On Regular Relations in Parametric Array Theories

Rodrigo Raya[1]

[1]Max-Planck Institute for Software Systems, Kaiserslautern, Germany

Abstract

Parametric array theories are extensions of the quantifier-free theory of arrays with relations that hold component-wise. Unlike more expressive theories of arrays they allow specifying linear cardinality constraints on interpreted sets of indices, a notion close to the Härtig quantifier from model theory. We apply the notion of generalised power of a structure to study the satisfiability problem of parametric array theories. We show that reasoning about component-wise relations, linear cardinality constraints and succinct regular relations can be done efficiently by reduction to propositional satisfiability. We indicate how our techniques can be adapted to theories of trees.

Keywords

decision procedures, satisfiability modulo theories, symbolic automata

1. Introduction

Many abstractions in computer science and mathematics are naturally modelled as collections of objects of certain type. When addressing the problem of automatically verifying properties of such abstractions it becomes crucial to choose an adequate language in which to express the properties of interest.

Our research is influenced by work in the area of deductive software verification [5, 23]. Research in this area led to the development of specialised algorithms that determine the validity of formulas in restricted fragments of first-order logic. The resulting algorithms are today studied in the so-called satisfiability modulo theories (SMT) framework.

Starting with [21], first-order theories under the name of "array theories" have been studied and, along the years, new decidable fragments and applications have been found. Bradley [6] carried out a systematic exploration of a very expressive and decidable fragment known as the "array property fragment". Most notably, this fragment allowed to express the property of an array being ordered while having an efficiently decidable satisfiable problem under mild restrictions. Moreover, Bradley's work showed that minor variations to the fragment's syntax would led to undecidability, by reduction from Hilbert's tenth problem.

In spite of the above, and after Bradley's work, a family of decidable array theories has been used in the verification of so-called array-based systems [17, 26, 13, 27, 15]. This framework has been used to model sequential programs manipulating arrays and lists, as well as parameterised concurrent systems with local and shared variables [3, 1, 20]. These programs are essential in specialised computing scenarios such as data-base driven systems [4] or business processes [15].

In [30, 29, 31, 32], we have investigated the structure of these array theories. We have observed that they extend classical array theories with point-wise relations, which are defined as in the element theory for every component of the arrays. The conclusion of our investigation is that these point-wise relations are the essential difficulty when designing decision procedures for these theories. In particular, we have described how the satisfiability problem of these theories can be reduced in polynomial time to the satisfiability problem of the theory of a power structure [28] and that the latter admits an efficient procedure for eliminating existential quantifiers [30].

Our results are applicable to a variety of array theories from the literature [10, 16, 14, 1] to which we refer to as "parametric" array theories since they often allow to be instantiated with different index and element theories. An interesting feature of parametric array theories is that the componentwise

ARQNL 2024: Automated Reasoning in Quantified Non-Classical Logics, 1 July 2024, Nancy, France

rraya@mpi-sws.org (R. Raya)

0000-0002-0866-9257 (R. Raya)

relations only require one universal quantifier to be expressed. For instance, one may define the addition of two arrays a and b as:

$$a + b = c \text{ if and only if for every } i \in I, a(i) + b(i) = c(i)$$

This is in contrast to other array theories such as the array property fragment [5], which allow properties using several related universally quantified indices. For instance, one may define in this fragment the property of an array being ordered:

$$a \text{ is ordered if and only if for every } i, j \in I, i \leq j \text{ implies } a[i] \leq a[j]$$

While "array properties" in [5] allow several universal quantifiers, this comes at the cost of severe syntactic restrictions. In contrast, parametric array theories offer the possibility of using linear cardinality relations on sets of indices [16, 14, 1, 30][1] as well as constraints on the sums of elements of array variables. These properties are inexpressible in the array property fragment.

These results motivate us to push further our investigation. In this paper, rather than moving to the design of algorithms for first-order theories (which would be justified by the incipient quantifier elimination method of [30]), we choose to further explore the possibilities in the quantifier-free setting which is the one relevant to the satisfiability modulo theories framework.

We take inspiration from the work of Feferman and Vaught [12], who introduced the notion of generalised power of a structure and motivated by the question of decidability of the weak monadic second order theory of one succesor (WS1S), raised by Tarski, discuss generalised powers with this theory of indices in the later sections of their paper. However, deciding WS1S is computationally intractable [34]. Thus, we present the definable relations of the theory in the form of regular expressions. It was proved by Büchi [7] that both formalisms are expressively equivalent. We refer to the relations on sets of indices induced by regular expressions or WS1S formulas as regular relations. [2]

There are several reasons that lead us to think that an extension of array theories with cardinality constraints and regular expressions is worthwhile investigating. First, this extends the work of Alberti, Ghilardi and Pagani [1] since it is well-known that WS1S is more expressive than Presburger arithmetic [35]. Second, this extension allows us to express properties of arrays such as those appearing in array folds logic [9]. While array folds logic only allows folding expressions over one array variable, this restriction does not appear in the fragment that we present. Third, a similar extension but without cardinality constraints has been considered concurrently to our work in [18].

Both in [18] and in our work, it seems that a non-trivial insight for the construction of the decision algorithm is needed. We point out to the reader that this insight is materialised in our paper in the partition variables introduced in Section 4.1. Indeed, since our specifications contain formulas whose interpretations, as sets of indices of the arrays, may overlap, it is essential to ensure that there exists a model adhering to the regular specification regardless of the overlaps in the semantic domain.

