting for the conjunctions and negations. Therefore, $|C| \leq 6 \cdot |\psi|$.

The following result shows the spirit of our complexity analysis: we take a classical construction (power structure) but analyze its complexity for quantifier-free fragment that is relevant for program verification.

Theorem 1. *$Th_{\exists^*}(\mathcal{M}) \in NP$ if and only if $Th_{\exists^*}(\Pi) \in NP$.*

Proof. 1) Assume that V_C is a polynomial time verifier for $Th_{\exists^*}(\mathcal{M})$. Figure 1 gives a polynomial time verifier V for $Th_{\exists^*}(\Pi)$. In what follows, we will use x to refer to the formula to be satisfied and w for the certificate or witness that the verifier takes. t^i_j are terms in the logical language L. We use a_j to indicate the arity of the relation symbol R_j. t is a natural number greater or equal than one. We show that the machine is a verifier for $Th_{\exists^*}(\Pi)$:

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

1. Take w and interpret it as:
   - Some disjunct of the DNF form for x:
   $$\varphi \equiv \exists x_1, \ldots, x_n. \bigwedge_{i=1}^{l} R_i(t^i_1, \ldots, t^i_{a_i}) \wedge \bigwedge_{j=l+1}^{k} \neg R_j(t^j_1, \ldots, t^j_{a_j})$$
   - A partition $P = \{p_1, \ldots, p_t\}$ of $\{l+1, \ldots, k\}$.
   - Certificates $C_0, \ldots, C_t$ for V_C on inputs:
   $$\varphi_0 \equiv \exists x_1, \ldots, x_n. \bigwedge_{i=1}^{l} R_i(t^i_1, \ldots, t^i_{a_i})$$
   $$\varphi_d \equiv \exists x_1, \ldots, x_n. \bigwedge_{i=1}^{l} R_i(t^i_1, \ldots, t^i_{a_i}) \wedge \bigwedge_{e \in p_d} \neg R_e(t^e_1, \ldots, t^e_{a_e})$$

   for each $p_d \in P$.
2. If $t \leq |I|$ then reject.
3. Otherwise, run V_C with $\langle \varphi_d, C_d\rangle$ for $d = 0, \ldots, t$.
4. Accept iff all runs accept.

Fig. 1. Verifier for $Th_{\exists^*}(M^I)$

– w has polynomial size in $|x|$:

By lemma 1, $|\varphi| = O(|x|)$.

Thus, $k = O(|x|)$.

$P = O(|x|^2)$ since P can be written with $k\log(k) + k$ bits.

Since $|C_d| = O(|\varphi_d|^{c_d})$ and $|\varphi_d| \leq |\varphi| = O(|x|)$, $|C_d| = O(|x|^{c_d})$.

Thus, $|w| = |\varphi| + |P| + \sum_{d=0,\dots,t} |C_d| = O\Big(|x|^{\max\{2, \max_d c_d\}}\Big)$.

– V runs in polynomial time in $|x|$:

Building the list of φ_d is $O(|x|^2)$.

As above, $|\varphi_d| \leq |\varphi| = O(|x|)$.

So each call to V_C runs in $O(|x|^f)$ (V_C is polynomial time).

Like before, $k = O(|x|)$.

Therefore, V runs in $O\Big(|x|^{\max\{2, f+1\}}\Big)$.

– V is a verifier for $Th_{\exists^*}(\Pi)$:

$\Rightarrow$) If $x \in Th_{\exists^*}(\Pi)$ then writing x in prenex DNF form, there is at least one disjunct φ (as in figure 1) true in the product. Thus, there is $\overline{s} \in M^I$ satisfying:

$$\begin{array}{c} \dots\dots\dots\dots\dots\dots\dots\dots \\ ({t_1^i}^{\Pi}[\overline{x} \mapsto \overline{s}], \dots, {t_{a_i}^i}^{\Pi}[\overline{x} \mapsto \overline{s}]) \in R_i^{\Pi} \\ \dots\dots\dots\dots\dots\dots\dots\dots \\ ({t_1^j}^{\Pi}[\overline{x} \mapsto \overline{s}], \dots, {t_{a_j}^j}^{\Pi}[\overline{x} \mapsto \overline{s}]) \notin R_j^{\Pi} \\ \dots\dots\dots\dots\dots\dots\dots\dots \end{array}$$

Using the semantics of products this means:

$$\begin{array}{c} \dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots \\ \forall r \in I.({t_1^i}^{\mathcal{M}}[\overline{x} \mapsto \overline{s}(r)], \dots, {t_{a_i}^i}^{\mathcal{M}}[\overline{x} \mapsto \overline{s}(r)]) \in R_i^{\mathcal{M}} \\ \dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots \\ \exists r \in I.({t_1^j}^{\mathcal{M}}[\overline{x} \mapsto \overline{s}(r)], \dots, {t_{a_j}^j}^{\mathcal{M}}[\overline{x} \mapsto \overline{s}(r)]) \notin R_j^{\mathcal{M}} \\ \dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots \end{array}$$

So there is a map $r : \{l+1, \dots, k\} \to I$ that assigns to each formula, one index where it holds. r induces a partition $P = r^{-1}(I)$ of $\{l+1, \dots, k\}$ with $t = |P| \leq \min(|I|, k-l)$. Each part $p_d = \{e_1, \dots, e_m\}$ and each associated index $r_d = r(e_i)$, satisfy the following system:

$$\begin{array}{c} \dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots \\ ({t_1^i}^{\mathcal{M}}[\overline{x} \mapsto \overline{s}(r_d)], \dots, {t_{a_i}^i}^{\mathcal{M}}[\overline{x} \mapsto \overline{s}(r_d)]) \in R_i^{\mathcal{M}} \\ \dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots \\ ({t_1^{e_1}}^{\mathcal{M}}[\overline{x} \mapsto \overline{s}(r_d)], \dots, {t_{a_{e_1}}^{e_1}}^{\mathcal{M}}[\overline{x} \mapsto \overline{s}(r_d)]) \notin R_{e_1}^{\mathcal{M}} \\ \dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots \\ ({t_1^{e_m}}^{\mathcal{M}}[\overline{x} \mapsto \overline{s}(r_d)], \dots, {t_{a_{e_m}}^{e_m}}^{\mathcal{M}}[\overline{x} \mapsto \overline{s}(r_d)]) \notin R_{e_m}^{\mathcal{M}} \end{array}$$

