# On Algebraic Array Theories 

Rodrigo Raya ${ }^{\text {a,* }}$, Viktor Kunčak ${ }^{\text {a }}$<br>${ }^{a}$ School of Computer and Communication Science<br>École Polytechnique Fédérale de Lausanne (EPFL), Switzerland


#### Abstract

Automatic verification of programs manipulating arrays relies on specialised decision procedures. A methodology to classify the theories handled by these procedures is introduced. It is based on decomposition theorems in the style of Feferman and Vaught. The method is applied to obtain an extension of combinatory array logic that is closed under propositional operations and Hoare triples. A classification according to expressiveness of six different fragments studied in the literature is given.


Keywords: decision procedures, satisfiability modulo theories, arrays, Feferman-Vaught, composition theorems

## 1. Introduction

The problem of specifying and verifying programs is nearly as old as the study of computer science itself. Turing [43] was first to describe the method of annotations as a way of proving the correctness of computer programs. Independently, McCarthy put forward the idea of proving the correctness of programs automatically. Floyd [19] synthesized both ideas in the verifying compiler: a computer program that would automatically check programs with annotations.

King [25] was the first to implement a verifying compiler. He observed that the verification of programs involving arrays was possibly the most serious weakness of the system and gave concrete examples where the naive simplification routines of his theorem prover yielded the verification of simple array programs unfeasible. He proposed to use the translation of array

[^0]programs to first-order logic proposed by McCarthy [32] and to design specialised decision procedures for the resulting theories.

A plethora of array theories and decision procedures have appeared since (see $[41,8,20,22,15,2,3,14,21,1,35]$ and the references therein).

However, modern verifying compilers [30, 42, 44, 45] mostly rely on the decision procedures implemented in industrial-strength satisfiability modulo theories (SMT) solvers [16, 5]. Both Z3 and the CVC family of SMT solvers only implement a fragment of the theory of arrays known as combinatory array logic (CAL) [15]. This theory extends the extensional theory of arrays with first-order symbols interpreted pointwise.

This paper studies combinatory array logic from an algebraic perspective as the set of sentences true in certain power structure. We characterise the expressivity of this fragment by decomposing the satisfiability problem of array formulae into the satisfiability problem of index theory formulae and the satisfiability problem of formulae in the theory of elements. This decomposition technique is reminiscent of a family of theorems in first-order model theory known as Feferman-Vaught theorems in honor of their originators [17].

As a result, we obtain a modular framework in which to express a family of array theories that have been studied in the literature [41, 15, 21, 1]. The insights provided by the decomposition technique allow us to extend these theories: we are able to express the relations on sets of indices definable in the weak second-order theory of one successor [11] while leaving unchanged the computational complexity of the underlying satisfiability problem.

## 2. Preliminaries

We recall next some concepts from first-order logic and computational complexity which make the paper self-contained.

### 2.1. First-order Logic

The formal theories we study are written using a first-order language.
Definition 1. A first-order language is one whose logical symbols are $\neg, \wedge, \vee, \forall$ and $\exists$, whose terms are either variables, constants or function symbols applied to terms and whose formulas are either atomic (relation symbols applied to terms) or general (atomic formulas and inductively, from formulas $A, B$, we get new formulae $\neg A, A \wedge B$ and $A \vee B$ and from a formula $A$ and $a$ variable symbol $x$ we get the new formulae $\exists x$. $A$ and $\forall x . A)$.

An occurrence of a variable in a formula is free if there is no occurrence of that variable under a quantifier on the path of the syntax tree of the formula reaching the occurrence of the variable. A formula without free variables is a sentence.

It is customary to study the computational properties of theories in terms of the quantifiers used. To this end one introduces the prenex form of formulae [37] which consists of a string of quantifiers (called the prefix of the formula) followed by a quantifier-free formula (known as the matrix of the formula) and proves that every first-order formula is equivalent to a formula in prenex normal form.

The computational properties of logical theories have been the object of intense research since the pioneering work described in [18]. This research showed that traditional decidable theories studied by logicians were strongly intractable in practice. For the sake of example, [40] proved that a circuit deciding the weak monadic second-order theory of one successor would not fit in the known universe.

The computational complexity usually increases with the number of quantifier alternations. For this reason, in applications one often focus on the existential fragment of theories: the subset of formulae of a theory whose prefix is made of existential quantifiers only. This is the case for the applications in program verification that are considered in this work [25, 7, 27].

More precisely, we will study existential fragments of theories defined semantically in terms of the models that the formulae in the theory satisfy.

Definition 2. A structure $\mathcal{A}$ is a tuple with four components: a set $A$ called the domain of $A$; a set of constant elements of $A$; for each positive integer $n$, a set of n-ary relations on $A$ (i.e. subsets of $A^{n}$ ), each of which is named by one or more n-ary relation symbols and for each positive integer $n$, a set of n-ary operations on $A$ (i.e maps from $A^{n}$ to $A$ ), each of which is named by one or more n-ary function symbols.

As announced, we will focus on the existential fragment of theories defined by some model.

Definition 3. The existential theory $T h_{\exists^{*}}(\mathcal{A})$ of a structure $\mathcal{A}$, is the set of existentially quantified formulas $\psi$ of $L$ such that $\mathcal{A} \models \psi$. A solution of the formula $\psi$ is a satisfying assignment to its existential variables.

Because we focus on existential fragments, the computational problems we study are similar to problems in predicate clause logic or propositional
logic [9]. Strictly speaking when determining membership in $T h_{\exists^{*}}(\mathcal{A})$ we are studying the validity problem for the existential fragment, while we are solving the satisfiability problem if we consider the same fragment as part of predicate clause logic.

More importantly, one can define normal forms for the matrix of the formulae in the studied fragment. The familiar notion of disjunctive normal form is used in the decision procedures below to reduce the satisfiability problem of the existential fragment to the satisfiability problem of a conjunction of literals. The notion of Stone normal form [24] is implicitly used when studying the corresponding theories with the cardinality operator on index sets. These are defined in a completely analogous way to the propositional case by forgetting the additional first-order structure.

### 2.2. Computational Complexity

We assume basic definitions in the theory of computation [4, 38] such as NP-hardness and NP-completeness. We will use the notion of polynomialtime verifier which is equivalent to that of non-deterministic polynomialtime procedure with the difference that the non-deterministic computation is encoded as a certificate.

Definition 4. A language $L \subseteq\{0,1\}^{*}$ is in $N P$ if there exists a polynomial $p: \mathbb{N} \rightarrow \mathbb{N}$ and a polynomial-time Turing machine $V$, called the verifier for $L$ such that for every $x \in\{0,1\}^{*}, x \in L$ if and only if there exists $C \in\{0,1\}^{p(|x|)}$ such that $V(x, C)=1$. If $x \in L$ and $u \in\{0,1\}^{p(|x|)}$ satisfy $V(x, C)=1$, then $C$ is called a certificate for $x$.

It is straightforward to generalise the notion of polynomial-time verifier so that it outputs a bit-string rather than a single bit. We use this notion to define NP-reductions which we use to formalise the idea that it suffices to solve formulae of the existential fragment that are conjunctions of literals.

Definition 5. A language $L \subseteq\{0,1\}^{*}$ is $N P$-reducible to a language $L^{\prime} \subseteq$ $\{0,1\}^{*}$, written $L \leq_{n p} L^{\prime}$, if there is a polynomial-time verifier $V$ such that for every $x \in\{0,1\}^{*}, x \in L$ if and only if there exists a certificate $C$ such that $V(x, C) \in L^{\prime}$.

Lemma 6. The relation $\leq_{n p}$ satisfies the following properties:

1. If $L \leq_{n p} L^{\prime}$ and $L^{\prime} \in N P$ then $L \in N P$.
2. If $L \leq_{n p} L^{\prime}$ and $L^{\prime} \leq_{n p} L^{\prime \prime}$ then $L \leq_{n p} L^{\prime \prime}$.

Proof. Ad 1), by hypothesis, we have a verifier $V$ satisfying Definition 5 for $L \leq_{n p} L^{\prime}$. We also have a verifier $V^{\prime}$ accepting $L^{\prime}$. Then $V^{\prime \prime}=V^{\prime} \circ V$ is a verifier for $L$. Ad 2), if $V$ satisfies Definition 5 for $L \leq_{n p} L^{\prime}$ and $V^{\prime}$ satisfies Definition 5 for $L^{\prime} \leq_{n p} L^{\prime \prime}$ then $V^{\prime} \circ V$ satisfies Definition 5 for $L \leq_{n p} L^{\prime \prime}$.

The following lemma allows our decision procedures to guess with a disjunct from the disjunctive normal form of the existential formula given as input.

Lemma 7. Let $\psi$ 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 $\psi$ 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 decision procedure for this problem will start by guessing a disjunct of the DNF form of the input formula. To ensure this can be done by a polynomial-time verifier we need to ensure that the size of that disjunct is polynomial in the size of the input formula.

Lemma 8. There is $N P$-reduction from the satisfiable formulae of the existential fragment of a theory to the satisfiable formulae of the existential fragment of the theory in conjunctive form.

Proof. We define a verifier $V$ that given an input $x$ and a certificate $w$, interprets the certificate as a disjunct $\varphi$ of the DNF of $x$ and checks that it is so. To check that the guessed disjunct is part of some DNF the verifier only needs to check the formula $\operatorname{prop}(\varphi) \Longrightarrow \operatorname{prop}(x)$, where prop is a function that transforms the original formula into propositional one syntactically: two atomic formulae are mapped to the same variable if and only if they coincide syntactically.
$V$ is a polynomial-time verifier. Indeed, if $x$ is satisfiable then there must exist a satisfiable disjunct in its DNF form. By Lemma 7, this disjunct of the DNF form can be written in linear space. Thus, there exists a certificate $C$ of polynomial-size encoding such disjunct. Thus, the output of $V(x, C)$ is a satisfiable formula in conjunctive form. Conversely, if the output of $V(x, C)$ is a satisfiabile formula in conjunctive form, by the check that $V$ performs,
it follows that $V(x, C)$ is a disjunct of some DNF form of $x$. It follows that $x$ itself was also satisfiable.

## 3. Power Structures

### 3.1. Definition of Power Structures

Given a first-order language $L$, a non-empty set $I$ and a structure $\mathcal{M}$ with carrier $M$, a power structure is a special case of product structure [23, Chapter 9] were all components are equal.

Definition 9. The power structure $\Pi$ associated to $L, I$ and $\mathcal{M}$ 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 $\left(a_{1}, \ldots, a_{n}\right) \in\left(M^{I}\right)^{n}$,

$$
f^{\Pi}\left(a_{1}, \ldots, a_{n}\right)(i)=f^{\mathcal{M}}\left(a_{1}(i), \ldots, a_{n}(i)\right)
$$

For each relation symbol $R, n \in \mathbb{N}$ and $\left(a_{1}, \ldots, a_{n}\right) \in\left(M^{I}\right)^{n}$,

$$
\left(a_{1}, \ldots, a_{n}\right) \in R^{\Pi} \text { if and only if for every } i \in I,\left(a_{1}(i), \ldots, a_{n}(i)\right) \in R^{\mathcal{M}}
$$

Notation 10. We abbreviate the tuple $\left(a_{1}, \ldots, a_{n}\right) \in\left(M^{I}\right)^{n}$ by $\bar{a}$ and the tuple $\left(a_{1}(i), \ldots, a_{n}(i)\right)$ by $\bar{a}(i)$.

The semantics of the language used for power structures is thus particularly simple: constants, functions and relations are interpreted pointwise. In spite of its simplicity, the existential fragment of the theory of a power structure is precisely the logical fragment that [15] automates under the name of combinatory array logic and which is implemented in the widely used Z3 and CVC family of SMT solvers. In the following section, we give a simple proof of the complexity of the decision problem of this fragment. It is the base of posterior developments.

### 3.2. Complexity of the Existential Fragment of Power Structures

In this section, we study the complexity of the satisfiability problem for the set $T h_{\exists *}(\Pi)$. We show that the satisfiability problem decomposes into that of the theory $T h_{\exists *}(\mathcal{M})$ and that of the theory $T h_{\exists *}(\langle I\rangle)$. The latter is trivial since it contains no sentences.

Theorem 11. $T h_{\exists^{*}}(\mathcal{M}) \in N P$ if and only if $T h_{\exists^{*}}(\Pi) \in N P$.