Unlike [18], we focus in the case of regular languages which should be more familiar to the readers. Nevertheless, we include a final section pointing out the main ingredients of the extension to regular tree languages. Also, for the sake of clarity, we have focused in cardinality constraints, but it should be clear that an extension to summation constraints is also possible.

Organisation of the paper. The rest of the paper is organised as follows. Section 2 describes generalised powers using specific theories of sets with cardinalities, theories describing their contents, and theories describing regular relations on the indices of these sets. Section 3 describes the satisfiability preserving encoding of arrays in generalised powers. Section 4 gives an algorithm that in polynomial time takes as input a generalised power structure specification and outputs an equivalent formula in

[1] A similar notion appears in the model theory literature under the name of Härtig's quantifier [2].

[2] We had considered regular expressions in our PhD thesis [29]. Here we consider regular expressions over first-order formulas. This formalism has been popularised in recent times under the name of symbolic regular expression and it can also be seen as motivated by Feferman-Vaught's results. This is what we mean by "succinct" regular relations. We also sometimes speak of "ordering" instead of regular relations since regular relations are precisely those expressible in the monadic theory of order [7].

the combination of the quantifier-free theory of Boolean algebra of sets with Presburger arithmetic and the alphabet's theory. Section 5 discusses the applicability of the technique in the setting of theories of trees, connecting to recent work. Section 6 concludes the paper.

2. Generalised powers

Let us start with the definition of generalised power structure as it is given in [12].

Definition 2.1. The generalised power $\mathscr{P}(\mathscr{M}, I)$ of a structure $\mathscr{M} = \langle M, ... \rangle$ is a structure whose carrier set is the set M^I of functions from the (possibly infinite) index set I to the carrier set M of the structure $\mathscr{M}$ and whose relations are interpreted as sets of the form

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

where n is a natural number, Φ is a Boolean algebra expression over $\mathscr{P}(I)$ using the symbols $\subseteq$, $\cup$, $\cap$ or $\cdot^c$ and each set variable S is interpreted as

$$S = \{i \in I \mid \theta(a_1(i), \dots, a_n(i))\}$$

where θ is a formula in the first-order theory of $\mathscr{M}$.

In the following, we will use the term "arrays" for the functional elements in the carrier from a generalised power $\mathscr{P}(\mathscr{M}, I)$, the term "elements" for the members of the carrier set of the structure $\mathscr{M}$ and the term "indices" for the members of the set I. We will use the notation $a(i)$ when we want to emphasize the algebraic perspective and the notation $a[i]$ when we want to emphasize the connection to array theories. In particular, we will use the latter notation when describing how to translate from parametric array theories to generalised powers.

Nothing prevents us from considering set interpretations of the form

$$S = \{i \in I \mid \psi(i)\}$$

where ψ is a formula that refers only to indices in the set I. In fact, this direction is pursued in [1] where a fragment of the theory of arrays is investigated that corresponds to a generalised power whose set interpretations conflate both the theory of indices and the theory of elements using the quantifier-free fragment of Presburger arithmetic to refer to both. We will use different set interpretations for indices and elements. We will use relations of the following form:

$$F(S_1, \dots, S_k) \wedge R(S_1, \dots, S_k) \wedge C(a_1, \dots, a_k) \tag{1}$$

Here F specifies linear cardinality constraints on the shared set variables $S_1, \dots, S_k$. R specifies the regular relations on the set of indices $S_1, \dots, S_k$. Finally, C specifies the componentwise relations on the arrays $a_1, \dots, a_k$. The precise description of Formula (1) occupies the rest of the section.

2.1. Sets of indices

Formulas F, R and C in (1) use variables $S_1, \dots, S_k$ representing subsets of an index set I. This is explicitly shown in (1) for F and R. The variables $S_1, \dots, S_k$ are omitted from formula C to emphasize the role of componentwise relations on array variables. Thus, the variables $S_1, \dots, S_k$ are used to combine the three theories. This approach to theory combination was pioneered in [37]. As we focus on arrays, $I = \mathbb{N}$. Generalisations to trees are also possible and in that case $I = \{0, 1\}^*$.

2.2. Linear cardinality constraints on the sets of indices

$F(S_1, \ldots, S_k)$ is a formula in the quantifier-free theory of Boolean algebra with Presburger arithmetic (QFBAPA) [25]. The syntax of QFBAPA is given in Figure 1. The top-level symbol F presents the Boolean structure of the formula, A stands for the atomic formulas which can be either Boolean algebra expressions on the sets denoted by the symbol B or Presburger arithmetic restrictions on numbers denoted by the symbol T. The operator dvd stands for the divisibility relation, which is used to ensure that the quantifier-free fragment has the same expressive power as the full first-order theory of Boolean algebra with Presburger arithmetic (BAPA)[24]. $\mathcal{U}$ represents the universal set I. Lowercase x and k represent Boolean and integer variables respectively. The remaining interpretations are standard in the respective theories (Boolean algebra of sets or Presburger arithmetic).

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

Figure 1: QFBAPA's syntax

Example 2.1. An example of QFBAPA formula is $|A| > 1 \wedge A \subseteq B \wedge |B \cap C| \leq 2$.

2.3. Componentwise relations on arrays

$C(a_1, \ldots, a_k)$ is a formula specifying componentwise relations. It does so with set interpretations of the form:

$$S_j = \{i \in I \mid \phi_j(\overline{a}(i), \overline{c})\}$$

where $\overline{a}$ denotes a tuple of array variables, $\overline{c}$ denotes a tuple of constants from the element theory and ϕ_j is a formula of the element theory. As in Definition 2.1, $a(i)$ denotes the i-th position of array a and $\overline{a}(i) = (a_1(i), \ldots, a_k(i))$.