Equivalently, for each $d \in \{1, \ldots, t\}$, $\mathcal{M} \models \varphi_d[\overline{x} \mapsto \overline{s}(r_d)]$. For $d = 0$, we set:

$$r_0 = \begin{cases} \text{any index } i \in I & \text{if } t = 0 \\ \text{some } r_d \in \{r_1, \ldots, r_t\} & \text{if } t > 0 \end{cases}$$

Then $\mathcal{M} \models \varphi_0[\overline{x} \mapsto \overline{s}(r_0)]$. By definition of V_C, there are polynomially-sized certificates $C_0, \ldots, C_t$ such that V_C accepts $\langle \varphi_d, C_d \rangle$ for each d. Thus V accepts $\langle x, \langle \varphi, P, C_0, \ldots, C_t \rangle \rangle$.

$\Leftarrow$) Let $w = \langle \varphi, P, \{C_d\}_{d \in \{0,\ldots,t\}} \rangle$ be a certificate such that V accepts $\langle x, w \rangle$. Then, by step 2, $t = |P| \leq |I|$ and for each $d \in \{0, \ldots, t\}$, V_C accepts $\langle \varphi_d, C_d \rangle$, i.e. $\mathcal{M} \models \varphi_d$. So there are solutions $x_{\cdot i} = (x_{1i}, \ldots, x_{ni})^t$ to the formulas:

$$\varphi_0 \equiv \exists x_{10}, \ldots, \exists x_{n0}. \bigwedge_{i=1}^{l} R_i(t_1^i, \ldots, t_{a_i}^i)$$

$$\varphi_d \equiv \exists x_{1d}, \ldots, \exists x_{nd}. \bigwedge_{i=1}^{l} R_i(t_1^i, \ldots, t_{s_i}^i) \wedge \bigwedge_{e \in p_d} \neg R_e(t_1^e, \ldots, t_{a_e}^e)$$

Fix distinct $i_1, \ldots, i_t \in I$. Consider the $n \times |I|$ matrix with entries:

$$s_{ji} = \begin{cases} x_{ji} & \text{if } i \in \{i_1, \ldots, i_t\} \\ x_{j0} & \text{otherwise} \end{cases}$$

The rows of this matrix $\overline{s} = \{s_1, \ldots, s_n\}$ are solutions of φ in the product structure:

$$\begin{array}{c} \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots \\ ({t_1^i}^{\Pi}[\overline{x} \mapsto \overline{s}], \ldots, {t_{a_i}^i}^{\Pi}[\overline{x} \mapsto \overline{s}]) \in R_i^{\Pi} \\ \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots \\ ({t_1^j}^{\Pi}[\overline{x} \mapsto \overline{s}], \ldots, {t_{a_j}^j}^{\Pi}[\overline{x} \mapsto \overline{s}]) \notin R_j^{\Pi} \\ \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots \end{array}$$

Using the definition of product, it is sufficient to show:

$$\begin{array}{c} \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots \\ \forall r \in I. ({t_1^i}^{\mathcal{M}}[\overline{x} \mapsto s(r)], \ldots, {t_{a_i}^i}^{\mathcal{M}}[\overline{x} \mapsto s(r)]) \in R_i^{\mathcal{M}} \\ \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots \\ \exists r \in I. ({t_1^j}^{\mathcal{M}}[\overline{x} \mapsto s(r)], \ldots, {t_{a_j}^j}^{\mathcal{M}}[\overline{x} \mapsto s(r)]) \notin R_j^{\mathcal{M}} \\ \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots \end{array}$$

For $i \in \{1, \ldots, l\}$ and each $r \in I$, the following formula needs to hold:

$$({t_1^i}^{\mathcal{M}}[\overline{x} \mapsto s(r)], \ldots, {t_{s_i}^i}^{\mathcal{M}}[\overline{x} \mapsto s(r)]) \in R_i^{\mathcal{M}}$$

If $r \in \{i_1, \ldots, i_t\}$ then $s(r) = x_{\cdot r}$ (i.e. all $x_{1r}, \ldots, x_{nr}$) and the equation holds since $\mathcal{M} \models \varphi_r[x_{\cdot r}]$. Otherwise, $s(r) = x_{\cdot 0}$ and the equation holds since $\mathcal{M} \models \varphi_0[x_{\cdot 0}]$.

For $j \in \{l+1, \ldots, k\}$ and some $r \in I$, the following formula needs to hold:

$$(t_1^{j\,\mathcal{M}}[\overline{x} \mapsto s(r)], \ldots, t_{s_j}^{j\;\mathcal{M}}[\overline{x} \mapsto s(r)]) \notin R_j^{\mathcal{M}}$$

We take $r = i_d$ such that $j \in p_d$. Then $s(r) = x_{\cdot r}$ and the equation holds since $\mathcal{M} \models \varphi_r[x_{\cdot r}]$.

2) Conversely, assume that V is a verifier for $Th_{\exists^*}(\Pi)$ and let's give a formal argument to show that $Th_{\exists^*}(\mathcal{M})$ is in NP. The idea is that one can extend the signature of L with relations R whose interpretation is that of any quantifier-free formula φ while retaining NP complexity. Indeed, let $\mathcal{N}$ be any structure for the language L and let $\varphi(x_1, \ldots, x_n)$ be any formula of L. Define $R(x_1, \ldots, x_n) := \varphi(x_1, \ldots, x_n)$ and $\mathcal{N}^{ext}$ the model $\mathcal{N}$ extended with the relation symbol R in such a way that $R^{\mathcal{N}^{ext}}(v_1, \ldots, v_n) = \varphi^{\mathcal{N}}(v_1, \ldots, v_n)$ for values v_i of the carrier of $\mathcal{N}$. We show that:

$$Th_{\exists^*}(\mathcal{N}) \in \text{NP} \iff Th_{\exists^*}(\mathcal{N}^{ext}) \in \text{NP}$$