Proof. 1) Assume that $V_{C}$ is a polynomial time verifier for $T h_{\exists^{*}}(\mathcal{M})$. Figure 1 gives a polynomial time verifier $V$ for $T h_{\exists^{*}}(\Pi)$. 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. $t_{j}^{i}$ 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 $T h_{\exists *}(\Pi)$ :

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

1. Take $w$ and interpret it as

- A partition $P=\left\{p_{1}, \ldots, p_{t}\right\}$ of $\{l+1, \ldots, k\}$.
- Certificates $C_{0}, \ldots, C_{t}$ for $V_{C}$ on inputs

$$
\begin{aligned}
\varphi_{0} & \equiv \exists x_{1}, \ldots, x_{n} \cdot \wedge_{i=1}^{l} R_{i}\left(t_{1}^{i}, \ldots, t_{a_{i}}^{i}\right) \\
\varphi_{d} & \equiv \exists x_{1}, \ldots, x_{n} \cdot \wedge_{i=1}^{l} R_{i}\left(t_{1}^{i}, \ldots, t_{a_{i}}^{i}\right) \wedge \wedge_{e \in p_{d}}^{\wedge} \neg R_{e}\left(t_{1}^{e}, \ldots, t_{a_{e}}^{e}\right)
\end{aligned}
$$

for each $p_{d} \in P$.
where the input $x$ is a term in conjunctive form

$$
\varphi \equiv \exists x_{1}, \ldots, x_{n} \cdot \wedge_{i=1}^{l} R_{i}\left(t_{1}^{i}, \ldots, t_{a_{i}}^{i}\right) \wedge \bigwedge_{j=l+1}^{k} \neg R_{j}\left(t_{1}^{j}, \ldots, t_{a_{j}}^{j}\right)
$$

2. If $t>|I|$ then reject.
3. Otherwise, run $V_{C}$ with $\left\langle\varphi_{d}, C_{d}\right\rangle$ for $d=0, \ldots, t$.
4. Accept iff all runs accept.

Figure 1: Verifier for $T h_{\exists^{*}}(\Pi)$

- $w$ has polynomial size in $|x|$ :
- By Lemma 7, $|\varphi|=O(|x|)$.
- Thus, $k=O(|x|)$.
- $P=O\left(|x|^{2}\right)$ since $P$ can be written with $k \log (k)+k$ bits.
- Since $\left|C_{d}\right|=O\left(\left|\varphi_{d}\right|^{c_{d}}\right)$ and $\left|\varphi_{d}\right| \leq|\varphi|=O(|x|),\left|C_{d}\right|=O\left(|x|^{c_{d}}\right)$.
- Thus, $|w|=|\varphi|+|P|+\sum_{d=0, \ldots, t}\left|C_{d}\right|=O\left(\left.|x|\right|_{d} ^{\max \left\{2, \max _{d} c_{d}\right\}}\right)$.
- $V$ runs in polynomial time in $|x|$.
- Building the list of $\varphi_{d}$ is $O\left(|x|^{2}\right)$.
- As above, $\left|\varphi_{d}\right| \leq|\varphi|=O(|x|)$.
- So each call to $V_{C}$ runs in $O\left(|x|^{f}\right)$ ( $V_{C}$ is polynomial time).
- Like before, $k=O(|x|)$.
- Therefore, $V$ runs in $O\left(|x|^{\max \{2, f+1\}}\right)$.
- $V$ is a verifier for $T h_{\exists^{*}}(\Pi)$.
$\Rightarrow)$ If $x \in T h_{\exists *}(\Pi)$ then writing $x$ in prenex DNF form, there is at least one disjunct $\varphi$ (as in figure 1) true in the product. Thus, there is $\bar{s} \in M^{I}$ satisfying

$$
\begin{aligned}
& \left(t_{1}^{i \Pi}[\bar{x} \mapsto \bar{s}], \ldots, t_{a_{i}}^{i}{ }^{\Pi}[\bar{x} \mapsto \bar{s}]\right) \in R_{i}^{\Pi} \\
& \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \\
& \left(t_{1}^{j \Pi}[\bar{x} \mapsto \bar{s}], \ldots, t_{a_{j}}^{j{ }^{\Pi}}[\bar{x} \mapsto \bar{s}]\right) \notin R_{j}^{\Pi}
\end{aligned}
$$

Using the semantics of products this means

$$
\begin{aligned}
& \forall r \in I .\left(t_{1}^{i \mathcal{M}}[\bar{x} \mapsto \bar{s}(r)], \ldots, t_{a_{i}}^{i}{ }^{\mathcal{M}}[\bar{x} \mapsto \bar{s}(r)]\right) \in R_{i}^{\mathcal{M}} \\
& \exists r \in I .\left(t_{1}^{j \mathcal{M}}[\bar{x} \mapsto \bar{s}(r)], \ldots, t_{a_{j}}^{j}{ }^{\mathcal{M}}[\bar{x} \mapsto \bar{s}(r)]\right) \notin R_{j}^{\mathcal{M}}
\end{aligned}
$$

So there is a map $r:\{l+1, \ldots, k\} \rightarrow I$ that assigns to each formula, one index where it holds. $r$ induces a partition $P=r^{-1}(I)$ of $\{l+1, \ldots, k\}$
with $t=|P| \leq \min (|I|, k-l)$. Each part $p_{d}=\left\{e_{1}, \ldots, e_{m}\right\}$ and each associated index $r_{d}=r\left(e_{i}\right)$, satisfy the following system

$$
\begin{aligned}
& \left(t_{1}^{i \mathcal{M}}\left[\bar{x} \mapsto \bar{s}\left(r_{d}\right)\right], \ldots, t_{a_{i}}^{i}{ }^{\mathcal{M}}\left[\bar{x} \mapsto \bar{s}\left(r_{d}\right)\right]\right) \in R_{i}^{\mathcal{M}} \\
& \left(t_{1}^{e_{1} \mathcal{M}}\left[\bar{x} \mapsto \bar{s}\left(r_{d}\right)\right], \ldots, t_{a_{e_{1}}}^{e_{1}} \mathcal{M}\left[\bar{x} \mapsto \bar{s}\left(r_{d}\right)\right]\right) \notin R_{e_{1}}^{\mathcal{M}} \\
& \left(t_{1}^{e_{m} \mathcal{M}}\left[\bar{x} \mapsto \bar{s}\left(r_{d}\right)\right], \ldots, t_{a_{e_{m}}}^{e_{m}} \mathcal{M}\left[\bar{x} \mapsto \bar{s}\left(r_{d}\right)\right]\right) \notin R_{e_{m}}^{\mathcal{M}}
\end{aligned}
$$

Equivalently, for each $d \in\{1, \ldots, t\}, \mathcal{M} \models \varphi_{d}\left[\bar{x} \mapsto \bar{s}\left(r_{d}\right)\right]$. For $d=0$, we set

$$
r_{0}= \begin{cases}\text { any index } i \in I & \text { if } t=0 \\ \text { some } r_{d} \in\left\{r_{1}, \ldots, r_{t}\right\} & \text { if } t>0\end{cases}
$$

Then $\mathcal{M} \models \varphi_{0}\left[\bar{x} \mapsto \bar{s}\left(r_{0}\right)\right]$. By definition of $V_{C}$, there are polynomiallysized certificates $C_{0}, \ldots, C_{t}$ such that $V_{C}$ accepts $\left\langle\varphi_{d}, C_{d}\right\rangle$ for each $d$. Thus $V$ accepts $\left\langle x,\left\langle\varphi, P, C_{0}, \ldots, C_{t}\right\rangle\right\rangle$.
$\Leftarrow)$ Let $w=\left\langle\varphi, P,\left\{C_{d}\right\}_{d \in\{0, \ldots, t\}}\right\rangle$ 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 $\left\langle\varphi_{d}, C_{d}\right\rangle$, i.e. $\mathcal{M} \models \varphi_{d}$. So there are solutions $x_{\cdot i}=\left(x_{1 i}, \ldots, x_{n i}\right)^{t}$ to the formulas

$$
\begin{aligned}
\varphi_{0} & \equiv \exists x_{10}, \ldots, \exists x_{n 0} \cdot{ }_{i=1}^{l} R_{i}\left(t_{1}^{i}, \ldots, t_{a_{i}}^{i}\right) \\
\varphi_{d} & \equiv \exists x_{1 d}, \ldots, \exists x_{n d} \cdot{ }_{i=1}^{l} R_{i}\left(t_{1}^{i}, \ldots, t_{s_{i}}^{i}\right) \wedge \wedge_{e \in p_{d}}^{\wedge} \neg R_{e}\left(t_{1}^{e}, \ldots, t_{a_{e}}^{e}\right)
\end{aligned}
$$

Fix distinct $i_{1}, \ldots, i_{t} \in I$. Consider the $n \times|I|$ matrix with entries

$$
s_{j i}= \begin{cases}x_{j i} & \text { if } i \in\left\{i_{1}, \ldots, i_{t}\right\} \\ x_{j 0} & \text { otherwise }\end{cases}
$$

We show that the rows of this matrix $\bar{s}=\left\{s_{1}, \ldots, s_{n}\right\}$ are solutions of $\varphi$ in the product structure, i.e.

$$
\begin{gathered}
\left(t_{1}^{i \Pi}[\bar{x} \mapsto \bar{s}], \ldots, t_{a_{i}}^{i}{ }^{\Pi}[\bar{x} \mapsto \bar{s}]\right) \in R_{i}^{\Pi} \\
\ldots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \\
\left(t_{1}^{j \Pi}[\bar{x} \mapsto \bar{s}], \ldots, t_{a_{j}}^{j}{ }^{\Pi}[\bar{x} \mapsto \bar{s}]\right) \notin R_{j}^{\Pi}
\end{gathered}
$$

Using the definition of product, it is sufficient to show

$$
\begin{aligned}
& \forall r \in I .\left(t_{1}^{i \mathcal{M}}[\bar{x} \mapsto \bar{s}(r)], \ldots, t_{a_{i}}^{i}{ }^{\mathcal{M}}[\bar{x} \mapsto \bar{s}(r)]\right) \in R_{i}^{\mathcal{M}} \\
& \exists r \in I .\left(t_{1}^{j \mathcal{M}}[\bar{x} \mapsto \bar{s}(r)], \ldots, t_{a_{j}}^{j}{ }^{\mathcal{M}}[\bar{x} \mapsto \bar{s}(r)]\right) \notin R_{j}^{\mathcal{M}}
\end{aligned}
$$

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

$$
\left(t_{1}^{i \mathcal{M}}[\bar{x} \mapsto \bar{s}(r)], \ldots, t_{s_{i}}^{i} \mathcal{M}[\bar{x} \mapsto \bar{s}(r)]\right) \in R_{i}^{\mathcal{M}}
$$

If $r \in\left\{i_{1}, \ldots, i_{t}\right\}$ then $\bar{s}(r)=x_{\text {.r }}$ (i.e. all $\left.x_{1 r}, \ldots, x_{n r}\right)$ and the equation holds since $\mathcal{M} \models \varphi_{r}\left[x_{. r}\right]$. Otherwise, $\bar{s}(r)=x_{.0}$ and the equation holds since $\mathcal{M} \models \varphi_{0}\left[x_{.0}\right]$.
For $j \in\{l+1, \ldots, k\}$ and some $r \in I$, the following formula needs to hold

$$
\left(t_{1}^{j \mathcal{M}}[\bar{x} \mapsto s(r)], \ldots, t_{s_{j}}^{j} \mathcal{M}^{\prime}[\bar{x} \mapsto s(r)]\right) \notin R_{j}^{\mathcal{M}}
$$

We take $r=i_{d}$ such that $j \in p_{d}$. Then $s(r)=x_{\cdot r}$ and the equation holds since $\mathcal{M} \vDash \varphi_{r}\left[x_{r}\right]$.
2) The converse direction is trivial. For details see [35].

## 4. The Cardinality Operator

The technique of Section 3 is closer to the methods of Mostowski [34]. He gave a decision procedure to reduce the (full) first-order theory of a power structure to the first-order theory of the elements and the first-order theory of the indices. Feferman and Vaught [17] generalised this by reducing the firstorder theory of product structures to the first-order theory of the elements and the second-order theory of the indices (that is quantification over set variables is allowed). In particular, they obtained decision procedures that would apply to index theory signatures containing the cardinality operator. Being able to specify counting properties is very important in verification (see for instance [13]) and in particular in verification using array theories $[14,1]$.

This section shows how to combine a certain second-order theory of indices including the cardinality operator with the existential fragment of the first-order theory of power structures.