Example 2.2. The equality between two arrays a_1 and a_2 can be written in the fragment of (1) as:

$$S = \{i \in I \mid a_1(i) = a_2(i)\} \wedge |S| = |\mathcal{U}|$$

where $\mathcal{U}$ as explained above, represents the universal set I.

2.4. Regular relations on the set of indices

$R(S_1, \ldots, S_k)$ is a formula specifying regular relations in the set of indices. For instance, the array could be specified by the symbolic regular expression:[3]

$$\phi_1(e, \overline{c})(\phi_1(e, \overline{c}) \vee \phi_2(e, \overline{c}))^* \phi_3(e, \overline{c}) \tag{2}$$

This specifies that the first element e of the array satisfies the formula $\phi_1(e, \overline{c})$, then there is a sequence of zero or more elements e satisfying either $\phi_1(e, \overline{c})$ or $\phi_2(e, \overline{c})$ and the last element e satisfies the formula $\phi_3(e, \overline{c})$. Each instantiation of e is different for each witness, while the value of the parameters in $\overline{c}$ must be the same for the whole array. However, this approach conflates the specifications of the indices and the specifications of the elements of the arrays.

[3]There are several possibilities to write regular relations. Here we use regular expressions for economy of notation.

Let us instead write a symbolic version of the regular expression above

$$S_1(S_1 \vee S_2)^* S_3 \tag{3}$$

To relate this symbolic expression and the theory of the elements, we let t be a sequence of bit-strings $t \in (\{0,1\}^3)^*$ and define

$$\begin{aligned} S_1 &= \{i \in I \mid t_1(i) = 1\} \wedge S_1 = \{i \in I \mid \phi_1(a(i), \bar{c})\} \\ S_2 &= \{i \in I \mid t_2(i) = 1\} \wedge S_2 = \{i \in I \mid \phi_2(a(i), \bar{c})\} \\ S_3 &= \{i \in I \mid t_3(i) = 1\} \wedge S_3 = \{i \in I \mid \phi_3(a(i), \bar{c})\} \end{aligned} \tag{4}$$

where t_1, t_2 and t_3 denote, respectively, the first, second and third rows of the sequence and $t_j(i)$ denotes the i-th position in t_j.

Then, satisfiability of (2) is equivalent to satisfiability of

$$\exists t \in (\{0,1\}^3)^*. t \vDash S_1(S_1 \vee S_2)^* S_3 \wedge (4) \tag{5}$$

where $t \vDash R(S_1, \dots, S_k)$ means that the bit-string sequence t satisfies propositionally the regular expression $R(S_1, \dots, S_k)$, that is, there is a word w of propositional formulas over the variables S_1, S_2 and S_3 generated by R such that for each i, $t(i) \vDash w(i)$ propositionally.

Example 2.3. A bit-string sequence t belonging to the language of the symbolic regular expression $S_1(S_1 \vee S_2)^* S_3$ is the following

$$\begin{pmatrix}1\\1\\0\end{pmatrix}\begin{pmatrix}1\\0\\0\end{pmatrix}\begin{pmatrix}0\\1\\0\end{pmatrix}\begin{pmatrix}0\\1\\1\end{pmatrix}\begin{pmatrix}0\\0\\1\end{pmatrix}$$

The values of its rows are respectively $t_1 = 11000$, $t_2 = 10110$ and $t_3 = 00011$.

It satisfies the word of propositional formulas $S_1(S_1 \vee S_2)(S_1 \vee S_2)(S_1 \vee S_2)S_3$ which is generated by $S_1(S_1 \vee S_2)^* S_3$.

We call the bit-string sequences t regular tables or simply tables and write $T(R)$ for the set of all tables satisfying the symbolic regular expression R.

3. Encoding of arrays

Let us briefly mention how would the terms of an array theory, most importantly, array "reads" and "writes", be written in the language of generalised powers, while preserving the satisfiability of formulas.

A componentwise specification is written as in Example 2.2.

An array read is a functional term $a[j]$. To encode this term in generalised powers, we introduce the set variable J representing the singleton set $\{j\}$ and require that $|J| = 1$. We then introduce an element theory variable a_j and require that $\{i \in I \mid a(i) = a_j\} \supseteq J$.

An array write is a functional term $store(a, j, v)$. To encode this term in generalised powers, we introduce a new variable b to stand for the term $store(a, j, v)$ and require that $\{i \in I \mid b(i) = v\} \supseteq J$ and $\{i \in I \mid a(i) = b(i)\} \supseteq \mathcal{U} \setminus J$.

Example 3.1. Consider the array formula from [5].

$$i_1 = j \wedge i_1 \neq i_2 \wedge a[j] = v_1 \wedge store(store(a, i_1, v_1), i_2, v_2)[j] \neq a[j]$$

For each index variable j, i_1, i_2, we introduce set variables J, I_1, I_2 and impose that $I_1 = J$, $I_1 \neq I_2$ and $|I_1| = |I_2| = |J|$.

The term $a[j] = v_1$ is translated into $\{i \in I \mid a(i) = v_1\} \supseteq J$.

We introduce the array variables b for $store(a, i_1, v_1)$ and c for $store(b, i_2, v_2)$. We can then encode the fourth conjunct as $\{i \in I \mid c(i) \neq a(i)\} \supseteq J$. The store operators are encoded as indicated above.

The resulting formula is in the theory of the generalised power and is equisatisfiable to the original.