First observe that $|\varphi(x_1, \ldots, x_n)|$ is an affine function in $|x_i|$: there is a constant term accounting for the logical symbols, plus terms $a_i|x_i|$ accounting for the occurrences of the x_i. Now, if $\psi \in Th_{\exists^*}(\mathcal{N})$ then when we contract the occurrences of φ into R and we still get a linear size in $|\psi|$. Therefore, the verifier for $Th_{\exists^*}(\mathcal{N}^{ext})$ gives the result. If on the other hand, $\psi \in Th_{\exists^*}(\mathcal{N}^{ext})$ then expanding the occurrences of $\mathcal{R}$ each $|x_i|$ is bounded in $|\psi|$, so the expanded expression augments its size by a quadratic factor $O(|\psi|^2)$. The verifier for $Th_{\exists^*}(\mathcal{N})$ gives the result. Finally, let's see that:

$$Th_{\exists^*}(\Pi^{ext}) \in \text{NP} \implies Th_{\exists^*}(\mathcal{M}) \in \text{NP}$$

Given $\varphi \in Th_{\exists^*}(\Pi^{ext})$, we define a relation $R := \varphi$ and consider the corresponding extended language $Th_{\exists^*}(\Pi^{ext(\varphi)})$ which by assumption is in NP. Thus, it is decidable in NP that R holds in the product structure. But, $R^{\Pi} \equiv \forall i \in I. \varphi^{\mathcal{M}}$. Given that I is non-empty, we have that the verifier for $Th_{\exists^*}(\Pi^{ext(\varphi)})$ can determine if $\varphi \in Th_{\exists^*}(\mathcal{M})$.

2.1 Corollary: Quantifier-Free Skolem Arithmetic is in NP

Although not needed for our final result, the technique of theorem 1 is of independent interest. An example is showing that the satisfiability problem for the quantifier-free fragment of Skolem arithmetic is in NP. This result was first proved by Grädel [11] using results by Sieveking and von zur Gathen [9] with a proof that appears, on the surface, to be specific to the arithmetic theories. We reproduce here the relevant definitions for the convenience of the reader. For more details see [7, 8, 11].

Informally, Skolem arithmetic is a fragment of Peano arithmetic with multiplication (and equality) but no addition. Its decidability properties are based on representing natural numbers in terms of their prime factors, which makes the structure isomorphic to a power structure with finitely many non-zero elements.

Definition 3. *Let e be a constant denoting an element in the base structure $\mathcal{M}$. The weak power structure Π^* over I has domain:*

$$M_*^I = \{f : I \to M \mid f(i) \neq e \text{ for only finitely many } i \in I\}$$

and interprets the symbols of L as in the power structure.

Definition 4. *Skolem arithmetic, abbreviated by SA, is the first-order theory of the structure $\langle \mathbb{N} \setminus \{0\}, \cdot, | \rangle$.*

Note that equality is easily definable writing $a|b \wedge b|a$ for $a = b$.

Lemma 2. *SA is isomorphic to the weak power of $\langle \mathbb{N}, +, \leq \rangle$ over $\mathbb{N}$.*

Proof. We give an isomorphism [13, Section 1.2] between the structures of SA and the weak product $\langle \mathbb{N}, +, \leq \rangle_*^{\mathbb{N}}$. Let $\varphi : \mathbb{N} \to \mathbb{N}_*^{\mathbb{N}}$ be a function given by $n \mapsto (e_0, e_1, \ldots, e_i, \ldots)$, where e_i are the exponents of the unique factor decomposition given by the fundamental theorem of arithmetic. Here, the tuples are taken with respect to some previously agreed order of primes $p_0, p_1, \ldots, p_i, \ldots$.

φ is well-defined because a natural number can only have a finite number of prime factors. We use the constant symbol 0 for the constant e appearing in definition 3. Furthermore, it is clear that φ is bijective by the fundamental theorem. It also respects the function and relation symbols. Thus, φ is an isomorphism.

Since two isomorphic structures are also elementary equivalent [13, Section 2.3], both structures satisfy the same first-order statements. In particular, the existential sentences of both structures coincide. We can now show:

Corollary 1. *$Th_{\exists^*}(SA) \in NP$.*

Proof. By Lemma 2, $Th_{\exists^*}(SA) \in$ NP if and only if $Th_{\exists^*}(\langle \mathbb{N}, +, \leq \rangle_*^{\mathbb{N}}) \in$ NP. A variation of the verifier in figure 1, checking that, in the case that $|I|$ is infinite, 0^n is a solution of φ_0 seen as a formula in $\mathcal{M}$, shows that this is equivalent to $Th_{\exists^*}(\langle \mathbb{N}, +, \leq \rangle) \in$ NP. But this last statement follows from the NP complexity of the satisfiability problem for quantifier-free formulas of Presburger arithmetic.

3 Explicit Sets of Indices and a Polynomial Verifier for QFBAPA

To prepare for generalization of the result from the previous section, we now review the QFBAPA complexity [16] using the notation of the present paper. The intuition for our approach is that the verifier of figure 1 is solving constraints on the array indices which can be schematically presented as in figure 2. The figure presents a Venn region of sets defined by formulas of L. All indices must remain within the boundaries of the main region A. This region corresponds to the positive literals of φ in figure 1: $\bigwedge_{i=1}^{l} R_i(t_1^i, \ldots, t_{a_i}^i)$. The negative literals

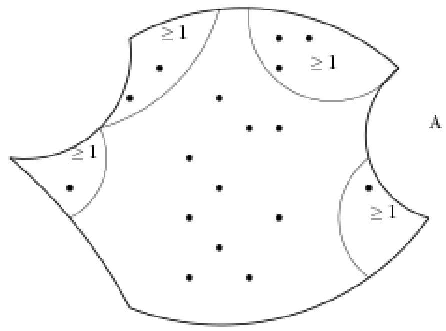

Fig. 2. An example Venn region with product constraints.

$\bigwedge_{j=l+1}^{k} \neg R_j(t_1^j, \ldots, t_{a_j}^j)$ generate existential constraints. These can be interpreted as requiring a cardinality greater or equal than one in certain subregions of A.