### 4.1. Explicit Sets of Indices

The intuition behind Theorem 11 is that the verifier of Figure 1 is solving constraints on the power structure indices. This is schematically presented in Figure 2. The figure represents a Venn region determined by applying Boolean operations (union, intersection and complement) to some set variables. These variables are interpreted to be sets of indices of the power structure. All indices must remain within the boundaries of the main region $A$. This region corresponds to the positive literals of the formula $\varphi$ of Figure 1: $\wedge_{i=1}^{l} R_{i}\left(t_{1}^{i}, \ldots, t_{a_{i}}^{i}\right)$. The negative literals ${ }_{j=l+1}^{k} \neg R_{j}\left(t_{1}^{j}, \ldots, t_{a_{j}}^{j}\right)$ when interpreted with the semantics of the power structure correspond to existential constraints. These effectively require a cardinality greater or equal than one in certain subregions of $A$.


Figure 2: An example Venn region with product constraints.
To generalize Theorem 11 we use the logic BAPA [28], whose language allows to express Boolean algebra and cardinality constraints on sets. The satisfiability problem for the quantifier-free fragment of BAPA, which we will denote by QFBAPA, is in NP (see Section 3 of [29]).

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 $M A X C$ gives the cardinality of $\mathcal{U}$. For now, we assume this cardinality to be finite. In Section 5, we show that this restriction can be lifted.

For our proofs we will rely on the method of [29] and on the following property of integer conic hulls

$$
\begin{aligned}
& F::=A\left|F_{1} \wedge F_{2}\right| F_{1} \vee F_{2} \mid \neg F \\
& A::=B_{1}=B_{2}\left|B_{1} \subseteq B_{2}\right| T_{1}=T_{2}\left|T_{1} \leq T_{2}\right| K \operatorname{dvd} T \\
& B::=x|\emptyset| \mathcal{U}\left|B_{1} \cup B_{2}\right| B_{1} \cap B_{2} \mid B^{c} \\
& T::=k|K| \operatorname{MAXC}\left|T_{1}+T_{2}\right| K \cdot T| | B \mid \\
& K::=\ldots|-2|-1|0| 1|2| \ldots
\end{aligned}
$$

Figure 3: QFBAPA's syntax

Definition 12. Given a subset $S \subseteq \mathbb{R}^{n}$, the integer conic hull of $S$ is the set:

$$
\operatorname{int}_{\text {cone }}(S)=\left\{\sum_{i=1}^{t} \lambda_{i} s_{i} \mid t \geq 0, s_{i} \in S, \lambda_{i} \in \mathbb{N}\right\}
$$

Theorem 13 (Eisenbrand-Shmonin). Let $X \subseteq \mathbb{Z}^{n}$ be a finite set of integer vectors and $b \in \operatorname{int}_{\text {cone }}(X)$. Then there exists a subset $X^{\prime} \subseteq X$ such that $b \in \operatorname{int}_{\text {cone }}\left(X^{\prime}\right)$ and $\left|X^{\prime}\right| \leq 2 n \log (4 n M)$ where $M=\max _{\mathbf{x} \in X}\|\mathbf{x}\|_{\infty}$.

### 4.2. Complexity of the Existential Fragment of QFBAPA with Interpreted

 SetsWe now extend the NP membership of product structures given in Theorem 11 and of QFBAPA to the situation where we interpret QFBAPA sets as subsets of the set $I$ in which quantifier-free formulae hold. As we saw in Section 4.1, this can be seen as a generalisation from cardinality constraints of size one to arbitrary linear arithmetic restrictions.

Definition 14. We consider the satisfiability problem for QFBAPA formulae $F$ whose set variables are index sets defined by quantifier-free formulas $\varphi_{i}$ of $L$ applied to either component theory constants or to components of the power structure variables
$\exists c_{1}, \ldots, c_{m} \cdot \exists x_{1}, \ldots, x_{n}$.

$$
F\left(S_{1}, \ldots, S_{k}\right) \wedge \bigwedge_{i=1}^{k} S_{i}=\left\{r \in I \mid \varphi_{i}\left(x_{1}(r), \ldots, x_{n}(r), c_{1}, \ldots, c_{m}\right)\right\}
$$

We call this problem interpreted QFBAPA, which we write as QFBAPAI.

Theorem 15. $T h_{\exists^{*}}(\mathcal{M}) \in N P$ if and only if $Q F B A P A I \in N P$.
Proof. 1) Let $V_{\text {QFBAPA }}$ be a polynomial time verifier for QFBAPA and let $V_{C}$ be a polynomial time verifier for the component theory. Figure 4 gives a verifier $V$ for QFBAPAI. We abbreviate $\left(x_{1}, \ldots, x_{n}\right)$ by $\bar{x}$ and $\left(c_{1}, \ldots, c_{m}\right)$ by $\bar{c}$.

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

1. Interpret $w$ as:
(a) a list of indices $i_{1}, \ldots, i_{N} \in\left\{0, \ldots, 2^{e}-1\right\}$ 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 $C^{\prime}$ for $V_{P A}$ on input $y^{\prime}$ defined below.
(d) a bit $b$ indicating if the solution sets cover the whole $I$.
2. Set $y=\exists \bar{c}, \bar{x}_{1}, \ldots, \bar{x}_{N} . \bigwedge_{\beta_{j} \in\left\{i_{1}, \ldots, i_{N}\right\}} \bigwedge_{i=1}^{k} \varphi_{i}\left(\bar{x}_{j}, \bar{c}\right)^{\beta_{j}(i)} \quad(*)$.
3. Set $y^{\prime}=\exists S_{1}^{\prime}, \ldots, S_{k}^{\prime} \cdot F\left(S_{1}^{\prime}, \ldots, S_{k}^{\prime}\right) \wedge \bigwedge_{\beta_{j} \in\left\{i_{1}, \ldots, i_{N}\right\}} \bigcap_{i=1}^{k} S_{i}^{\beta_{j}(i)} \neq \emptyset \quad(* *)$.
4. If $b=0$ then $\operatorname{set}(*)=\wedge \neg \bigvee_{\beta_{j} \in\left\{i_{1}, \ldots, i_{N}\right\}} \bigwedge_{i=1}^{k} \varphi_{i}\left(\bar{x}_{0}, \bar{c}\right)^{\beta_{j}(i)}$ and add $\bar{x}_{0}$ as a top-level existential quantifier.
If $b=1$ then set $(* *)=\wedge \bigcup_{\beta_{j} \in\left\{i_{1}, \ldots, i_{N}\right\}} \bigcap_{i=1}^{k} S_{i}^{\beta_{j}(i)}=I$.
5. Run $V_{C}$ on $\langle y, C\rangle$.
6. Run $V_{\text {QFBAPA }}$ on $\left\langle y^{\prime},\left\langle\left\{i_{1}, \ldots, i_{N}\right\}, C^{\prime}\right\rangle\right\rangle$.
7. Accept iff all runs accept.

Figure 4: Verifier for QFBAPA interpreted over index-sets.
$\Rightarrow)$ If $x \in$ QFBAPAI then there exist $\bar{c}, \bar{s}$ satisfying

$$
F\left(S_{1}, \ldots, S_{k}\right) \wedge \bigwedge_{i=1}^{k} S_{i}=\left\{r \in I \mid \varphi_{i}(\bar{s}(r), \bar{c})\right\}
$$

Define $S_{i}:=\left\{r \in I \mid \varphi_{i}(\bar{s}(r), \bar{c})\right\}$. Then, the method of [29] applied to $F\left(S_{1}, \ldots, S_{k}\right)$ yields a formula $G \wedge \bigwedge_{i=0}^{p}\left|b_{i}\right|=k_{i}$. Using $\left|b_{i}\right|=\sum_{\beta \models b_{i}}\left|\bigcap_{i=1}^{k} S_{i}^{\beta(i)}\right|$ and setting $p_{\beta}:=\bigcap_{i=1}^{k} S_{i}^{\beta(i)}, l_{\beta}:=\left|p_{\beta}\right|$, yields $G \wedge \bigwedge_{i=0}^{p} \sum_{j=0}^{2^{e}-1} \llbracket b_{i} \rrbracket_{\beta_{j}} \cdot l_{\beta_{j}}=k_{i}$. Remove those $\beta$ where $l_{\beta}=0$. Since

$$
p_{\beta}=\bigcap_{i=1}^{k}\left\{r \in I \mid \varphi_{i}(\bar{s}(r), \bar{c})\right\}^{\beta(i)}=\left\{r \in I \mid \bigwedge_{i=1}^{k} \varphi_{i}(\bar{s}(r), \bar{c})^{\beta(i)}\right\}
$$

this includes those $\beta$ such that $\bigwedge_{i=1}^{k} \varphi_{i}(\bar{s}(r), \bar{c})^{\beta(i)}$ is not satisfiable. We obtain a reduced set of indices $\mathcal{R} \subseteq\left\{0, \ldots, 2^{e}-1\right\}$ where $G \wedge \bigwedge_{i=0}^{p} \sum_{\beta \in \mathcal{R}} \llbracket b_{i} \rrbracket_{\beta} \cdot l_{\beta}=k_{i}$. Eisenbrand-Shmonin's theorem yields a polynomial family of indices such that $G \wedge \bigwedge_{i=0}^{p} \sum_{\beta \in\left\{i_{1}, \ldots, i_{N}\right\} \subseteq \mathcal{R}} \llbracket b_{i} \rrbracket_{\beta} \cdot l_{\beta}^{\prime}=k_{i}$ for non-zero $l_{\beta}^{\prime}$.

For each $\beta \in\left\{i_{1}, \ldots, i_{N}\right\}$, since $l_{\beta} \neq 0$, there exists $r_{\beta} \in I$ such that $\bigwedge_{i=1}^{k} \varphi_{i}\left(\bar{s}\left(r_{\beta}\right), \bar{c}\right)^{\beta(i)}$. So the formula $y$ without $\left(^{*}\right)$ is satisfied.

The satisfiability of the cardinality restrictions on $l_{\beta}^{\prime}$ implies the existence of sets of indices $S_{i}^{\prime}$ such that for each $\beta \in\left\{i_{1}, \ldots, i_{N}\right\},\left|p_{\beta}^{\prime}\right|=l_{\beta}^{\prime}$. Observe that $|I|=\sum_{\beta \in \mathcal{R}} l_{\beta}$. Distinguish two cases:

- If $|I|>\sum_{\beta \in\left\{i_{1}, \ldots, i_{N}\right\}} l_{\beta}^{\prime}$ then there is at least one index $r_{0}$ such that $\bar{s}\left(r_{0}\right)$ satisfies $\bigwedge_{i=1}^{k} \varphi_{i}\left(\bar{s}\left(r_{0}\right), \bar{c}\right)^{\beta(i)}$ for $\beta \notin\left\{i_{1}, \ldots, i_{N}\right\}$. Therefore, the formula $y$ with $\left({ }^{*}\right)$ is satisfied. In this case, define:

$$
\bar{s}^{\prime}(r)= \begin{cases}\bar{s}\left(r_{\beta}\right) & \text { if } r \in p_{\beta}^{\prime} \\ \bar{s}\left(r_{0}\right) & \text { otherwise }\end{cases}
$$

and choose $b=0$.

- If $|I|=\sum_{\beta \in\left\{i_{1}, \ldots, i_{N}\right\}} l_{\beta}^{\prime}$ then define:

$$
\bar{s}^{\prime}(r)=\left\{\bar{s}\left(r_{\beta}\right) \quad \text { if } r \in p_{\beta}^{\prime} \text { and } \beta \in\left\{i_{1}, \ldots, i_{N}\right\}\right.
$$

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 $S_{i}^{\prime \prime}=\left\{r \in I \mid \varphi_{i}\left(\bar{s}^{\prime}(r), \bar{c}\right)\right\}$. Then $S_{1}^{\prime \prime}, \ldots, S_{k}^{\prime \prime}$ satisfy $y^{\prime}$ by construction

- Observe that for each $\beta \in\left\{i_{1}, \ldots, i_{N}\right\}, p_{\beta}^{\prime \prime}=p_{\beta}^{\prime}$.
- For each $\beta \in\left\{i_{1}, \ldots, i_{N}\right\}, p_{\beta}^{\prime \prime} \neq \emptyset$, since $l_{\beta}^{\prime} \neq 0$.
- If $b=1$ then $\bigcup_{\beta \in\left\{i_{1}, \ldots, i_{N}\right\}}^{\bigcup} p_{\beta}^{\prime \prime}=I$ since $|I|=\sum_{\beta \in\left\{i_{1}, \ldots, i_{N}\right\}} l_{\beta}^{\prime}$.
- The cardinality restrictions are satisfied by definition.