4. Deciding generalised powers

Let us now give a method to decide satisfiability of formulas of the form (1). These are of the following form:

$$F(S_1, \ldots, S_k) \wedge \exists t \in T(R). \bigwedge_{i=1}^{k} S_i = \{ n \in \mathbb{N} \mid \phi_i(\bar{a}(n), \bar{c}) \} = \{ n \in \mathbb{N} \mid t_i(n) = 1 \} \tag{6}$$

The satisfiability problem requires showing that the following formula is true:

$$\begin{aligned} &\exists a \in \mathscr{D}^*, t \in T(R), \bar{c}. \\ &\qquad F(S_1, \ldots, S_k) \wedge \bigwedge_{i=1}^{k} S_i = \{ n \in \mathbb{N} \mid \phi_i(\bar{a}(n), \bar{c}) \} = \{ n \in \mathbb{N} \mid t_i(n) = 1 \} \end{aligned} \tag{7}$$

where $\mathscr{D}$ is the domain of the array elements.

We show how to compute in polynomial time a formula in the combination of QFBAPA and the existential fragment of the first-order theory of the domain $\mathscr{D}$ such that Formula (7) is equivalent to the computed formula.

Theorem 4.1. There is a polynomial time algorithm that given formula (7) computes an equivalent formula (8) in the combination of QFBAPA and the existential fragment of the first-order theory of $\mathscr{D}$, $Th_{\exists^*}(\mathscr{D})$.

4.1. Construction of the equivalent formula

The equivalent formula has three parts. The first is an existential prefix shared between both QFBAPA and $Th_{\exists^*}(\mathscr{D})$

$$\exists N \leq p(|F|), \exists s \in [m], \sigma : [s] \hookrightarrow [m], \exists \beta_1, \ldots, \beta_N \in \{0,1\}^k.$$

where N is a natural number, p is a polynomial, $|F|$ is the number of symbols used to write F, $[n] := \{1, \ldots, n\}$ abbreviates the set of the first n natural numbers, m is the number of propositional formulas used in M_S, σ is an injection from $[s]$ to $[m]$ and k is the number of set variables used in F.

The second part of the formula is an expression in the theory $Th_{\exists^*}(\mathscr{D})$

$$C(\beta_1, \ldots, \beta_N) := \exists \bar{c}. \bigwedge_{j=1}^{N} \exists e \in D. \phi^{\beta_j}(e, \bar{c})$$

where we use the notation that given the list $\phi_1, \ldots, \phi_k$ of formulas specifying the elements in Formula 6 and given a bit-string $\beta \in \{0,1\}^k$, $\phi^\beta := \bigwedge_{i=1}^{k} \phi_i^{\beta(i)}$.

The third part of the formula is in QFBAPA

$$\begin{aligned} H(c_1, \ldots, c_k, \ldots) := &\rho(c_1, \ldots, c_m) \wedge F(S_1, \ldots, S_k) \wedge \bigwedge_{i=1}^{s} P_i \subseteq p_{L_{\sigma(i)}} \wedge \\ &\bigwedge_{i=1}^{s} |P_i| = c_{\sigma(i)} \wedge \cup_{i=1}^{k} S_i = \cup_{i=1}^{m} p_{L_i} = \cup_{i=1}^{N} p_{\beta_i} = \dot{\cup}_{i=1}^{s} P_i \end{aligned}$$

where ρ is a formula in the existential fragment of Presburger arithmetic of size linear in the size of the regular expression R. ρ describes the Parikh image of R. The description of the Parikh image in terms of linear-size existential Presburger arithmetic formulas is based on a result from [33].

Definition 4.1 (Parikh Image)**.**
The Parikh image of a symbolic regular expression R using propositional letters $L_1, \ldots, L_m$ over the variables $S_1, \ldots, S_k$ is the set

$$\mathsf{Parikh}(R) = \{(|\bar{s}|_{L_1}, \ldots, |\bar{s}|_{L_m}) \mid \bar{s} \in M_S(L_1, \ldots, L_m)\}$$

where $\bar{s}$ is a word of propositional formulas and $|\bar{s}|_{L_i}$ is the number of occurrences of L_i in the symbolic table $\bar{s}$.

Example 4.1. Continuing Example 2.3, we had a symbolic regular expression $S_1(S_1 \vee S_2)^* S_3$ and a word of propositional formulas $S_1(S_1 \vee S_2)(S_1 \vee S_2)(S_1 \vee S_2)S_3$. Thus, one vector in the Parikh image is $(1, 3, 1)$. In general, the Parikh image of $S_1(S_1 \vee S_2)^* S_3$ would contain the vectors $(1, k, 1)$ for each $k \in \mathbb{N}$.

Continuing with the description of the third part of the formula, F is the QFBAPA term in Formula 6, $p_\beta := \cap_{i=1}^{k} S_i^{\beta(i)}$ where $\beta \in \{0,1\}^k$, $p_L := \cup_{\beta \vDash L} p_\beta$ where $\vDash$ is the propositional satisfaction relation and $\beta \in \{0,1\}^k$, $|\cdot|$ denotes the cardinality of the argument set expression, $\dot\cup_{i \in I} S_i$ is the set $\cup_{i \in I} S_i$ where we emphasize that each pair of sets S_i, S_j are disjoint.

Shortening tuples of variables with an overline, we write Formula (8) as

$$\exists N \leq p(|F|), \exists s \in [m], \sigma : [s] \hookrightarrow [m], \exists \overline{\beta} \in \{0,1\}^k . C(\overline{\beta}) \wedge \exists \overline{c}, \overline{P} . H(\overline{c}, \overline{P}, \overline{\beta}) \tag{8}$$

This formula can be computed in polynomial time.