To generalize our result we use the logic BAPA [14], whose language allows to express boolean algebra and cardinality constraints on sets. The satisfiability problem for the quantifier-free fragment of BAPA, often written as QFBAPA, is in NP (see section 3 of [16]). Figure 3 shows the syntax of the fragment. 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 and $MAXC$ gives the cardinality of $\mathcal{U}$. We will assume this cardinality to be finite for simplicity of the presentation. That said, we believe it is straightforward to generalize the NP membership result to the case where the universe is infinite and the language contains an additional predicate expressing finiteness of a set [15, Section 3], which is useful for expressing generalizations of weak powers through formulas stating that the set of indices where a condition holds is finite.

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

Fig. 3. QFBAPA's syntax

The basic argument to establish NP complexity of QFBAPA is based on a theorem by Eisenbrand and Shmonin [6], which in our context says that any element of an integer cone can be expressed in terms of a polynomial number of generators. Figure 4 gives a verifier for this basic version of the algorithm. The algorithm uses an auxiliary verifier V_{PA} for the quantifier-free fragment of Presburger arithmetic.

The key step is showing equisatisfiability between 2.(b) and 2.(c). If $x_1, \ldots, x_k$ are the variables occurring in $b_0, \ldots, b_p$ then we write $p_\beta = \bigcap_{i=1}^{k} x_i^{e_i}$ for $\beta = (e_1, \ldots, e_k)$, $[\![b_i]\!]_{\beta_j}$ the evaluation of b_i as a propositional formula with the assignment given in β and introduce variables $l_\beta = |p_\beta|$. Now, $|b_i| = \sum_{j=0}^{2^e-1} [\![b_i]\!]_{\beta_j} l_{\beta_j}$, so the restriction $\bigwedge_{i=0}^{k} |b_i| = k_i$ becomes $\bigwedge_{i=0}^{p} \sum_{j=0}^{2^e-1} [\![b_i]\!]_{\beta_j} l_{\beta_j} = k_i$ which can be seen as a linear combination in $\{([\![b_0]\!]_{\beta_j}, \ldots, [\![b_p]\!]_{\beta_j}).j \in \{0, \ldots, 2^e - 1\}\}$. Eisenbrand-Shmonin's result allows then to derive 2.(c) for N polynomial in $|x|$. In the other direction, it is sufficient to set $l_{\beta_j} = 0$ for $j \in \{0, \ldots, 2^e - 1\} \setminus \{i_1, \ldots, i_N\}$. Thus, we have:

Theorem 2 ([16]). *QFBAPA is in NP.*

4 NP Complexity for QFBAPAI

We are now ready to present our main result, which extends NP membership of product structures and of QFBAPA to the situation where we interpret QFBAPA sets as sets of indices (subsets of the set I) in which quantifier-free formulas hold.

Definition 5. *We consider the satisfiability problem for QFBAPA formulas F whose set variables are index sets defined by quantifier-free formulas φ_i of L applied to either component theory constants or to components of the array variables:*

$\exists c_1, \ldots, c_m . \exists x_1, \ldots, x_n .$

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

We call this problem QFBAPAI, standing for interpreted QFBAPA.

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

Proof. 1) Let V_{QFBAPA} be a polynomial time verifier for QFBAPA and let V_C be a polynomial time verifier for the component theory. Figure 5 gives a verifier V for QFBAPAI. We abbreviate $(x_1, \ldots, x_n)$ by $\overline{x}$ and $(c_1, \ldots, c_m)$ by $\overline{c}$.

$\Rightarrow$) If $x \in$ QFBAPAI then there exist $\overline{c}, \overline{s}$ satisfying:

$$F(S_1, \ldots, S_k) \wedge \bigwedge_{i=1}^{k} S_i = \{r \in I | \varphi_i(\overline{s}(r), \overline{c})\}$$

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

1. Interpret w as:
   (a) a list of indices $i_1, \ldots, i_N \in \{0, \ldots, 2^e - 1\}$ where e is the number of set variables in x.
   (b) a certificate C for V_{PA} on input x' defined below.
2. Transform x into x' by:
   (a) rewriting boolean expressions according to the rules:
   $$b_1 = b_2 \mapsto b_1 \subseteq b_2 \wedge b_2 \subseteq b_1$$
   $$b_1 \subseteq b_2 \mapsto |b_1 \cap b_2^c| = 0$$
   (b) introducing variables k_i for cardinality expressions:
   $$G \wedge \bigwedge_{i=0}^{p} |b_i| = k_i$$
   where G is the resulting quantifier-free Presburger arithmetic formula.
   (c) rewriting into:
   $$G \wedge \bigwedge_{j=i_1,\ldots,i_N} l_{\beta_j} \geq 0 \wedge \bigwedge_{i=0}^{p} \sum_{j=i_1,\ldots,i_N} [\![b_i]\!]_{\beta_j} \cdot l_{\beta_j} = k_i$$
3. Run V_{PA} on $\langle x', C\rangle$.
4. Accept iff V_{PA} accepts.

Fig. 4. Verifier for QFBAPA

Define $S_i := \{r \in I | \varphi_i(\overline{s}(r), \overline{c})\}$. Then, the method of theorem 2 applied to $F(S_1, \ldots, S_k)$ yields a formula $G \wedge \bigwedge_{i=0}^{p} |b_i| = k_i$. Using $|b_i| = \sum_{\beta \models b_i} \left| \bigcap_{i=1}^{k} S_i^{\beta(i)} \right|$ and setting $p_\beta := \bigcap_{i=1}^{k} S_i^{\beta(i)}, l_\beta := \left| p_\beta \right|$, yields $G \wedge \bigwedge_{i=0}^{p} \sum_{j=0}^{2^e-1} [\![b_i]\!]_{\beta_j} \cdot l_{\beta_j} = k_i$. Remove those β where $l_\beta = 0$. Since:

$$p_\beta = \bigcap_{i=1}^{k} \{r \in I | \varphi_i(\overline{s}(r), \overline{c})\}^{\beta(i)} = \left\{ r \in I \,\middle|\, \bigwedge_{i=1}^{k} \varphi_i(\overline{s}(r), \overline{c})^{\beta(i)} \right\}$$