Again, since $N$ is polynomial in $|x|,\left|y^{\prime}\right|$ is polynomial in $|x|$ too. By the above, it is also satisfiable. Thus, there exists a polynomially-sized certificate $C^{\prime}$ for $V_{P A}$ such that $V_{\text {QFBAPA }}$ accepts $\left\langle\left\{i_{1}, \ldots, i_{N}\right\}, C^{\prime}\right\rangle$ in polynomial time. So $V$ accepts $\left\langle x,\left\langle\left\{i_{1}, \ldots, i_{N}\right\}, C, C^{\prime}, b\right\rangle\right\rangle$ in polynomial time.
$\Leftarrow$ If $V$ accepts $\langle x, w\rangle$ in polynomial time then

- $\langle y, C\rangle$ is accepted by $V_{C}$, so there is a tuple $\bar{c}$ and for each $\beta \in$ $\left\{i_{1}, \ldots, i_{N}\right\}$, there are tuples $s_{\beta}$, such that $\bigwedge_{i=1}^{k} \varphi_{i}\left(s_{\beta}(1), \ldots, s_{\beta}(n), \bar{c}\right)^{\beta(i)}$.
- $\left\langle y^{\prime},\left\langle\left\{i_{1}, \ldots, i_{N}\right\}, C^{\prime}\right\rangle\right\rangle$ is accepted by $V_{\text {QFBAPA }}$, so there exist sets $S_{i}^{\prime}$ such that:

$$
F\left(S_{1}^{\prime}, \ldots, S_{k}^{\prime}\right) \wedge \bigwedge_{\beta \in\left\{i_{1}, \ldots, i_{N}\right\}} \bigcap_{i=1}^{k} S_{i}^{\prime \beta(i)} \neq \emptyset
$$

Interpreting $S_{i}^{\prime}$ as index sets, we define an array $\bar{s}$ distinguishing two cases:

- If $b=0$ then $V_{C}$ accepts:

$$
\left\langle\exists \bar{c}, \exists \bar{x}_{1}, \ldots, \bar{x}_{N}, \bar{x}_{0} \ldots \neg \bigvee_{\beta \in\left\{i_{1}, \ldots, i_{N}\right\}} \bigwedge_{i=1}^{k} \varphi_{i}\left(\bar{x}_{0}, \bar{c}\right)^{\beta(i)}, C\right\rangle
$$

Let $s_{0}$ be a satisfying tuple for $\bar{x}_{0}$. Define:

$$
\bar{s}(r)= \begin{cases}s_{\beta} & \text { if } r \in p_{\beta}^{\prime} \text { and } \beta \in\left\{i_{1}, \ldots, i_{N}\right\} \\ s_{0} & \text { otherwise }\end{cases}
$$

- If $b=1$ then $S_{i}^{\prime}$ satisfies $\bigcup_{\beta \in\left\{i_{1}, \ldots, i_{N}\right\}} \bigcap_{i=1}^{k} S_{i}^{\prime \beta(i)}=I$. Define:

$$
\bar{s}(r)=\left\{s_{\beta} \quad \text { if } r \in p_{\beta}^{\prime} \text { and } \beta \in\left\{i_{1}, \ldots, i_{N}\right\}\right.
$$

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

$$
\exists \bar{c}, \bar{x} \cdot F\left(S_{1}, \ldots, S_{k}\right) \wedge \bigwedge_{i=1}^{k} S_{i}=\left\{r \in I \mid \varphi_{i}(\bar{x}(r), \bar{c})\right\}
$$

For each $\beta \in\left\{i_{1}, \ldots, i_{N}\right\}$ :

$$
p_{\beta}=\left\{r \in I \mid \bigwedge_{i=1}^{k} \varphi(\bar{s}(r), \bar{c})^{\beta(i)}\right\}=p_{\beta^{\prime}}
$$

so the cardinality conditions are met.
2) As in Theorem 11.

From a syntactic perspective, Theorem 15 shows that the formula

$$
\exists S_{1}, \ldots, S_{k} \cdot F\left(S_{1}, \ldots, S_{k}\right) \wedge \exists x_{1}, \ldots, x_{m} \cdot \bigwedge_{i=1}^{k} S_{i}=\left\{n \in \mathbb{N} \mid \varphi_{i}\left(x_{1}(n), \ldots, x_{m}(n)\right)\right\}
$$

is equivalent to the formula

$$
\begin{aligned}
& \exists N \in\{0,1\}^{\log _{2}(p(|F|))}, \beta_{1}, \ldots, \beta_{N} \in\{0,1\}^{k}, b \in\{0,1\} . \\
& \exists x_{1}, \ldots, x_{m} . \bigwedge_{\beta_{j} \in\left\{i_{1}, \ldots, i_{N}\right\}} \bigwedge_{i=1}^{k} \varphi_{i}\left(x_{j}\right)^{\beta_{j}(i)} \wedge\left(b \vee \neg \bigvee_{\beta_{j} \in\left\{i_{1}, \ldots, i_{N}\right\}} \bigwedge_{i=1}^{k} \varphi_{i}\left(\bar{x}_{0}\right)^{\beta_{j}(i)}\right) \\
& \wedge \exists S_{1}, \ldots, S_{k} \cdot F\left(S_{1}, \ldots, S_{k}\right) \wedge\left(\neg b \vee \bigcup_{\beta_{j} \in\left\{i_{1}, \ldots, i_{N}\right\}} \bigcap_{i=1}^{k} S_{i}^{\beta_{j}(i)}=I\right)
\end{aligned}
$$

for $p$ a polynomial and $|F|$ the size of formula $F$. The advantage of this form is that the conjuncts do not share variables. Thus, the part of the formula referring to the elements in the component of the power structure and the part of the formula referring to the index set can be solved independently. ${ }^{1}$

A closely related result can be found in [1, Section 5] in the context of the study of array theories. The more general setting of power structures allows to generalize this result to an arbitrary index and element theories while retaining the NP bound.

Theorem 15 is also interesting from the point of view of combination of theories as it extends the results of [46] in an unexpected direction by interpreting the shared set variables as sets of elements.

## 5. Weak Power Structures

In this section, we lift the restriction on the size of the universe of the set of indices in the index theory. Then we recall the definition of weak power structures. Adapting the theory of Section 4, we will show that the existential fragment of the first-order theory of a weak power structure is also in NP.

### 5.1. Complexity of the Existential Fragment of QFBAPA with the Finiteness predicate

We extend the language of QFBAPA in Figure 3 with the ability of specifying that some subsets are finite or infinite. Figure 5 shows the syntax of the fragment, which we call QFBAPA ${\aleph_{0}}$. A formula $F$ is a propositional

[^1]\[

$$
\begin{aligned}
& F::=A\left|F_{1} \wedge F_{2}\right| F_{1} \vee F_{2} \mid \neg F \\
& A::=B_{1} \subseteq B_{2}\left|T_{1} \leq T_{2}\right||B|=T| | B\left|\geq \aleph_{0}\right||B|<\aleph_{0} \\
& B::=x|\emptyset| \mathcal{U}\left|B_{1} \cup B_{2}\right| B_{1} \cap B_{2} \mid B^{c} \\
& T::=k|K| T_{1}+T_{2} \mid K \cdot T \\
& K::=\ldots|-2|-1|0| 1|2| \ldots
\end{aligned}
$$
\]

Figure 5: QFBAPA ๗ $_{\aleph_{0}}$ 's syntax
combination of atoms $A$. An atom $A$ is either the inclusion of two Boolean expressions of sets $B$, an inequality of arithmetic expressions $T$, a statement specifying the cardinality of a set $B$ or specifying whether $B$ is infinite or finite. Set variables are represented by $x$ and the universe of interpretation is represented by $\mathcal{U}$. Similarly, $k$ denotes an integer variable and $K$ an integer constant. The remaining interpretations are standard.

Theorem 16. $Q F B A P A_{\aleph_{0}}$ is in $N P$.
Proof. Figure 6 gives a polynomial time verifier for QFBAPA $\aleph_{\aleph_{0}}$. By Lemma 8, it suffices to prove the correctness of the procedure for disjuncts in the DNF of the input formula. These will be of the form

$$
\begin{equation*}
\varphi \equiv F\left(b_{1}, \ldots, b_{k}\right) \wedge \bigwedge_{i=1}^{n} \operatorname{Fin}\left(c_{i}\right) \wedge \bigwedge_{j=1}^{m} \neg \operatorname{Fin}\left(d_{j}\right) \tag{1}
\end{equation*}
$$

where the $b_{l}, c_{i}, d_{j}$ are Boolean expressions.
$\Rightarrow)$ If $\varphi \in$ QFBAPA $_{\aleph_{0}}$ then there exist sets of indices $b_{l}, c_{i}, d_{j}$ satisfying (1). Each $d_{j}$ is infinite and thus, there exists at least one infinite elementary Venn region $\beta_{j}$ contained in $d_{j}$. Moreover, each $\beta_{j}$ may not be a part of a Boolean expression $b$ occurring in $F$ as a purely arithmetic formula or under the finiteness operator, it can only occur on atoms of the form $|b| \geq \aleph_{0}$ since otherwise its cardinality should be finite. We will denote by $C$ the set of all such Boolean expressions.

In total, this gives a linear number of Venn regions $\beta_{1}, \ldots, \beta_{N}$ in the size of the formula. These elementary Venn regions do not appear in $F$. So it follows that they form a polynomial-sized certificate such that the following
formula is satisfiable

$$
\begin{equation*}
F\left(b_{1}, \ldots, b_{k}\right) \wedge \bigwedge_{j=1}^{m}\left|p_{\beta_{j}}\right|=0 \wedge \bigwedge_{j=1}^{m} \beta_{j} \models d_{j} \wedge \bigwedge_{l \in\left\{l . b_{l} \in C\right\}} \beta_{j} \not \models b_{l} \tag{2}
\end{equation*}
$$

$\Leftarrow)$ If there is a polynomial-sized certificate such that formula 2 holds then it follows that the cardinality of each $\beta_{j}$ can be changed without affecting the truth of $F\left(b_{1}, \ldots, b_{k}\right)$. By hypothesis, $\beta_{j}$ is not in any $c_{i}$ so whathever (finite) model was obtained for $c_{i}$ is preserved. Finally, we can change the cardinality of $\beta_{j}$ to make the cardinality of $d_{j}$ infinite. With this change, formula 1 holds too.

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

1. Interpret $w$ as
(a) a list of elementary Venn regions $\beta_{1}, \ldots, \beta_{N} \in\{0,1\}^{k}$ where $e$ is the number of set variables in $x$.
(b) a certificate $C$ for $V_{\text {QFBAPA }}$ on input

$$
x^{\prime}=F\left(b_{1}, \ldots, b_{k}\right) \wedge \bigwedge_{j=1}^{m}\left|p_{\beta_{j}}\right|=0 \wedge \bigwedge_{j=1}^{m} \beta_{j} \models d_{j} \wedge \bigwedge_{l \in\left\{l . b_{l} \in C\right\}} \beta_{j} \not \models b_{l}
$$

2. Accept iff $V_{\text {QFBAPA }}$ accepts $\left\langle x^{\prime}, C\right\rangle$.

Figure 6: Verifier for QFBAPA $\aleph_{\aleph_{0}}$

### 5.2. Definition of Weak Power Structures

As in Definition 9, we fix a first-order language $L$, a non-empty set $I$ and a structure $\mathcal{M}$ with carrier $M$. In this context, we also fix a distinguished element $e \in M$.
Definition 17. The weak power structure $\Pi^{*}$ over I has domain

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

and interprets the symbols of $L$ 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 $\bar{a} \in\left(M^{I}\right)^{n}$, $f^{\Pi}(\bar{a})(i)=f^{\mathcal{M}}(\bar{a}(i))$. For each relation symbol $R, n \in \mathbb{N}$ and $\bar{a} \in\left(M^{I}\right)^{n}$, $\bar{a} \in R^{\Pi}$ if and only if for every $i \in I, \bar{a}(i) \in R^{\mathcal{M}}$.