Intuitively, the partition variables P_i determine which propositional formula generated each value of the arrays accepted, so that, even if these formulas overlap, a model corresponding to a run of the automaton can be rebuilt. The reason why we need to check the existence of only one witness per elementary Venn region follows from the fact that we can "replicate" this witness in each of the indices that satisfied the corresponding formula ϕ^β (see also [30, 31]).

4.2. Proof of the theorem

We prove that Formula (6) and Formula (13) are equivalent.

$\Rightarrow$) If Formula (7) is true, then there are sets $S_1, \ldots, S_k$, a finite array a and a table $t \in T(R)$ such that

$$F(S_1, \ldots, S_k) \wedge \bigwedge_{i=1}^{k} S_i = \{\, n \in \mathbb{N} \mid \phi_i(\overline{a}(n), \overline{c}) \,\} = \{\, n \in \mathbb{N} \mid t_i(n) = 1 \,\} \tag{9}$$

Let $\bar{s}$ be the symbolic table corresponding to t, that is, the table made of the propositional formulas generated by $R(S_1, \ldots, S_k)$ such that t satisfies $\bar{s}$.

Define $c_i := |\bar{s}|_{L_i}$ for $i \in \{1, \ldots, m\}$ as the number of occurrences of L_i in the symbolic table $\bar{s}$, $s = |\{\, i \mid c_i \neq 0 \,\}|$, σ mapping the indices in $[s]$ to the indices i of the terms for which c_i is non-zero and $P_i = \left\{\, n \in \mathbb{N} \;\middle|\; \bar{s}(n) = L_{\sigma(i)} \,\right\}$. From the equalities $S_i = \{\, n \in \mathbb{N} \mid t_i(n) = 1 \,\} = \{\, n \in \mathbb{N} \mid \phi_i(\overline{a}(n), \overline{c}) \,\}$ in (9), we can show that the following holds

$$\begin{aligned} &\rho(c_1, \ldots, c_m) \wedge F(S_1, \ldots, S_k) \wedge \bigwedge_{i=1}^{s} P_i \subseteq p_{L_{\sigma(i)}} \wedge \\ &\bigwedge_{i=1}^{s} |P_i| = c_{\sigma(i)} \wedge \cup_{i=1}^{k} S_i = \cup_{i=1}^{m} p_{L_i} = \dot\cup_{i=1}^{s} P_i \end{aligned} \tag{10}$$

Formula (10) is in QFBAPA. Following the procedure from [25], we eliminate Boolean algebra expressions and the cardinality operator yielding a system of equations of the form

$$\exists k_1, \ldots, k_p . G \wedge \sum_{j=0}^{2^k - 1} \begin{pmatrix} [\![b_0]\!]_{\beta_j} \\ \vdots \\ [\![b_p]\!]_{\beta_j} \end{pmatrix} \cdot l_{\beta_j} = \begin{pmatrix} k_1 \\ \vdots \\ k_p \end{pmatrix} \tag{11}$$

where G is the existential Presburger arithmetic formula that results from (10) after the elimination and $l_{\beta_j} = |p_{\beta_j}|$.

We remove from the sum those terms corresponding to elementary Venn regions β such that $l_\beta = 0$. This includes regions whose associated formula in the interpreted Boolean algebra $\phi^\beta(a, \bar{c})$ is unsatisfiable,

and regions corresponding to bit-strings not occurring in t. This transformation leaves a reduced set of indices $\mathscr{R}$ participating in the sum:

$$\exists k_1, \ldots, k_p.G \wedge \sum_{\beta \in \{\beta_1, \ldots, \beta_N\} \subseteq \mathscr{R}} \begin{pmatrix} [\![b_0]\!]_{\beta_j} \\ \vdots \\ [\![b_p]\!]_{\beta_j} \end{pmatrix} \cdot l_{\beta_j} = \begin{pmatrix} k_1 \\ \vdots \\ k_p \end{pmatrix} \tag{12}$$

We now give the key auxiliary result from [25] proved in [11].

Definition 4.2. Given a subset $S \subseteq \mathbb{R}^n$, the integer conic hull of S is the set

$$int_{cone}(S) = \left\{ \sum_{i=1}^{t} \lambda_i s_i \Big| t \geq 0, s_i \in S, \lambda_i \in \mathbb{N} \right\}$$

Theorem 4.2. Let $X \subseteq \mathbb{Z}^n$ be a finite set of integer vectors, $b \in int_{cone}(X)$ and $M = \max_{\mathbf{x} \in X} \|\mathbf{x}\|_\infty = \max_{\mathbf{x} \in X} \max\{|x_1|, \ldots, |x_n|\}$ where $|x|$ denotes the absolute value of the number x. There exists a subset $X' \subseteq X$ such that $b \in int_{cone}(X')$ and $|X'| \leq 2n \log_2(4nM)$ where $|X'|$ denotes the cardinality of the set X'.