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

1. Interpret w as:
   (a) a list of indices $i_1, \dots, i_N \in \{0, \dots, 2^e - 1\}$ where e is the number of set variables in y.
   (b) a certificate C for V_C on input y defined below.
   (c) a certificate C' for V_{PA} on input y' defined below.
   (d) a bit b indicating if the solution sets cover the whole I.
2. Set $y = \exists \overline{c}, \overline{x}_1, \dots, \overline{x}_N. \bigwedge_{\beta_j \in \{i_1,\dots,i_N\}} \bigwedge_{i=1}^{k} \varphi_i(\overline{x}_j, \overline{c})^{\beta_j(i)} \quad (*)$.
3. Set $y' = \exists S'_1, \dots, S'_k . F(S'_1, \dots, S'_k) \wedge \bigwedge_{\beta_j \in \{i_1,\dots,i_N\}} \bigcap_{i=1}^{k} {S'_i}^{\beta_j(i)} \neq \emptyset \quad (**)$.
4. If $b = 0$ then set $(*) = \wedge \neg \bigvee_{\beta_j \in \{i_1,\dots,i_N\}} \bigwedge_{i=1}^{k} \varphi_i(\overline{x}_0, \overline{c})^{\beta_j(i)}$ and add $\overline{x}_0$ as a top-level existential quantifier.
   If $b = 1$ then set $(**) = \wedge \bigcup_{\beta_j \in \{i_1,\dots,i_N\}} \bigcap_{i=1}^{k} {S'_i}^{\beta_j(i)} = I$.
5. Run V_C on $\left\langle y, C \right\rangle$.
6. Run V_{QFBAPA} on $\left\langle y', \langle \{i_1, \dots, i_N\}, C' \rangle \right\rangle$.
7. Accept iff all runs accept.

Fig. 5. Verifier for QFBAPA interpreted over index-sets.

this includes those β such that $\bigwedge_{i=1}^{k} \varphi_i(\overline{s}(r), \overline{c})^{\beta(i)}$ is not satisfiable. We obtain a reduced set of indices $\mathcal{R} \subseteq \{0, \dots, 2^e - 1\}$ where $G \wedge \bigwedge_{i=0}^{p} \sum_{\beta \in \mathcal{R}} [\![b_i]\!]_\beta \cdot l_\beta = k_i$. Eisenbrand-Shmonin's theorem yields a polynomial family of indices such that $G \wedge \bigwedge_{i=0}^{p} \sum_{\beta \in \{i_1,\dots,i_N\} \subseteq \mathcal{R}} [\![b_i]\!]_\beta \cdot l'_\beta = k_i$ for non-zero l'_β.

For each $\beta \in \{i_1, \dots, i_N\}$, since $l_\beta \neq 0$, there exists $r_\beta \in I$ such that $\bigwedge_{i=1}^{k} \varphi_i(\overline{s}(r_\beta), \overline{c})^{\beta(i)}$. So the formula y without (*) is satisfied.

The satisfiability of the cardinality restrictions on l'_β implies the existence of sets of indices S'_i such that for each $\beta \in \{i_1, \dots, i_N\}$, $|p'_\beta| = l'_\beta$. Observe that $|I| = \sum_{\beta \in \mathcal{R}} l_\beta$. Distinguish two cases:

– If $|I| > \sum_{\beta \in \{i_1,\dots,i_N\}} l'_\beta$ then there is at least one index r_0 such that $\overline{s}(r_0)$ satisfies $\bigwedge_{i=1}^{k} \varphi_i(\overline{s}(r_0), \overline{c})^{\beta(i)}$ for $\beta \notin \{i_1, \dots, i_N\}$. Therefore, the formula y with (*) is satisfied. In this case, define:

$$\overline{s}'(r) = \begin{cases} \overline{s}(r_\beta) & \text{if } r \in p'_\beta \text{ and } \beta \in \{i_1, \dots, i_N\} \\ \overline{s}(r_0) & \text{otherwise} \end{cases}$$

and choose $b = 0$.

– If $|I| = \sum_{\beta \in \{i_1,\dots,i_N\}} l'_\beta$ then define:

$$\overline{s}'(r) = \begin{cases} \overline{s}(r_\beta) & \text{if } r \in p'_\beta \text{ and } \beta \in \{i_1, \dots, i_N\} \end{cases}$$

Here we choose $b = 1$.

In any case, the formula y that V_C receives as input is satisfied. Since N is polynomial in $|x|$, this gives a polynomially-sized certificate C such that V_C accepts $\langle y, C \rangle$ in polynomial time.

Let $S''_i = \{r \in I | \varphi_i(\overline{s}'(r), \overline{c})\}$. Then $S''_1, \dots, S''_k$ satisfy y' by construction:

– Observe that for each $\beta \in \{i_1, \dots, i_N\}$, $p''_\beta = p'_\beta$.

– For each $\beta \in \{i_1, \dots, i_N\}$, $p''_\beta \neq \emptyset$, since $l'_\beta \neq 0$.

– If $b = 1$ then $\bigcup_{\beta \in \{i_1,\dots,i_N\}} p''_\beta = I$ since $|I| = \sum_{\beta \in \{i_1,\dots,i_N\}} l'_\beta$.

– The cardinality restrictions are satisfied by definition.

Again, since N is polynomial in $|x|$, $|y'|$ is polynomial in $|x|$ too. By the above, it is also satisfiable. Thus, there exists a polynomially-sized certificate C' for V_{PA} such that V_{QFBAPA} accepts $\langle \{i_1, \dots, i_N\}, C' \rangle$ in polynomial time. So V accepts $\langle x, \langle \{i_1, \dots, i_N\}, C, C', b \rangle \rangle$ in polynomial time.

$\Leftarrow$) If V accepts $\langle x, w \rangle$ in polynomial time then:

– $\left\langle y, C \right\rangle$ is accepted by V_C, so there is a tuple $\overline{c}$ and for each $\beta \in \{i_1, \dots, i_N\}$, there are tuples s_β, such that $\bigwedge_{i=1}^{k} \varphi_i(s_\beta(1), \dots, s_\beta(n), \overline{c})^{\beta(i)}$.