### 5.3. Complexity of the Existential Fragment of Weak Power Structures

Definition 18. We consider the satisfiability problem for QFBAPA formulas $F$ whose set variables are index sets defined by quantifier-free formulas $\varphi_{i}$ of $L$ applied to either component theory constants or to components of the weak power structure variables

$$
\begin{aligned}
& \exists c_{1}, \ldots, c_{m} \cdot \exists x_{1}, \ldots, x_{n} \\
& \qquad F\left(S_{1}, \ldots, S_{k}\right) \wedge \bigwedge_{i=1}^{k} S_{i}=\left\{r \in I \mid \varphi_{i}\left(x_{1}(r), \ldots, x_{n}(r), c_{1}, \ldots, c_{m}\right)\right\}
\end{aligned}
$$

We call this problem finite interpreted QFBAPA, which we write as QFBAPAFI.
Corollary 19. $T h_{\exists^{*}}(\mathcal{M}) \in N P$ if and only if $Q F B A P A F I \in N P$.
Proof. It suffices to combine the verifier of Figure 6 with that of Figure 4.
Corollary 20. $T h_{\text {ヨ* }}\left(\Pi^{*}\right) \in N P$.
Proof. It suffices to write a formula of $T h_{\exists^{*}}\left(\Pi^{*}\right)$ in the language of QFBAPAFI. A positive literal $R\left(t_{1}, \ldots, t_{a}\right)$ is written as $\left\{i \in I \mid R\left(t_{1}(i), \ldots, t_{a}(i)\right)\right\}=\mathcal{U}$. A negative literal $\neg R\left(t_{1}(i), \ldots, t_{a}(i)\right)$ is written as $\left|\left\{i \in I \mid R\left(t_{1}, \ldots, t_{j}\right)\right\}\right| \geq 1$. Finally, for each existential variable $a$ from the power structure domain, one needs to ensure that $|\{i \in I \mid a(i) \neq e\}|<\aleph_{0}$.

## 6. Power Structures with Order

This section describes the extension of QFBAPAI with an ordering relation. The ordering relation will be encoded with regular expressions with large alphabets, defined in Section 6.1. Section 6.2 gives an NP decision procedure for the combination of regular expressions and QFBAPA. Section 6.3 shows that the resulting theory can be combined by set interpretation with power structures.

### 6.1. Regular Expressions with Large Alphabets

Kleene introduced in [26] a theory of regular sets of tables to describe the behaviour of McCulloch-Pitts nerve nets. In his interpretation, the columns of these tables are indexed by the time unit and the rows by the neuron in the described system. A particular entry is set to one if and only if the corresponding neuron fires at the time indicated by its column index. Büchi
showed $[11,10]$ that under the interpretation that the rows correspond to sets of natural numbers and the entries are interpreted as membership in these sets, then the relations between the sets represented by the rows are exactly those definable in weak monadic second-order theory of one successor (WS1S). ${ }^{2}$ In particular, [11, Corollary 1] gives an alternative proof to [17, Theorem 10.1] of the decidability of this theory.

Example 21. Consider the regular set of tables given by the expression $\left(\binom{1}{0}\binom{0}{1}\right)^{*}$. One possible table satisfying this expression would be

$$
\begin{array}{lllllllllllllllllll}
\text { A } & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \cdots \\
\text { B } & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \cdots
\end{array}
$$

Here $A=\{1,3,5,7\}$ and $B=\{2,4,6,8\}$. In general, the corresponding WS1S formula would specify the set of the first n odd natural numbers for $A$ and the set of the first $n$ even natural numbers for $B$.

Note that a minor difference between the interpretation of Kleene and that of Büchi is that the latter specifies an infinite table which is filled with strings of zeros in all columns but in a finite subset.

We enrich the theory of the indices of power structures by considering those relations definable in WS1S. In particular, we allow to express an order between the components. To this end, we will further combine our algebra of indices with the sets expressed in the formalism of Büchi, which we term regular expressions over large alphabets. Figure 7 shows the syntax of such expressions.

### 6.2. Combining QFBAPA and Regular Expressions using the Parikh Image

This section studies conditions under which we can combine the specifications of QFBAPA with regular expressions. We see that combining models of each theory reduces to computing the Parikh image of the regular language denoted by the regular expression.

[^2]\[

R:: \left.=\left($$
\begin{array}{c}
l_{1} \\
l_{2} \\
\ldots \\
l_{K}
\end{array}
$$\right)\left|R_{1} R_{2}\right| R^{*} \right\rvert\, R_{1} \cup R_{2}
\]

Figure 7: Syntax of Regular Expressions over Large Alphabets

Definition 22. RegExp-QFBAPA is the theory consisting of conjunctions of QFBAPA formulae $F$ and regular expressions $R$.

Definition 23. A model of a RegExp-QFBAPA formula $F \wedge R$ consists of

- a cardinality for each Venn region.
- a regular table [26] t such that:

$$
\begin{aligned}
& t \in \llbracket R \rrbracket \\
& \text { for each letter } l \in \Sigma, \operatorname{count}_{t}(l)=\operatorname{card}(l)
\end{aligned}
$$

where count $t_{t}(l)$ denotes the number of times letter $l$ appears in the table $t$.
The condition $t \in \llbracket R \rrbracket$ can be checked independently by a procedure for the theory of regular sets of tables. On the other hand, the condition $\operatorname{count}_{t}(l)=\operatorname{card}(l)$ depends also on the theory of sets and cardinalities. We remove this dependence by computing for each table $t \in \llbracket R \rrbracket$ and each letter $l \in \Sigma, \operatorname{count}_{t}(l)$. We then collect for each table $t$ the corresponding vector of counts.

Definition 24 (Parikh Image).
The Parikh image of a regular expression with large alphabet $R$ is the set

$$
\operatorname{Parikh}(R)=\left\{\left(\operatorname{count}_{t}\left(l_{1}\right), \ldots, \operatorname{count}_{t}\left(l_{2^{K}}\right)\right) \mid t \in \llbracket R \rrbracket\right\}
$$

To compute the Parikh image of a regular expression $R$ we use the following two well-known results

Lemma 25 ([12]).
Any regular expression can be converted into an NFA in $\Theta(|R|)$ time.

Lemma 26 ([36]).
Given an NFA A, the set Parikh $(A)$ is definable by an existential Presburger formula of size $O(|A|)$.

Combining Lemma 25 and 26, we obtain an $O(|R|)$ size existential Presburger formula $\psi(\bar{x})$ such that $\operatorname{Parikh}(R)=\operatorname{Parikh}(A)=\{\bar{x} \mid \psi(\bar{x})\}$. A structure is a model of $F \wedge R$ if and only if there exists $t \in \llbracket R \rrbracket$ and all Venn regions corresponding to alphabet letters occurring in $R$ have the cardinality given by the Parikh image of $t$.

If $F$ has variables $S_{1}, \ldots, S_{k}$ then we defined the Venn region corresponding to alphabet letter $l$ as $p_{l}:=\bigcap_{i=1}^{k} S_{i}^{l_{i}}$. If $l_{1}, \ldots, l_{s}$ are the alphabet letters occurring in $R$, then the above condition translates to the formula $\left(\left|p_{l_{1}}\right|, \ldots,\left|p_{l_{s}}\right|\right) \in \operatorname{Parikh}(R)$. This is equivalent to $\psi\left(\left|p_{l_{1}}\right|, \ldots,\left|p_{l_{s}}\right|\right)$. Conjoining this with $F$ and applying the decision procedure for QFBAPA yields an NP algorithm for RegExp-QFBAPA.

Theorem 27. RegExp-QFBAPA is in NP.

### 6.3. Combination with Power Structures

This section shows that the satisfiability problem of RegExp-QFBAPA constraints conjoined with power structure constraints is solvable in NP. The proof is similar to Theorem 15 but regular constraints allow to avoid the use of sparse solutions of QFBAPA formulae.

Definition 28. Consider conjunctions of formulas $F$ in QFBAPA, regular expressions $R$ over $s$ letters with $1 \leq s \leq 2^{k}$ and component structure specifications $\varphi_{i}$ with $1 \leq i \leq k$, i.e. of the form

$$
\begin{aligned}
& \exists S_{1}, \ldots, S_{k} \cdot F\left(S_{1}, \ldots, S_{k}\right) \wedge \\
& \exists t_{1}, \ldots, t_{k} \cdot\left(\left(t_{1}, \ldots, t_{k}\right) \in R\left(l_{1}, \ldots, l_{s}\right) \wedge \bigwedge_{i=1}^{k} S_{i}=\left\{n \in \mathbb{N} \mid t_{i}(n)\right\}\right) \wedge \\
& \exists x_{1}, \ldots, x_{m} \cdot \bigwedge_{i=1}^{k} S_{i}=\left\{n \in \mathbb{N} \mid \varphi_{i}\left(x_{1}(n), \ldots, x_{m}(n)\right)\right\}
\end{aligned}
$$

where $\varphi_{i}$ are formulae of a theory whose satisfiability problem is decidable in NP. We call this theory RegExp-QFBAPA-Power.

The resulting constraints are schematically represented in Figure 8. Intuitively, the regular constraints impose an order on elementary Venn regions. We represent this ordering as an automaton whose states are particular elements in the Venn region denoted by bold squares and whose transitions are marked by arrows between states. Each dot in the diagram (circle or square) represents an index in $\mathcal{U}$. Theorem 29 shows that whenever the theory of the formulae $\varphi_{i}$ is decidable in NP, the specifications of Definition 28 are also decidable in NP. Intuitively, this is because the procedure only needs to know the value of the power structures elements $x_{i}$ on one index per elementary Venn region (i.e. on the squares of Figure 8. The remaining positions of the power structure element are filled by copying the value of these indices.


Figure 8: An example Venn region with regular constraints.

Theorem 29. If $T h_{\exists^{*}}(\mathcal{M}) \in N P$ then RegExp-QFBAPA-Power $\in N P$.
Proof. 1) Let $V_{Q F B A P A_{\aleph_{0}}}$ be a polynomial time verifier for QFBAPA $_{\aleph_{0}}$ and let $V_{C}$ be a polynomial time verifier for the component theory. Figure 9 gives a verifier $V$ for RegExp-QFBAPA-Power. We abbreviate $\left(x_{1}, \ldots, x_{m}\right)$ by $\bar{x}$ and $\left(t_{1}, \ldots, t_{k}\right)$ by $\bar{t}$. If $S_{i}$ are the set variables of the formula and $\beta \in\{0,1\}^{k}$ then we write $p_{\beta}=\cap_{i=1}^{k} S_{i}^{\beta(i)}, s_{\beta}=\left|p_{\beta}\right|, \varphi_{\beta}(\bar{x})=\wedge_{i=1}^{k} \varphi_{i}(\bar{x})^{\beta(i)}$ and $t_{\beta}(n)=\wedge_{i=1}^{k} t_{i}(n)^{\beta(i)}$. Finally, $l_{0}$ denotes a bit-string of $k$ zeros.

We show that $x \in$ RegExp-QFBAPA-Power if and only if there exists a polynomial-size certificate $w$ such that $V$ accepts $\langle x, w\rangle$.
$\Rightarrow)$ If $x \in \operatorname{RegExp-QFBAPA}-P o w e r ~ t h e n ~ t h e r e ~ e x i s t ~ s, ~ \bar{t}$ satisfying:

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

1. Interpret $w$ as:
(a) a list of letters $l_{1}, \ldots, l_{c} \in\{0,1\}^{k}$.
(b) a certificate $C$ for $V_{C}$ on input $y$.
(c) a certificate $C^{\prime}$ for $V_{Q F B A P A_{\aleph_{0}}}$ on input $y^{\prime}$.
where $y=\bigwedge_{l \in\left\{l_{0}, \ldots, l_{c}\right\}} \exists \bar{x} \cdot \varphi_{l}(\bar{x})$ and
$y^{\prime}=\exists S_{1}, \ldots, S_{k} \cdot F\left(S_{1}, \ldots, S_{k}\right) \wedge \psi\left(\left|p_{l_{1}}\right|, \ldots,\left|p_{l_{s}}\right|\right) \wedge \bigwedge_{l \in\left\{l_{0}, \ldots, l_{c}\right\}} p_{l} \neq \emptyset \wedge$

$$
\bigvee_{l \in\left\{l_{0}, \ldots, l_{c}\right\}} p_{l}=\mathcal{U} \wedge s_{l_{0}}=\aleph_{0} \wedge \bigwedge_{i=1}^{k} s_{l_{i}}<\aleph_{0}
$$

2. Run $V_{C}$ on $\langle y, C\rangle$ and $V_{Q F B A P A_{\aleph_{0}}}$ on $\left\langle y^{\prime}, C^{\prime}\right\rangle$. Accept iff both runs accept.