Using Theorem 4.2, there is a polynomial family of Venn regions $\beta_1, \ldots, \beta_N$ and corresponding cardinalities $l'_{\beta_1}, \ldots, l'_{\beta_N}$ such that:

$$\exists k_1, \ldots, k_p.G \wedge \sum_{\beta \in \{\beta_1, \ldots, \beta_N\} \subseteq \mathscr{R}} \begin{pmatrix} [\![b_0]\!]_{\beta_j} \\ \vdots \\ [\![b_p]\!]_{\beta_j} \end{pmatrix} \cdot l'_{\beta_j} = \begin{pmatrix} k_1 \\ \vdots \\ k_p \end{pmatrix} \tag{13}$$

We can assume that each cardinality variable l'_β is non-zero, since otherwise, we can remove it from the sum. With this assumption, $l'_{\beta_1}, \ldots, l'_{\beta_N}$ lists the cardinalities of a model of (10) which defines the cardinality of the elementary Venn region $S_1'^{\beta_1} \cap \ldots \cap S_k'^{\beta_k}$ to be equal to l'_β if $\beta \in \{\beta_1, \ldots, \beta_N\}$ and zero otherwise. In particular, we have that the following formula, corresponding to the subformula H in (8), holds

$$\begin{gathered} \rho(c_1, \ldots, c_m) \wedge F(S'_1, \ldots, S'_k) \wedge \bigwedge_{i=1}^{k} P'_i \subseteq p'_{L_{\sigma(i)}} \wedge \\ \bigwedge_{i=1}^{s} |P'_i| = c_{\sigma(i)} \wedge \cup_{i=1}^{k} S'_i = \cup_{i=1}^{m} p'_{L_i} = \cup_{i=1}^{N} p_{\beta_i} = \dot{\cup}_{i=1}^{s} P'_i \end{gathered} \tag{14}$$

For each $\beta \in \{\beta_1, \ldots, \beta_N\}$, the formula $\exists e.\phi^\beta(e, \bar{c})$ is true, since $l'_\beta > 0$. Thus, the subformula C in (8) is also true.

$\Leftarrow$) If Formula (8) is true, then there is a natural number $N \leq p(|F|)$ where p is a polynomial, $s \in [m]$, $\beta_1, \ldots, \beta_N \in \{0,1\}^k$, $c_1, \ldots, c_m \in \mathbb{N}$ and sets $S_1, \ldots, S_k, P_1, \ldots, P_s$ such that

$$\begin{gathered} \exists \bar{c}. \bigwedge_{j=1}^{N} \exists e.\phi^{\beta_j}(e, \bar{c}) \wedge \rho(c_1, \ldots, c_m) \wedge F(S_1, \ldots, S_k) \wedge \bigwedge_{i=1}^{s} P_i \subseteq p_{L_{\sigma(i)}} \wedge \\ \wedge \bigwedge_{i=1}^{s} |P_i| = c_{\sigma(i)} \wedge \cup_{i=1}^{k} S_i = \cup_{i=1}^{m} p_{L_i} = \dot{\cup}_{i=1}^{s} P_i = \cup_{i=1}^{N} p_{\beta_i} \end{gathered} \tag{15}$$

From (15) and the definition of ρ, follows that there is a symbolic table $\bar{s}$ generated by the symbolic regular expression R such that $|\bar{s}|_{L_i} = c_i$ for each $L_i \in \{L_1, \ldots, L_m\}$ occurring in $\bar{s}$. Moreover, from

$$\bigwedge_{i=1}^{s} P_i \subseteq p_{L_{\sigma(i)}} \wedge \cup_{i=1}^{m} p_{L_i} = \dot{\cup}_{i=1}^{s} P_i$$

we have that all the sets p_{L_i} are empty except $p_{L_{\sigma(1)}}, \ldots, p_{L_{\sigma(s)}}$.

From the subformula

$$\cup_{i=1}^{m} p_{L_i} = \dot{\cup}_{i=1}^{s} P_i \wedge \bigwedge_{i=1}^{s} P_i \subseteq p_{L_{\sigma(i)}} \wedge \bigwedge_{i=1}^{s} |P_i| = c_{\sigma(i)}$$

follows that we may define P_i to consist of the indices of $\bar{s}$ labelled with the formula $L_{\sigma(i)}$ for each $i \in \{1, \ldots, s\}$. This is because the values in P_i are guaranteed to satisfy the formula $L_{\sigma(i)}$. Note that the values in P_i could satisfy other formulas $L_{\sigma(j)}$. However, the role of the variables P_i is to determine which formula of the regular expression generated the values satisfying $L_{\sigma(i)}$ regardless of whether the witnesses satisfied other formulas too.

From the subformula

$$\exists \bar{c}. \bigwedge_{j=1}^{N} \exists e. \phi^{\beta_j}(e, \bar{c}) \wedge \dot{\cup}_{i=1}^{s} P_i = \cup_{i=1}^{N} p_{\beta_i}$$

it follows that there exist values $e_{\beta_1}, \ldots, e_{\beta_N}$ satisfying the formulas $\phi^{\beta_1}(e, \bar{c}), \ldots, \phi^{\beta_N}(e, \bar{c})$ and moreover, all the elements in p_{β_j} belong to a single set P_i.

We define a table t by substituting in $\bar{s}$ the indices in P_i by the values β_j such that $p_{\beta_j} \subseteq P_i$. Similarly, we build an array a by substituting in $\bar{s}$ the indices in P_i by the values e_{β_j} such that $p_{\beta_j} \subseteq P_i$.

Observe that $F(S_1, \ldots, S_k)$ holds by assumption. Moreover,

$$S_j = \cup_{\{i \mid \beta_i(j)=1\}} p_{\beta_i} = \{\, n \in \mathbb{N} \mid t_j(n) = 1 \,\}$$
$$S_j = \cup_{\{i \mid \beta_i(j)=1\}} p_{\beta_i} = \{\, n \in \mathbb{N} \mid \phi_j(\bar{a}(n), \bar{c}) \,\}$$

Thus, Formula (7) is true. □

4.3. Computational complexity of the combination

Note that Theorem 4.1 also allows to assert at once the complexity of the underlying logical theory.