– $\langle y', \langle \{i_1, \ldots, i_N\}, C' \rangle\rangle$ is accepted by V_{QFBAPA}, so there exist sets S'_i such that:

$$F(S'_1, \ldots, S'_k) \wedge \bigwedge_{\beta \in \{i_1, \ldots, i_N\}} \bigcap_{i=1}^{k} S_i'^{\beta(i)} \neq \emptyset$$

Interpreting S'_i as index sets, we define an array $\overline{s}$ distinguishing two cases:

– If $b = 0$ then V_C accepts:

$$\left\langle \exists \overline{c}, \exists \overline{x}_1, \ldots, \overline{x}_N, \overline{x}_0. \ldots \neg \bigvee_{\beta \in \{i_1, \ldots, i_N\}} \bigwedge_{i=1}^{k} \varphi_i(\overline{x}_0, \overline{c})^{\beta(i)}, C \right\rangle$$

Let s_0 be a satisfying tuple for $\overline{x}_0$. Define:

$$\overline{s}(r) = \begin{cases} s_\beta & \text{if } r \in p'_\beta \text{ and } \beta \in \{i_1, \ldots, i_N\} \\ s_0 & \text{otherwise} \end{cases}$$

– If $b = 1$ then S'_i satisfies $\bigcup_{\beta \in \{i_1, \ldots, i_N\}} \bigcap_{i=1}^{k} S_i'^{\beta(i)} = I$. Define:

$$\overline{s}(r) = \begin{cases} s_\beta & \text{if } r \in p'_\beta \text{ and } \beta \in \{i_1, \ldots, i_N\} \end{cases}$$

Then, by construction, $\overline{c}, \overline{s}$ form a solution of:

$$\exists \overline{c}, \overline{x}. F(S_1, \ldots, S_k) \wedge \bigwedge_{i=1}^{k} S_i = \{r \in I \mid \varphi_i(\overline{x}(r), \overline{c})\}$$

For each $\beta \in \{i_1, \ldots, i_N\}$:

$$p_\beta = \left\{ r \in I \middle| \bigwedge_{i=1}^{k} \varphi(\overline{s}(r), \overline{c})^{\beta(i)} \right\} = p_{\beta'}$$

so the cardinality conditions are met.

2) Conversely, if we assume that QFBAPAI $\in$ NP then it is easy to give a poly-time reduction from $Th_{\exists^*}(\mathcal{M})$ to QFBAPAI and using the preservation property of NP obtain that $Th_{\exists^*}(\mathcal{M}) \in$ NP too. To be concrete, the reduction maps each formula $\exists y_1, \ldots, y_n. \varphi(y_1, \ldots, y_n)$ to the formula:

$$\exists y_1, \ldots, y_n. I = \{r \in I | \varphi(y_1(r), \ldots, y_n(r))\}$$

The reduction is clearly polynomial and the correctness property also holds: a solution of the component formula can be repeated to get a solution of the array formula and any component of the array formula gives a solution for the component formula.

5 Combination with the Array Theory

In this section we show, through a syntactic translation, that the conventional and generalized array operations can be expressed in QFBAPAI. The combinatory array logic (CAL) fragment of de Moura and Bjørner [19] can be presented as a multi-sorted structure:

$$\mathcal{A} = \langle A, I, V, \cdot[\cdot], \mathrm{store}(\cdot,\cdot,\cdot), \{c_i^v\}, \{f_i^v\}, \{R_i^v\}, \{c_j\}, \{f_j\}, \{R_j\}\rangle$$

where $\mathcal{V} = \langle V, \{c_i^v\}, \{f_i^v\}, \{R_i^v\}\rangle$ is the structure modelling array elements and I is a non-empty set which parametrizes the read $(\cdot[\cdot])$ and store $(store(\cdot,\cdot,\cdot))$ operations. Finally, $\Pi = \mathcal{V}^I = \langle A, \{c_j\}, \{f_j\}, \{R_j\}\rangle$ is the power structure with base $\mathcal{V}$ and index set I. Note that, according to the definition of a power structure, there is a one to one correspondence between the symbols of the component language and those of the array language. We use the superscript v to distinguish between value symbols and power structure symbols. The read and store operations use a mixture of sorts. The read operation corresponds to a parametrized version of the canonical projection homomorphism of product structures [13]. It is interpreted as:

$$\begin{aligned} \cdot[\cdot] : & A \times I \longrightarrow V \\ (a, i) & \longmapsto a(i) \end{aligned}$$

On the other hand, the store operation lacks a canonical counterpart in model theory. It is to be interpreted as the function:

$$\begin{aligned} store : & A \times I \times V \longrightarrow A \\ (a, i, v) & \longmapsto store(a, i, v) \end{aligned}$$

where:

$$store(a, i, v)(j) = \begin{cases} a(j) & \text{if } j \neq i \\ v & \text{if } j = i \end{cases}$$

The goal of this section is to give a satisfiability preserving translation from CAL to QFBAPAI in such a way that the size of the transformed formula is bounded by a polynomial in the size of the original input. Since CAL formulas cannot express equicardinality constraints, $|A| = |B|$, this means that we have increased the expresive power of the fragment while retaining the same complexity bound. The translation is written in terms of a list of basic primitives explained below. The complete translation is shown in figure 6.

Since we are dealing with quantifier-free formulas, we map the propositional structure to boolean operations and concentrate in the encoding of non-propositional symbols. These symbols are atomic relations in either the component theory or the array theory.

Relations in the component theory. An atomic formula of the component theory has the following shape:

$$R^v(f_i\{a_1[i_1], \ldots, a_n[i_n], c_1, \ldots, c_m\})$$

Here and in the rest of the section we use the notation $R(f_i\{p_1, \ldots, p_n\})$ for a list of arity(R) function terms of the form $f_i\{p_1, \ldots, p_n\}$ where f_i is a function symbol using a subset of the parameters in $\{p_1, \ldots, p_n\}$. Both f_i and the parameters p_i must have the same sort as R. We use the letter a to denote either an array variable x or a *store* term and the letter v to denote an element value in contrast to a read term $a[j]$.