Figure 9: Verifier for RegExp-QFBAPA-Power.

$$
\begin{aligned}
& F\left(S_{1}, \ldots, S_{k}\right) \wedge \\
& \bar{t} \in R\left(l_{1}, \ldots, l_{s}\right) \wedge \bigwedge_{i=1}^{k} S_{i}=\left\{n \in \mathbb{N} \mid t_{i}(n)\right\} \wedge \\
& \bigwedge_{i=1}^{k} S_{i}=\left\{n \in \mathbb{N} \mid \varphi_{i}(\bar{s}(n))\right\}
\end{aligned}
$$

Define $S_{i}:=\left\{n \in \mathbb{N} \mid \varphi_{i}(\bar{s}(n))\right\}=\left\{n \in \mathbb{N} \mid t_{i}(n)\right\}$. By Section 6.2, we know that expression $\bar{t} \in R\left(l_{1}, \ldots, l_{s}\right)$ is true if and only if $\psi\left(\left|p_{l_{1}}\right|, \ldots,\left|p_{l_{s}}\right|\right)$ is true. Applying the method of [29] to $F\left(S_{1}, \ldots, S_{k}\right) \wedge \psi\left(\left|p_{l_{1}}\right|, \ldots,\left|p_{l_{s}}\right|\right)$ yields a formula $G \wedge \bigwedge_{i=0}^{p} \sum_{j=0}^{2^{k}-1} \llbracket b_{i} \rrbracket_{\beta_{j}} \cdot s_{\beta_{j}}=k_{i}$. Removing those $\beta$ where $s_{\beta}=0$, we find a set of indices $\mathcal{R} \subseteq\left\{0, \ldots, 2^{k}-1\right\}$ where $G \wedge \bigwedge_{i=0}^{p} \sum_{\beta \in \mathcal{R}} \llbracket b_{i} \rrbracket_{\beta} \cdot s_{\beta}=k_{i}$.

Note that $l_{0} \in \mathcal{R}$ always holds since we are using Büchi's interpretation. From $p_{\beta}=\left\{n \in \mathbb{N} \mid \varphi_{\beta}(\bar{s}(n))\right\}=\left\{n \in \mathbb{N} \mid t_{\beta}(n)\right\}$, follows that $\mathcal{R} \subseteq\left\{l_{0}, \ldots, l_{s}\right\}$ and $l \in \mathcal{R}$ implies that $\varphi_{\beta}(\bar{s}(n))$ is true in some index $n_{l}$. If $c+1=|\mathcal{R}|=$ $O(|R|)$ then we write $\mathcal{R}=\left\{l_{0}, l_{1}, \ldots, l_{c}\right\}$.
$y$ is satisfiable since for each conjunct $\exists \bar{x} \cdot \varphi_{l}(\bar{x})$ we have a witness $\bar{s}\left(n_{l}\right)$.
$y^{\prime}$ is satisfiable since $s_{l} \neq 0$ for each $l \in \mathcal{R}$ by hypothesis, $s_{l_{0}}=\aleph_{0}$ and $\bigwedge_{i=1}^{k} s_{l_{i}}<\aleph_{0}$ follows from Büchi's interpretation of regular expressions and $\psi\left(\left|p_{l_{1}}\right|, \ldots,\left|p_{l_{s}}\right|\right)$ is true since $\bar{t} \in R\left(l_{1}, \ldots, l_{s}\right)$ holds by assumption.

The size of the formulas is linear in the size of the input. Thus, there exist polynomial-size certificates $C$ and $C^{\prime}$ such that $V_{C}$ and $V_{\text {QFBAPA }_{\aleph_{0}}}$ accept. Then, $w=\left\langle l_{0}, \ldots, l_{c}, C, C^{\prime}\right\rangle$ is a polynomial-size certificate such that $V$ accepts $\langle x, w\rangle$.
$\Leftarrow)$ If $V$ accepts $\langle x, w\rangle$ in polynomial time then $\langle y, C\rangle$ is accepted by $V_{C}$, so for each $l \in\left\{l_{0}, \ldots, l_{c}\right\}$ there is a tuple $\overline{x_{l}}$ such that $\varphi_{l}\left(\overline{x_{l}}\right)$ holds, and $\left\langle y^{\prime}, C^{\prime}\right\rangle$ is accepted by $V_{Q F B A P A_{\aleph_{0}}}$, so there exist sets $S_{i}$ making true the formula

$$
\begin{aligned}
& F\left(S_{1}, \ldots, S_{k}\right) \wedge \psi\left(\left|p_{l_{1}}\right|, \ldots,\left|p_{l_{s}}\right|\right) \wedge \bigwedge_{l \in\left\{l_{0}, \ldots, l_{c}\right\}} p_{l} \neq \emptyset \wedge \\
& \bigvee_{l \in\left\{l_{0}, \ldots, l_{c}\right\}} p_{l}=\mathcal{U} \wedge s_{l_{0}}=\aleph_{0} \wedge \bigwedge_{i=1}^{k} s_{l_{i}}<\aleph_{0}
\end{aligned}
$$

We need to exhibit $\bar{s}, \bar{t}$ satisfying

$$
\bar{t} \in R\left(l_{1}, \ldots, l_{s}\right) \wedge \bigwedge_{i=1}^{k} S_{i}=\left\{n \in \mathbb{N} \mid t_{i}(n)\right\} \wedge \bigwedge_{i=1}^{k} S_{i}=\left\{n \in \mathbb{N} \mid \varphi_{i}(\bar{s}(n))\right\}
$$

Define $\bar{s}(n)=\bar{x}_{l}$ if $n \in p_{l}$. This is well-defined for every $n \in \mathbb{N}$ since $\vee_{l \in\left\{l_{0}, \ldots, l_{c}\right\}} p_{l}=\mathcal{U}$. Moreover, $p_{l}=\left\{n \in \mathbb{N} \mid \varphi_{l}(\bar{s}(n))\right\}$ by double inclusion noting that $\varphi_{l} \wedge \varphi_{l^{\prime}}$ is unsatisfiable for $l \neq l^{\prime}$. Thus $S_{i}=\left\{n \in \mathbb{N} \mid \varphi_{i}(\bar{s}(n))\right\}$ since

$$
S_{i}=\bigcup_{l \in\left\{l_{0}, \ldots, l_{c}\right\}, l_{i}=1} p_{l}=\bigcup_{l \in\left\{l_{0}, \ldots, l_{c}\right\}, l_{i}=1}\left\{n \in \mathbb{N} \mid \varphi_{l}(\bar{s}(n))\right\}=\left\{n \in \mathbb{N} \mid \varphi_{i}(\bar{s}(n))\right\}
$$

By hypothesis $\psi\left(\left|p_{l_{1}}\right|, \ldots,\left|p_{l_{s}}\right|\right)$ is true. Therefore there exists $\bar{t} \in R\left(l_{1}, \ldots, l_{s}\right)$ such that $s_{l_{i}}=\left|\left\{n \in \mathbb{N} \mid t_{l_{i}}(n)\right\}\right|$ for $i=1, \ldots, c, s_{l_{0}}=\aleph_{0}$ and $s_{\beta}=0$ for $\beta \in\{0,1\}^{k} \backslash\left\{l_{0}, \ldots, l_{c}\right\}$, thus

$$
S_{i}=\bigcup_{l \in\left\{l_{0}, \ldots, l_{c}\right\}, l_{i}=1} p_{l}=\bigcup_{l \in\left\{l_{0}, \ldots, l_{c}\right\}, l_{i}=1}\left\{n \in \mathbb{N} \mid t_{l}(n)\right\}=\left\{n \in \mathbb{N} \mid t_{i}(n)\right\}
$$

## 7. Closure Properties

With the aim of studying the feasibility of integrating the developed specification languages with formal systems such as SMT solvers and Hoare logic, this section studies closure properties under propositional and imperative commands operations of the following extension of RegExp-QFBAPA-Power

Definition 30. The formulae of the power fragment are of the form

$$
\begin{aligned}
\exists \bar{k}^{\prime}, \bar{t}^{\prime}, \bar{x}^{\prime} \cdot F\left(\bar{S}, \bar{k}, \bar{k}^{\prime}\right) & \wedge\left(\bar{t}, \bar{t}^{\prime}\right) \in R\left(l_{1}, \ldots, l_{s}\right) \\
& \wedge \bigwedge_{i} S_{i}=\left\{n \in \mathbb{N} \mid \psi_{i}\left(\bar{t}(n), \bar{t}^{\prime}(n)\right)\right\} \\
& \wedge \bigwedge_{j} S_{j}=\left\{n \in \mathbb{N} \mid \varphi_{j}\left(\bar{x}(n), \bar{x}^{\prime}(n), \bar{c}\right)\right\}
\end{aligned}
$$

where the formulae $\psi_{i}$ are propositional and the formulae $\varphi_{j}$ are from a $N P$ decidable fragment.

### 7.1. Closure under Propositional Operations

Closure under propositional operations means that if we are given formulae $F, F_{1}, F_{2}$ in our fragment then $F_{1} \wedge F_{2}, F_{1} \vee F_{2}$ and $\neg F$ can also be written in our fragment (for details see [6]).

We will need the following two well-known facts about NFAs

- Every NFA has an equivalent regular expression. [47, Section 3.3]
- Regular languages are closed under complement. [47, Theorem 4.1]

Proposition 31. The formulae of the power fragment are closed under conjunction and disjunction. The formulae of obtained from the power fragment by removing existential quantifiers are also closed under negation.

Proof. 1) For conjunction, we assume that we are given two formulae

$$
\begin{aligned}
\varphi_{1}=\exists \bar{k}_{1}^{\prime}, \bar{t}_{1}^{\prime}, \bar{x}_{1}^{\prime} \cdot F_{1}\left(\bar{S}_{1}, \bar{k}_{1}, \bar{k}_{1}^{\prime}\right) \wedge\left(\bar{t}_{1}, \bar{t}_{1}^{\prime}\right) \in R\left(l_{11}, \ldots, l_{s 1}\right) \wedge \\
\bigwedge_{i 1} S_{i 1}=\left\{n \in \mathbb{N} \mid \psi_{i 1}\left(\bar{t}_{1}(n), \bar{t}_{1}^{\prime}(n)\right)\right\} \wedge \\
\bigwedge_{j 1} S_{j 1}=\left\{n \in \mathbb{N} \mid \varphi_{j 1}\left(\bar{x}_{1}(n), \bar{x}_{1}^{\prime}(n), \bar{c}_{1}\right)\right\} \\
\varphi_{2}=\exists \bar{k}_{2}^{\prime}, \bar{t}_{2}^{\prime}, \bar{x}_{2}^{\prime} \cdot F_{2}\left(\bar{S}_{2}, \bar{k}_{2}, \bar{k}_{2}^{\prime}\right) \wedge\left(\bar{t}_{2}, \bar{t}_{2}^{\prime}\right) \in R\left(l_{12}, \ldots, l_{t 2}\right) \wedge \\
\bigwedge_{i 2} S_{i 2}=\left\{n \in \mathbb{N} \mid \psi_{i 2}\left(\bar{t}_{2}(n), \bar{t}_{2}^{\prime}(n)\right)\right\} \wedge \\
\bigwedge_{j 2} S_{j 2}=\left\{n \in \mathbb{N} \mid \varphi_{j 2}\left(\bar{x}_{2}(n), \bar{x}_{2}^{\prime}(n), \bar{c}_{2}\right)\right\}
\end{aligned}
$$

then their conjunction is

$$
\begin{aligned}
& \varphi_{1} \wedge \varphi_{2}=\exists \bar{k}_{1}^{\prime}, \bar{t}_{1}^{\prime}, \bar{x}_{1}^{\prime} \cdot \exists \bar{k}_{2}^{\prime}, \bar{t}_{2}^{\prime}, \bar{x}_{2}^{\prime} . \\
& F_{1}\left(\bar{S}_{1}, \bar{k}_{1}, \bar{k}_{1}^{\prime}\right) \wedge\left(\bar{t}_{1}, \bar{t}_{1}^{\prime}\right) \in R\left(l_{11}, \ldots, l_{s 1}\right) \wedge \\
& F_{2}\left(\bar{S}_{2}, \bar{k}_{2}, \bar{k}_{2}^{\prime}\right) \wedge\left(\bar{t}_{2}, \bar{t}_{2}^{\prime}\right) \in R\left(l_{12}, \ldots, l_{t 2}\right) \wedge \\
& \bigwedge_{i 1} S_{i 1}=\left\{n \in \mathbb{N} \mid \psi_{i 1}\left(\bar{t}_{1}(n), \bar{t}_{1}^{\prime}(n)\right)\right\} \wedge \\
& \bigwedge_{j 1} S_{j 1}=\left\{n \in \mathbb{N} \mid \varphi_{j 1}\left(\bar{x}_{1}(n), \bar{x}_{1}^{\prime}(n), \bar{c}_{1}\right)\right\} \\
& \bigwedge_{i 2} S_{i 2}=\left\{n \in \mathbb{N} \mid \psi_{i 2}\left(\bar{t}_{2}(n), \bar{t}_{2}^{\prime}(n)\right)\right\} \wedge \\
& \bigwedge_{j 2} S_{j 2}=\left\{n \in \mathbb{N} \mid \varphi_{j 2}\left(\bar{x}_{2}(n), \bar{x}_{2}^{\prime}(n), \bar{c}_{2}\right)\right\}
\end{aligned}
$$