Corollary 4.1. Let $\mathscr{C}$ be the complexity class to which the satisfiability problem of $Th_{\exists^*}(\mathscr{D})$ belongs. If $\mathscr{C} = \mathrm{P}$ then the satisfiability problem of formulas of the form (6) is in NP. If $\mathscr{C} \supseteq \mathrm{NP}$ then the satisfiability problem of formulas of the form (6) is in $\mathscr{C}$.

5. The case of trees

One can adapt the techniques of Section 4 to the case when the regular specification is given by a parametric tree automaton, thus extending the results of [18]. The main difference with the procedure of Section 4 is that in the case of parametric tree automata one needs to compute the Parikh image of a regular tree language. This is done in a completely analogous way as it is done in Definition 4.1 for parametric finite automata. Note that it is easy to convert from non-deterministic top-down to non-deterministic bottom-up tree automata [8, Theorem 1.6.1]. One can then use the observation of Klaedtke and Rueß [22, Lemma 17] which allows to reduce the problem to the computation of the Parikh image of a context-free grammar. Finally, [36] says that the Parikh image of a context-free grammar can be described by a linear-sized existential Presburger arithmetic formula.

6. Conclusion

We have shown how to extend decision procedures for satisfiability of parametric array fragments with regular constraints. In terms of quantifiers, this shows how to simultaneously support Härtig's quantifiers and WS1S second-order quantification. Our techniques extend previous results of Alberti, Ghilardi and Pagani [1] since the relations expressible in WS1S extend those expressible in Presburger

arithmetic. They also extend recent results in the literature [18], which did not handle the cardinality operator.

Our work mixes ideas from decision algorithms for three different logical theories: quantifier-free BAPA [25], combinations of BAPA with WS1S and WS2S [37] and existential fragment of power structures [30]. However, a crucial technical difficulty has been overcome to make the combination work. This difficulty stems from the fact that while Büchi's automata are over finite alphabets, the corresponding automata (or equivalently, regular expressions) that appear in the Feferman-Vaught framework use first-order formulas in their transitions, since set variables are interpreted. We demonstrated, as our colleagues [18], that one can still use the Parikh image of this symbolic automata to combine theories. Since one is now counting formulas rather than symbols from a finite alphabet, and formulas can overlap in the semantic domain, it becomes necessary to indicate to the QFBAPA constraint which transition of the automaton produced each index. We achieved this by introducing a partition of the sets of indices of the array, where each part corresponds to the indices generated by a given transition.

The implementation of the algorithm could be achieved mixing the techniques of ARCA-SAT [1] with software computing the Parikh image of regular expressions and context-free grammars. For the latter, there exist several implementations, we mention for instance [19], where a fix to the construction original of Verma et alii [36], is also described.

Possible applications of the decision procedure include automatic verification of array manipulating programs in deductive verification systems and model checking of distributed protocols. Nevertheless, due to the number of ideas that are combined in this work, we would not be surprised if further applications are found in the future.

Acknowledgements

The author wishes to express his gratitude to Viktor Kunčak for suggesting us the application of the methods of our thesis to symbolic automata. Anthony Lin and Oliver Markgraf provided useful remarks on the final presentation of the results in this paper. Research supported by the Swiss NSF Project P500PT_222338

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] J. Barwise and S. Feferman, editors. *Model-Theoretic Logics*. Perspectives in Logic. Cambridge University Press, Cambridge, 2017. doi:10.1017/9781316717158.

[3] Roderick Bloem, Ayrat Khalimov, Swen Jacobs, Igor Konnov, Helmut Veith, Josef Widder, and Sasha Rubin. *Decidability of Parameterized Verification*. Synthesis Lectures on Distributed Computing Theory. Springer International Publishing, Cham, 2015. doi:10.1007/978-3-031-02011-7.

[4] Mikołaj Bojańczyk, Luc Segoufin, and Szymon Toruńczyk. Verification of database-driven systems via amalgamation. In *Proceedings of the 32nd ACM SIGMOD-SIGACT-SIGAI symposium on Principles of database systems*, PODS '13, pages 63–74, New York, NY, USA, June 2013. Association for Computing Machinery. doi:10.1145/2463664.2465228.

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

[6] Aaron R Bradley. *Safety analysis of systems*. PhD thesis, Stanford University, May 2007.

[7] J. Richard Büchi. Weak Second-Order Arithmetic and Finite Automata. *Mathematical Logic Quarterly*, 6(1-6):66–92, 1960. doi:10.1002/malq.19600060105.

[8] Hubert Comon, Max Dauchet, Remi Gilleron, Florent Jacquemard, Denis Lugiez, Christof Loding, Sophie Tison, and Marc Tommasi. *Tree Automata Techniques and Applications*. INRIA, 2008.

[9] 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.

[10] 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.

[11] Friedrich Eisenbrand and Gennady Shmonin. Carathéodory bounds for integer cones. *Operations Research Letters*, 34(5):564–568, September 2006. doi:10.1016/j.orl.2005.09.008.

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

[13] Paolo Felli, Alessandro Gianola, and Marco Montali. SMT-based Safety Checking of Parameterized Multi-Agent Systems. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(7):6321–6330, May 2021. Number: 7. URL: https://ojs.aaai.org/index.php/AAAI/article/view/16785, doi:10.1609/aaai.v35i7.16785.

[14] Klaus Freiherr von Gleissenthall. *Cardinalities in Software Verification*. PhD thesis, Technische Universität München, Fakultät für Informatik, 2016.

[15] 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. doi:10.1007/978-3-031-42746-6.

[16] 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.