We transform the above constraint using the following rules:

1. ABSTRACT READS (≤ 1): if there are more than two parameters that use the read function $\cdot[\cdot]$ applied to a variable, we rewrite all occurrences $x_j[i]$ but one into value constants x_{ji}. Note that a read from a constant array need not create a new value variable. Instead, we rewrite $c[i]$ as c^v. In this case, no further changes are required in later stages.
2. IMPOSE READS: for each abstracted read $x_j[i]$ add the condition:
$$\{l \in I | x_j(l) = x_{ji}\} \supseteq \{i\}$$
3. ABSTRACT WRITES: rewrite the innermost store operations $store(x, i, v)$ into array variables x_{iv}.
4. IMPOSE WRITES: for each abstracted store x_{iv}, we impose the condition:
$$\{l \in I | x_{iv}(l) = v\} \supseteq \{i\} \wedge \{l \in I | x_{iv}(l) = x(l)\} \supseteq \{i\}^c$$

This process is repeated until there is no change in the manipulated formula. In this last case, we have obtained a relation:

$$R^v(f_i\{x[i], \text{abs}_1, \ldots, \text{abs}_k, c_1, \ldots, c_m\})$$

where abs_j are the newly introduced array or value variables. We then perform one last step:

5. IMPOSE VALUE CONSTRAINT: add the constraint:
$$\{l \in I | R^v(f_i\{x(l), \text{abs}_1, \ldots, \text{abs}_k, c_1, \ldots, c_m\})\} \supseteq \{i\}$$

Relations in the power structure theory. An atomic formula of the product theory has the shape:

$$R(f_i\{a_1, \ldots, a_n, c_1, \ldots, c_m\})$$

where $c_1, \ldots, c_m$ are to be interpreted as constants of the product. We repeat a variation of the steps 1-4 where ABSTRACT READS (≤ 1) is changed into ABSTRACT READS ($= 0$). The only difference between the two is that the latter removes all reads. The result of this operation is a relation:

$$R(f_i\{x_1, \ldots, x_s, \text{abs}_1, \ldots, \text{abs}_k, c_1, \ldots, c_m\})$$

where abs_j are the newly introduced array variables. We cannot have value variables since in this case value expressions are not top-level.

In this case, we do the following as a last step:

Given a formula ψ of CAL in negation normal form:

1. Rewrite $\wedge \mapsto \cap, \vee \mapsto \cup$ and $\neg \mapsto \cdot^c$.
2. Consider the following auxiliary procedure P receiving one bit b as parameter.

   Repeat until no more constraints are added:

   (a) If $b = 0$ then ABSTRACT READS($= 0$)
   else ABSTRACT READS($<= 1$).
   (b) IMPOSE READS
   (c) ABSTRACT WRITES
   (d) IMPOSE WRITES
3. For each relation in the array theory call P with $b = 0$.
4. For each relation in the component theory call P with $b = 1$.

Fig. 6. Translation scheme from CAL to QFBAPAI.

5. IMPOSE ARRAY CONSTRAINT: add the constraint:

$$\{l \in I | R(f_i\{x_1(l), \ldots, x_s(l), \text{abs}_1(l), \ldots, \text{abs}_k(l), c_1, \ldots, c_m\})\} = I$$

Satisfiability preservation and size of the transformed formula. It is clear that each transformation step yields an equisatisfiable formula. In particular, this ensures that the order of introduction of new variables does not matter. Even if the transformed formula may contain duplicates, the existence of a solution is equivalent in both formulas.

Regarding the size of the transformed formula, we observe that during the analysis of a relation we create as many variables as the size of such relation. Thus, the number of variables created is at most linear in the size of the formula. This means that the total number of variables and constants that are either present in the original formula or created by the algorithm, C, is in $O(|\psi|)$.

The creation of each variable implies the creation of at most three restrictions: this happens in the IMPOSE WRITES case, where the third restriction specifies that the size of $\{i\}$ is one. Each restriction uses at most two variables, so we can encode it using $O(log_2(|\psi|))$ space. Thus, to encode all the added restrictions we need $O(|\psi| \log_2(|\psi|))$ space.

Each relation generates an additional constraint, which may use all the set of C variables. So we may need up to $O(|\psi| \cdot \log_2(|\psi|))$ to encode the constraint. Since there are $O(|\psi|)$ relations, we need $O(|\psi|^2 \log_2(|\psi|))$ space to encode them.

Overall, the size increase is in $O(|\psi|^2 \log_2(|\psi|)$, as desired to preserve NP complexity.

6 Further Related Work

Our work is related to a long tradition of decision procedures for the theories of arrays [4, 5, 10, 12, 17, 21]. Our direct inspiration is combinatory array logic [19]. We have extended this fragment with cardinality constraints while preserving membership in NP.

In our study, we have given priority to those procedures that decide satisfiability within the NP complexity class. From these, [1] and [5] are the more closely related since they also address counting properties. The main difference with these works is that our index theory is arbitrary and that the element theory is any one in NP. This gives access to a greater degree of compositionality. For instance, we can profit of the properties of QFBAPA to handle infinite cardinalities in the index theory [15]. On the other hand, the work of [5] allows for a great expressivity, achieving NP complexity on particular fragments, but it is PSPACE-complete in the general case.

Other influential works in the theory of integer arrays include [4] and [12]. [4] treats a fragment capable of expressing ordering conditions and Presburger restrictions on the indices. [12] complements the work above based on automata considerations. In both cases, the complexity of the satisfiability problem for the full fragment remains, to our knowledge, open. Parametric theories of arrays include [19, 21] and [3]. However, the line of work in [3] as consolidated in the doctoral thesis [2], only shows decidability and NEXPTIME completeness on particular instances. None of [2–4, 12, 19, 21] treat cardinality constraints.

7 Conclusion and Future Work

We have identified the model theoretic structure behind a state of the art fragment of the theory of arrays. We have given self-contained proofs of complexity which shed light on the underlying constraints that the fragment addresses. This has allowed to generalize the fragment to encode arbitrary cardinality constraints. Our work also shows that the set variables of BAPA can be interpreted to encode useful restrictions.

As future work, we plan to build on the efforts in [19], to provide an efficient implementation of the fragment. We would also like to perform a cross-fertilization with other fragments of the theory of arrays providing counting capabilities, while exploring the interactions between their seemingly different foundations.

Postscript

The methodology used in the paper can be split in two parts. The first part projects the indices of the product structure on a boolean algebra which contains the desired features that one wants to model. Different techniques are used to show that the boolean algebra admits small solutions. None of these are new. Observe thought that varying the algebra allows to describe more interesting properties, such as considering only contiguous chunks of memory. The second part rebuilds a solution to the original product structure. The model that gets rebuilt is known as boolean power [13, section 9.7]. This is the truly new part.