Since QFBAPA is closed under propositional operations and the conjunction of set interpretations is a set interpretation, it remains to analyse closure of a conjunction of regular expressions of the form

$$
\left(\bar{t}_{1},,_{1}^{\prime}\right) \in R\left(l_{11}, \ldots, l_{s 1}\right) \wedge\left(\bar{t}_{2}, \bar{t}_{2}^{\prime}\right) \in R\left(l_{12}, \ldots, l_{t 2}\right)
$$

In other words, we are looking for a third regular expression $R_{3}$ over letters that are the concatenation of one letter of $R\left(l_{11}, \ldots, l_{s 1}\right)$ and one letter of $R\left(l_{12}, \ldots, l_{t 2}\right)$, i.e. of the form $l_{i 1} \cdot l_{j 2}$, such that

$$
\left(\bar{t}_{1}, \bar{t}_{1}^{\prime}, \bar{t}_{2}, \bar{t}_{2}^{\prime}\right) \in R_{3}
$$

We show the the existence of such regular expression by converting to NFA's. By Lemma 25, there exist automata $A_{1}=\left(Q_{1}, \Sigma_{1}, \delta_{1}, q_{01}, F_{1}\right)$ and $A_{2}=$ $\left(Q_{2}, \Sigma_{2}, \delta_{2}, q_{02}, F_{2}\right)$ accepting the languages $R_{1}$ and $R_{2}$ respectively. Let $A_{3}=$ $\left(Q_{3}, \Sigma_{3}, \delta_{3}, q_{03}, F_{3}\right)$ be an automaton such that

- $Q_{3}=Q_{1} \times Q_{2}, \Sigma_{3}=\Sigma_{1} \times \Sigma_{2}, q_{03}=\left(q_{01}, q_{02}\right), F_{3}=F_{1} \times F_{2}$
- $\delta_{3}\left(\left(q_{1}, q_{2}\right),\left(a_{1}, a_{2}\right)\right)= \begin{cases}\left(\delta_{1}\left(q_{1}, a_{1}\right), \delta_{2}\left(q_{2}, a_{2}\right)\right) & \\ \left(q_{1}, \delta_{2}\left(q_{2}, a_{2}\right)\right) & \text { if } q_{1} \in F_{1} \text { and } a_{1}=0 \\ \left(\delta_{1}\left(q_{1}, a_{1}\right), q_{2}\right) & \text { if } q_{2} \in F_{2} \text { and } a_{2}=0\end{cases}$
where we can assume that the final states have no outgoing transitions but those labelled by the input letter 0 and leading to a common final state with a single loop labelled with 0 . This is done to account for Büchi's interpretation of regular expressions (Section 6.1). With this modification, the language accepted by $A_{3}$ is that represented by $R_{3}$. This completes the case of conjunctions.

2) For disjunction, we assume that we are given two formulae

$$
\begin{gathered}
\varphi_{1}=\exists \bar{k}_{1}^{\prime}, \bar{t}_{1}^{\prime}, \bar{x}_{1}^{\prime} \cdot F_{1}\left(\bar{S}_{1}, \bar{k}_{1}, \bar{k}_{1}^{\prime}\right) \wedge\left(\bar{t}_{1}, \bar{t}_{1}^{\prime}\right) \in R\left(l_{11}, \ldots, l_{s 1}\right) \wedge \\
\bigwedge_{i 1} S_{i 1}=\left\{n \in \mathbb{N} \mid \psi_{i 1}\left(\bar{t}_{1}(n), \bar{t}_{1}^{\prime}(n)\right)\right\} \wedge \\
\bigwedge_{j 1} S_{j 1}=\left\{n \in \mathbb{N} \mid \varphi_{j 1}\left(\bar{x}_{1}(n), \bar{x}_{1}^{\prime}(n), \bar{c}_{1}\right)\right\} \\
\varphi_{2}=\exists \bar{k}_{2}^{\prime},,_{2}^{\prime}, \bar{x}_{2}^{\prime} \cdot F_{2}\left(\bar{S}_{2}, \bar{k}_{2}, \bar{k}_{2}^{\prime}\right) \wedge\left(\bar{t}_{2}, \bar{t}_{2}^{\prime}\right) \in R\left(l_{12}, \ldots, l_{t 2}\right) \wedge \\
\bigwedge_{i 2} S_{i 2}=\left\{n \in \mathbb{N} \mid \psi_{i 2}\left(\bar{t}_{2}(n), \bar{t}_{2}^{\prime}(n)\right)\right\} \wedge \\
\bigwedge_{j 2} S_{j 2}=\left\{n \in \mathbb{N} \mid \varphi_{j 2}\left(\bar{x}_{2}(n), \bar{x}_{2}^{\prime}(n), \bar{c}_{2}\right)\right\}
\end{gathered}
$$

then their disjunction is

$$
\begin{aligned}
& \varphi_{1} \vee \varphi_{2}=\exists \bar{k}_{1}^{\prime}, \bar{t}_{1}^{\prime}, \bar{x}_{1}^{\prime} \cdot \exists \bar{k}_{2}^{\prime}, \bar{t}_{2}^{\prime}, \bar{x}_{2}^{\prime} . \\
&\left(F_{1}\left(\bar{S}_{1}, \bar{k}_{1}, \bar{k}_{1}^{\prime}\right) \wedge\left(\bar{t}_{1}, \bar{t}_{1}^{\prime}\right) \in R\left(l_{11}, \ldots, l_{s 1}\right) \wedge\right. \\
& \bigwedge_{i 1} S_{i 1}=\left\{n \in \mathbb{N} \mid \psi_{i 1}\left(\bar{t}_{1}(n), \bar{t}_{1}^{\prime}(n)\right)\right\} \wedge \\
&\left.\bigwedge_{j 1} S_{j 1}=\left\{n \in \mathbb{N} \mid \varphi_{j 1}\left(\bar{x}_{1}(n), \bar{x}_{1}^{\prime}(n), \bar{c}_{1}\right)\right\}\right) \vee \\
&\left(F_{2}\left(\bar{S}_{2}, \bar{k}_{2}, \bar{k}_{2}^{\prime}\right) \wedge\left(\bar{t}_{2}, \bar{t}_{2}^{\prime}\right) \in R\left(l_{12}, \ldots, l_{t 2}\right) \wedge\right. \\
& \bigwedge_{i 2} S_{i 2}=\left\{n \in \mathbb{N} \mid \psi_{i 2}\left(\bar{t}_{2}(n), \bar{t}_{2}^{\prime}(n)\right)\right\} \wedge \\
&\left.\bigwedge_{j 2} S_{j 2}=\left\{n \in \mathbb{N} \mid \varphi_{j 2}\left(\bar{x}_{2}(n), \bar{x}_{2}^{\prime}(n), \bar{c}_{2}\right)\right\}\right)
\end{aligned}
$$

We claim that $\varphi$ is equisatisfiable to

$$
\begin{aligned}
& \varphi^{\prime}=\exists \bar{k}_{1}^{\prime}, \bar{t}_{1}^{\prime}, \bar{x}_{1}^{\prime} \cdot \exists \bar{k}_{2}^{\prime}, \bar{t}_{2}^{\prime}, \bar{x}_{2}^{\prime} . \\
&\left(F_{1}\left(\bar{S}_{1}, \bar{k}_{1}, \bar{k}_{1}^{\prime}\right) \vee F_{2}\left(\bar{S}_{2}, \bar{k}_{2}, \bar{k}_{2}^{\prime}\right)\right) \wedge\left(\bar{t}_{1}, \bar{t}_{1}^{\prime}, \bar{t}_{2}, \bar{t}_{2}^{\prime}\right) \in R_{3} \wedge \\
& \bigwedge_{i 1} S_{i 1}=\left\{n \in \mathbb{N} \mid \psi_{i 1}\left(\bar{t}_{1}(n), \bar{t}_{1}^{\prime}(n)\right)\right\} \wedge \\
&\left.\bigwedge_{j 1} S_{j 1}=\left\{n \in \mathbb{N} \mid \varphi_{j 1}\left(\bar{x}_{1}(n), \bar{x}_{1}^{\prime}(n), \bar{c}_{1}\right)\right\}\right) \wedge \\
& \bigwedge_{i 2} S_{i 2}=\left\{n \in \mathbb{N} \mid \psi_{i 2}\left(\bar{t}_{2}(n), \bar{t}_{2}^{\prime}(n)\right)\right\} \wedge \\
& \bigwedge_{j 2} S_{j 2}=\left\{n \in \mathbb{N} \mid \varphi_{j 2}\left(\bar{x}_{2}(n), \bar{x}_{2}^{\prime}(n), \bar{c}_{2}\right)\right\}
\end{aligned}
$$

where $R_{3}$ is the regular expression formula built in part 1).
If $\mathcal{M} \models \varphi_{1} \vee \varphi_{2}$ then $\mathcal{M} \models \varphi_{i}$ for some $i \in\{1,2\}$. Without loss in generality, let us assume that $i=1$. We redefine the table variables $\left(\bar{t}_{j 2}, \bar{t}_{j 2}^{\prime}\right)$ in such a way that $R_{3}$ is satisfied. This can be done because the regular expressions of Figure 7 are always satisfiable (i.e. there is no regular expression representing the empty set of tables). Call the new model $\mathcal{M}^{\prime}$. Then $\mathcal{M}^{\prime} \models \varphi^{\prime}$ since $S_{i 2}$ and $S_{j 2}$ do not introduce variables occurring in $F_{1}$.

Conversely, if $\mathcal{M} \models \varphi^{\prime}$ then

$$
\begin{aligned}
& \mathcal{M} \models \exists \bar{k}_{1}^{\prime}, \bar{t}_{1}^{\prime}, \bar{x}_{1}^{\prime} \cdot \exists \bar{k}_{2}^{\prime}, \bar{t}_{2}^{\prime}, \bar{x}_{2}^{\prime} . \\
& F_{i}\left(\bar{S}_{i}, \bar{k}_{i}, \bar{k}_{i}^{\prime}\right) \wedge\left(\bar{t}_{1}, \bar{t}_{1}^{\prime}, \bar{t}_{2}, \bar{t}_{2}^{\prime}\right) \in R_{3} \wedge \\
& \bigwedge_{i 1} S_{i 1}=\left\{n \in \mathbb{N} \mid \psi_{i 1}\left(\bar{t}_{1}(n), \bar{t}_{1}^{\prime}(n)\right)\right\} \wedge \\
& \bigwedge_{j 1} S_{j 1}=\left\{n \in \mathbb{N} \mid \varphi_{j 1}\left(\bar{x}_{1}(n), \bar{x}_{1}^{\prime}(n), \bar{c}_{1}\right)\right\} \vee \\
& \bigwedge_{i 2} S_{i 2}=\left\{n \in \mathbb{N} \mid \psi_{i 2}\left(\bar{t}_{2}(n), \bar{t}_{2}^{\prime}(n)\right)\right\} \wedge \\
&\left.\bigwedge_{j 2} S_{j 2}=\left\{n \in \mathbb{N} \mid \varphi_{j 2}\left(\bar{x}_{2}(n),,_{2}^{\prime}(n), \bar{c}_{2}\right)\right\}\right)
\end{aligned}
$$

for some $i \in\{1,2\}$. From this, it follows that $\mathcal{M} \vDash \varphi_{1} \vee \varphi_{2}$.
This concludes the case of disjunctions.
3) For negation, we assume that we are given a formula of the power fragment without existential quantifiers, i.e. of the form

$$
\begin{aligned}
F(\bar{S}, \bar{k}) & \wedge \bar{t} \in R\left(l_{1}, \ldots, l_{s}\right) \\
& \wedge \bigwedge_{i} S_{i}=\left\{n \in \mathbb{N} \mid \psi_{i}(\bar{t}(n))\right\} \\
& \wedge \bigwedge_{j} S_{j}=\left\{n \in \mathbb{N} \mid \varphi_{j}(\bar{x}(n), \bar{c})\right\}
\end{aligned}
$$