[17] Arie Gurfinkel, Sharon Shoham, and Yuri Meshman. SMT-based verification of parameterized systems. In *Proceedings of the 2016 24th ACM SIGSOFT International Symposium on Foundations of Software Engineering*, FSE 2016, pages 338–348, New York, NY, USA, November 2016. Association for Computing Machinery. URL: https://dl.acm.org/doi/10.1145/2950290.2950330, doi:10.1145/2950290.2950330.

[18] Matthew Hague, Artur Jeż, and Anthony W. Lin. Parikh's Theorem Made Symbolic. *Proceedings of the ACM on Programming Languages*, 8(POPL):65:1945–65:1977, January 2024. URL: https://dl.acm.org/doi/10.1145/3632907, doi:10.1145/3632907.

[19] Matthew Hague and Anthony Widjaja Lin. Synchronisation- and Reversal-Bounded Analysis of Multithreaded Programs with Counters. In P. Madhusudan and Sanjit A. Seshia, editors, *Computer Aided Verification*, Lecture Notes in Computer Science, pages 260–276, Berlin, Heidelberg, 2012. Springer. doi:10.1007/978-3-642-31424-7_22.

[20] Chih-Duo Hong and Anthony W. Lin. Regular Abstractions for Array Systems. *Proceedings of the ACM on Programming Languages*, 8(POPL):638–666, January 2024. URL: https://dl.acm.org/doi/10.1145/3632864, doi:10.1145/3632864.

[21] James Cornelius King. *A program verifier*. PhD thesis, Carnegie-Mellon University, Pittsburgh Pennsylvania USA, September 1969. Section: Technical Reports. URL: https://apps.dtic.mil/sti/citations/AD0699248.

[22] Felix Klaedtke and Harald Rueß. Parikh Automata and Monadic Second-Order Logics with Linear Cardinality Constraints. Technical Report 177, Freiburg University, Institute of Computer Science, 2002.

[23] 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.

[24] Viktor Kunčak, Huu Hai Nguyen, and Martin Rinard. Deciding Boolean Algebra with Presburger Arithmetic. *Journal of Automated Reasoning*, 36(3):213–239, April 2006. doi:10.1007/s10817-006-9042-1.

[25] Viktor Kunčak and Martin Rinard. Towards Efficient Satisfiability Checking for Boolean Algebra with Presburger Arithmetic. In *Automated Deduction – CADE-21*, Lecture Notes in Computer

Science, pages 215–230, Berlin, Heidelberg, 2007. Springer. doi:10.1007/978-3-540-73595-3_15.

[26] Haojun Ma, Aman Goel, Jean-Baptiste Jeannin, Manos Kapritsos, Baris Kasikci, and Karem A. Sakallah. I4: incremental inference of inductive invariants for verification of distributed protocols. In *Proceedings of the 27th ACM Symposium on Operating Systems Principles*, pages 370–384, Huntsville Ontario Canada, October 2019. ACM. doi:10.1145/3341301.3359651.

[27] Makai Mann, Ahmed Irfan, Alberto Griggio, Oded Padon, and Clark Barrett. Counterexample-Guided Prophecy for Model Checking Modulo the Theory of Arrays. In Jan Friso Groote and Kim Guldstrand Larsen, editors, *Tools and Algorithms for the Construction and Analysis of Systems*, pages 113–132, Cham, 2021. Springer International Publishing. doi:10.1007/978-3-030-72016-2_7.

[28] Andrzej Mostowski. On direct products of theories. *The Journal of Symbolic Logic*, 17(1):1–31, March 1952. doi:10.2307/2267454.

[29] Rodrigo Raya. *Decision Procedures for Power Structures*. PhD thesis, EPFL, 2023. doi:10.5075/epfl-thesis-10546.

[30] Rodrigo Raya and Viktor Kunčak. NP Satisfiability for Arrays as Powers. In *23rd International Conference on 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.

[31] 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.

[32] Rodrigo Raya and Viktor Kunčak. Succinct ordering and aggregation constraints in algebraic array theories. *Journal of Logical and Algebraic Methods in Programming*, 140:100978, August 2024. doi:10.1016/j.jlamp.2024.100978.

[33] Helmut Seidl, Thomas Schwentick, Anca Muscholl, and Peter Habermehl. Counting in Trees for Free. In *Automata, Languages and Programming*, Lecture Notes in Computer Science, pages 1136–1149, Berlin, Heidelberg, 2004. Springer. doi:10.1007/978-3-540-27836-8_94.

[34] Larry Stockmeyer and Albert R. Meyer. Cosmological lower bound on the circuit complexity of a small problem in logic. *Journal of the ACM*, 49(6):753–784, November 2002. doi:10.1145/602220.602223.

[35] Wolfgang Thomas. Languages, Automata, and Logic. In *Handbook of Formal Languages*, pages 389–455. Springer Berlin Heidelberg, Berlin, Heidelberg, 1997. URL: http://link.springer.com/10.1007/978-3-642-59126-6_7, doi:10.1007/978-3-642-59126-6_7.

[36] Kumar Neeraj Verma, Helmut Seidl, and Thomas Schwentick. On the Complexity of Equational Horn Clauses. In *Automated Deduction – CADE-20*, volume 3632, pages 337–352, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg. doi:10.1007/11532231_25.

[37] Thomas Wies, Ruzica Piskac, and Viktor Kunčak. Combining Theories with Shared Set Operations. In *Frontiers of Combining Systems*, Lecture Notes in Computer Science, pages 366–382, Berlin, Heidelberg, 2009. Springer. doi:10.1007/978-3-642-04222-5_23.