One of the questions that the reviewers of the paper had was about the relation between theorem I and theorem III. One should see theorem I as embedding the power structure in propositional logic, which corresponds to the boolean algebra $\langle \{0,1\}, \wedge, \vee, \neg \rangle$. It is natural to generalize the algebra and in particular, with concrete algebras (namely having a "finite" operator), one recovers the results of Skolem arithmetic.

Note that conveniently extended to model the original CAL language, this paper shows the restrictions that de Moura and Bjørner wrote in their paper, namely, they said that the cardinalities of sets and bags could not be expressed.

References

1. Alberti, F., Ghilardi, S., Pagani, E.: Cardinality constraints for arrays (decidability results and applications). Formal Methods in System Design **51**(3), 545–574 (Dec 2017). https://doi.org/10.1007/s10703-017-0279-6
2. Alberti, F.: An SMT-based verification framework for software systems handling arrays. Ph.D. thesis, Università della Svizzera Italiana (Apr 2015), `http://www.falberti.it/thesis/phd.pdf`
3. Alberti, F., Ghilardi, S., Sharygina, N.: Decision Procedures for Flat Array Properties. Journal of Automated Reasoning **54**(4), 327–352 (Apr 2015). https://doi.org/10.1007/s10817-015-9323-7
4. Bradley, A.R., Manna, Z., Sipma, H.B.: What's Decidable About Arrays? In: Emerson, E.A., Namjoshi, K.S. (eds.) Verification, Model Checking, and Abstract Interpretation. pp. 427–442. Lecture Notes in Computer Science, Springer, Berlin, Heidelberg (2006). https://doi.org/10.1007/11609773_28
5. Daca, P., Henzinger, T.A., Kupriyanov, A.: Array Folds Logic. In: Chaudhuri, S., Farzan, A. (eds.) Computer Aided Verification. pp. 230–248. Lecture Notes in Computer Science, Springer International Publishing, Cham (2016). https://doi.org/10.1007/978-3-319-41540-6_13
6. Eisenbrand, F., Shmonin, G.: Carathéodory bounds for integer cones. Operations Research Letters **34**(5), 564–568 (Sep 2006). https://doi.org/10.1016/j.orl.2005.09.008
7. Feferman, S., Vaught, R.: The first order properties of products of algebraic systems. Fundamenta Mathematicae **47**(1), 57–103 (1959), `https://eudml.org/doc/213526`
8. Ferrante, J., Rackoff, C.W.: The Computational Complexity of Logical Theories. Lecture Notes in Mathematics, Springer-Verlag, Berlin Heidelberg (1979). https://doi.org/10.1007/BFb0062837
9. Gathen, J.v.z., Sieveking, M.: A Bound on Solutions of Linear Integer Equalities and Inequalities. Proceedings of the American Mathematical Society **72**(1), 155–158 (1978). https://doi.org/10.2307/2042554, publisher: American Mathematical Society
10. Ghilardi, S., Nicolini, E., Ranise, S., Zucchelli, D.: Decision procedures for extensions of the theory of arrays. Annals of Mathematics and Artificial Intelligence **50**(3), 231–254 (Aug 2007). https://doi.org/10.1007/s10472-007-9078-x

11. Grädel, E.: Dominoes and the complexity of subclasses of logical theories. Annals of Pure and Applied Logic **43**(1), 1–30 (Jun 1989). https://doi.org/10.1016/0168-0072(89)90023-7
12. Habermehl, P., Iosif, R., Vojnar, T.: What Else Is Decidable about Integer Arrays? In: Amadio, R. (ed.) Foundations of Software Science and Computational Structures. pp. 474–489. Lecture Notes in Computer Science, Springer, Berlin, Heidelberg (2008). https://doi.org/10.1007/978-3-540-78499-9_33
13. Hodges, W.: Model Theory. Encyclopedia of Mathematics and its Applications, Cambridge University Press, Cambridge (1993). https://doi.org/10.1017/CBO9780511551574
14. Kuncak, V., Nguyen, H.H., Rinard, M.: Deciding Boolean Algebra with Presburger Arithmetic. Journal of Automated Reasoning **36**(3), 213–239 (Apr 2006). https://doi.org/10.1007/s10817-006-9042-1
15. Kuncak, V., Piskac, R., Suter, P.: Ordered Sets in the Calculus of Data Structures. In: Dawar, A., Veith, H. (eds.) Computer Science Logic. pp. 34–48. Lecture Notes in Computer Science, Springer, Berlin, Heidelberg (2010). https://doi.org/10.1007/978-3-642-15205-4_5
16. Kuncak, V., Rinard, M.: Towards Efficient Satisfiability Checking for Boolean Algebra with Presburger Arithmetic. In: Pfenning, F. (ed.) Automated Deduction – CADE-21. pp. 215–230. Lecture Notes in Computer Science, Springer, Berlin, Heidelberg (2007). https://doi.org/10.1007/978-3-540-73595-3_15
17. McCarthy, J.: Towards a Mathematical Science of Computation. In: Colburn, T.R., Fetzer, J.H., Rankin, T.L. (eds.) Program Verification: Fundamental Issues in Computer Science, pp. 35–56. Studies in Cognitive Systems, Springer Netherlands, Dordrecht (1993). https://doi.org/10.1007/978-94-011-1793-7_2
18. Mostowski, A.: On direct products of theories. The Journal of Symbolic Logic **17**(1), 1–31 (Mar 1952). https://doi.org/10.2307/2267454, publisher: Cambridge University Press
19. de Moura, L., Bjorner, N.: Generalized, efficient array decision procedures. In: 2009 Formal Methods in Computer-Aided Design. pp. 45–52. IEEE, Austin, TX (Nov 2009). https://doi.org/10.1109/FMCAD.2009.5351142
20. de Moura, L., Bjørner, N.: Z3: An Efficient SMT Solver. In: Tools and Algorithms for the Construction and Analysis of Systems (TACAS). vol. 4963, pp. 337–340. Springer (2008). https://doi.org/10.1007/978-3-540-78800-3_24, series Title: LNCS
21. Stump, A., Barrett, C., Dill, D., Levitt, J.: A decision procedure for an extensional theory of arrays. In: Proceedings 16th Annual IEEE Symposium on Logic in Computer Science. pp. 29–37. IEEE Comput. Soc, Boston, MA, USA (2001). https://doi.org/10.1109/LICS.2001.932480