Negating it, we obtain a disjunction of three terms

$$
\begin{aligned}
\neg F(\bar{S}, \bar{k}) & \vee \bar{t} \in R\left(l_{1}, \ldots, l_{s}\right) \vee \\
& \neg\left(\bigwedge_{i} S_{i}=\left\{n \in \mathbb{N} \mid \psi_{i}(\bar{t}(n))\right\} \wedge \bigwedge_{j} S_{j}=\left\{n \in \mathbb{N} \mid \varphi_{j}(\bar{x}(n), \bar{c})\right\}\right)
\end{aligned}
$$

We know there exists a regular expression $R^{c}$ for the complement of $R$ and thus we obtain an equivalent formula ${ }^{3}$

$$
\begin{aligned}
\neg F(\bar{S}, \bar{k}) & \vee \bar{t} \in R^{c}\left(l_{1}, \ldots, l_{s}\right) \vee \\
& \neg\left(\bigwedge_{i} S_{i}=\left\{n \in \mathbb{N} \mid \psi_{i}(\bar{t}(n))\right\} \wedge \bigwedge_{j} S_{j}=\left\{n \in \mathbb{N} \mid \varphi_{j}(\bar{x}(n), \bar{c})\right\}\right)
\end{aligned}
$$

[^3]$\bar{t} \in R^{c}\left(l_{1}, \ldots, l_{s}\right)$ is equivalent to the condition $\left(\left|p_{l_{1}}\right|, \ldots,\left|p_{l_{s}}\right|\right) \in \operatorname{Parikh}\left(R^{c}\right)$. Thus we have
\[

$$
\begin{aligned}
\neg F(\bar{S}, \bar{k}) & \vee\left(\left|p_{l_{1}}\right|, \ldots,\left|p_{l_{s}}\right|\right) \in \operatorname{Parikh}\left(R^{c}\right) \vee \\
& \neg\left(\bigwedge_{i} S_{i}=\left\{n \in \mathbb{N} \mid \psi_{i}(\bar{t}(n))\right\} \wedge \bigwedge_{j} S_{j}=\left\{n \in \mathbb{N} \mid \varphi_{j}(\bar{x}(n), \bar{c})\right\}\right)
\end{aligned}
$$
\]

Finally, we introduce variables $D_{i}, D_{i}^{\prime}$

$$
\begin{aligned}
& \left(\neg F \vee\left(\left|p_{l_{1}}\right|, \ldots,\left|p_{l_{p}}\right|\right) \in \operatorname{Parikh}\left(R^{c}\right) \vee \bigvee S_{i} \neq D_{i} \vee \bigvee S_{i}^{\prime} \neq D_{i}^{\prime}\right) \wedge \\
& \wedge \bigwedge D_{i}=\left\{n \in \mathbb{N} \mid \psi_{i}(\bar{t}(n))\right\} \wedge \bigwedge D_{i}^{\prime}=\left\{n \in \mathbb{N} \mid \varphi_{j}(\bar{x}(n), \bar{c})\right\}
\end{aligned}
$$

The resulting formula is a conjunction of a QFBAPA formula and an interpreted formula, which is also in the fragment.

### 7.2. Closure under an Imperative Commands Language

In this section, we introduce a simple imperative programming language that can be used as a target to translate programs with annotations but without loops nor procedure calls into formulae of our logical fragment. We show that the relations induced by the commands of this programming language on the states of the program are definable by formulas of the fragment.

The imperative programming language is shown in Figure 10. The command Assume allows to specify conditions that need to hold at the given point in the program. The expression $a[i]:=v$ updates the array pointed by $a$ at position $i$ with value $v$. The expression $i+=1$ updates the value of integer variable $i$ by one. ${ }^{4} \quad P_{1} ; P_{2}$ denotes the sequential composition of $P_{1}$ and $P_{2} . P_{1} \square P_{2}$ executes either $P_{1}$ or $P_{2}$ non-deterministically.

$$
P::=\operatorname{Assume}(\varphi)|a[i]:=e| i+=1|e=a[i]| P_{1} ; P_{2} \mid P_{1} \square P_{2}
$$

Figure 10: Simple loop-free programming language.

[^4]Proposition 32. Let the state of our programs be represented as a tuple $(\bar{i}, \bar{v}, \bar{a})$ where $\bar{i}$ is a tuple of integer variables, $\bar{e}$ is a tuple of element variables and $\bar{a}$ is a tuple of power structure variables. We denote by $(\bar{i}, \bar{e}, \bar{a})$ the input of the program and by $\left(\bar{i}^{\prime}, \bar{e}^{\prime}, \bar{a}^{\prime}\right)$ its output. Then we show that the relations $R(c)$ between input and output variables induced by the commands $c$ of the programming language of Figure 10 can be written as

$$
R(c)=\left\{\left(\bar{i}, \bar{e}, \bar{a}, \bar{i}^{\prime}, \bar{e}^{\prime}, \bar{a}^{\prime}\right) \mid \psi\left(\bar{i}, \bar{e}, \bar{a}, \bar{i}^{\prime}, \bar{e}^{\prime}, \bar{a}^{\prime}\right)\right\}
$$

where $\psi$ is a formula of the power fragment.
Proof. We prove it by structural induction on the shape of the formula.

- Closure under Assume statement.

The formula associated with $\operatorname{Assume}(\varphi)$ is

$$
R(\operatorname{Assume}(\varphi))=\left\{\left(\bar{i}, \bar{e}, \bar{a}, \bar{i}^{\prime}, \bar{e}^{\prime}, \bar{a}^{\prime}\right) \mid \varphi \wedge \bigwedge_{v \in V} v^{\prime}=v\right\}
$$

that is, $\varphi$ holds and all variables remain unchanged after execution of the statement.

Since the fragment is closed under conjunctions, we only need to check that $v^{\prime}=v$ is in the fragment for each kind of variable.
For variables $e$ of the element theory this can be encoded as

$$
S \neq \emptyset \wedge S=\left\{i \in I \mid e=e^{\prime}\right\}
$$

For variables $i$ of the index theory this can be encoded as

$$
i=i^{\prime}
$$

For variables $a$ of the power structure theory this can be encoded as

$$
S=I \wedge S=\left\{i \in I \mid a(i)=a^{\prime}(i)\right\}
$$

- Closure under memory write statement.

The formula associated with $a[j]=e$ is

$$
R(a[j]=e)=\left\{\begin{array}{l|l}
\left(\bar{i}, \bar{e}, \bar{a}, \bar{i}^{\prime}, \bar{e}^{\prime}, \bar{a}^{\prime}\right) & \begin{array}{l}
\left\{i \in I ; a(i)=a^{\prime}(i)\right\}=I \backslash\left\{j^{\prime}\right\} \wedge \\
\{i \in I \mid a(i)=e\} \supseteq\left\{j^{\prime}\right\} \wedge \\
j^{\prime}=j \wedge e^{\prime}=e
\end{array}
\end{array}\right\}
$$

encoded in our fragment the right-hand side reads

$$
\begin{aligned}
& \exists t_{1}, \ldots, t_{3} \cdot j^{\prime}=j \wedge e^{\prime}=e \wedge S_{1}=I \backslash S_{4} \wedge S_{2} \supseteq S_{4} \wedge \\
& \quad\left|S_{1}\right|=j \wedge\left|S_{1} \cap S_{2} \cap S_{3}\right|=1 \wedge\left|S_{1} \cap S_{3}\right|=1 \wedge \\
& \quad t_{1} \in 1^{*} \wedge t_{2} \in 0^{*} 1 \wedge t_{3} \in 0^{*} 11 \\
& \quad S_{1}=\left\{i \in I \mid a(i)=a^{\prime}(i)\right\} \wedge S_{2}=\{i \in I \mid a(i)=e\} \wedge \\
& \quad S_{3}=\left\{n \in \mathbb{N} \mid t_{1}(n)\right\} \wedge S_{4}=\left\{n \in \mathbb{N} \mid t_{2}(n)\right\} \wedge \\
& \\
& S_{5}=\left\{n \in \mathbb{N} \mid t_{3}(n)\right\}
\end{aligned}
$$

where $S_{4}$ is the singleton set $\left\{j^{\prime}\right\}$ due to the conditions imposed. For a visual explanation, see Figure 11.

- Closure under index increment.

The formula associated with $i=i+1$ is

$$
R(i=i+1)=\left\{\left(\bar{i}, \bar{e}, \bar{a}, \bar{i}^{\prime}, \bar{e}^{\prime}, \bar{a}^{\prime}\right) \mid i^{\prime}=i+1\right\}
$$

- Closure under memory read.

The formula associated with $e=a[j]$ is
$R(e=a[j])=\left\{\left(\bar{i}, \bar{e}, \bar{a}, \bar{i}^{\prime}, \bar{e}^{\prime}, \bar{a}^{\prime}\right) \mid\{j\} \subseteq\left\{i \in I \mid a(i)=e^{\prime}\right\} \wedge j=j^{\prime} \wedge a=a^{\prime}\right\}$
Encoded in our fragment the right-hand side reads

$$
\begin{aligned}
& \exists t_{1}, \ldots, t_{3} \cdot j^{\prime}=j \wedge S_{1}=I \wedge S_{2} \supseteq S_{4} \wedge \\
& \quad\left|S_{1}\right|=j \wedge\left|S_{1} \cap S_{2} \cap S_{3}\right|=1 \wedge\left|S_{1} \cap S_{3}\right|=1 \wedge \\
& \quad t_{1} \in 1^{*} \wedge t_{2} \in 0^{*} 1 \wedge t_{3} \in 0^{*} 11 \\
& \\
& S_{1}=\left\{i \in I \mid a(i)=a^{\prime}(i)\right\} \wedge S_{2}=\left\{i \in I \mid a(i)=e^{\prime}\right\} \wedge \\
& \\
& S_{3}=\left\{n \in \mathbb{N} \mid t_{1}(n)\right\} \wedge S_{4}=\left\{n \in \mathbb{N} \mid t_{2}(n)\right\} \wedge \\
& \\
& S_{5}=\left\{n \in \mathbb{N} \mid t_{3}(n)\right\}
\end{aligned}
$$

- Closure under sequential composition.

The formula associated with $P_{1} ; P_{2}$ is

$$
R\left(P_{1} ; P_{2}\right)=\left\{\left(\bar{x}, \bar{x}^{\prime}\right) \mid \exists \bar{x}^{\prime \prime} \cdot F\left(P_{1}\right)\left[\bar{x}^{\prime}:=\bar{x}^{\prime \prime}\right] \wedge F\left(P_{2}\right)\left[\bar{x}:=\bar{x}^{\prime \prime}\right]\right\}
$$

i.e. first $P_{1}$ is executed and its output is renamed to an intermediate value $\bar{x}^{\prime \prime}$ and then $P_{2}$ is executed and its input value is renamed to $\bar{x}^{\prime \prime}$. Note that we have abbreviated the tuple $(\bar{i}, \bar{e}, \bar{a})$ to $\bar{x}$ and $\left(\bar{i}^{\prime}, \bar{e}^{\prime}, \bar{a}^{\prime}\right)$ to $\bar{x}^{\prime \prime}$. To prove this is again a formula in the fragment we will first rename the occurrences of $\bar{x}^{\prime \prime}$ in each conjunct by adding new set variables

- for each element variable in $\bar{x}^{\prime \prime}, x$, we create two constant element variables $c_{1}, c_{2}$ and impose $S=\left\{i \in I \mid c_{1}=c_{2}\right\} \wedge S=I$. Then we replace $x$ by $c_{1}$ in the first substitution and by $c_{2}$ in the second substitution.
- for each index variable in $\bar{x}^{\prime \prime}$, $i$, we create two index variables $i_{1}, i_{2}$ and impose $i_{1}=i_{2}$. Then we replace $x$ by $i_{1}$ in the first substitution and by $i_{2}$ in the second substitution.
- for each power variable in $\bar{x}^{\prime \prime}, a$, we create two power variables $a_{1}, a_{2}$ and impose $S=\left\{i \in I \mid a_{1}(i)=a_{2}(i)\right\} \wedge S=I$. Then we replace $a$ by $a_{1}$ in the first substitution and by $a_{2}$ in the second substitution.

This allows to rewrite the formula $\exists \bar{x}^{\prime \prime} \cdot F\left(P_{1}\right)\left[\bar{x}^{\prime}:=\bar{x}^{\prime \prime}\right] \wedge F\left(P_{2}\right)\left[\bar{x}:=\bar{x}^{\prime \prime}\right]$ as

$$
\exists \bar{x}_{1}^{\prime \prime}, \bar{x}_{2}^{\prime \prime} \cdot F\left(P_{1}\right)\left[\bar{x}^{\prime}:=\bar{x}_{1}^{\prime \prime}\right] \wedge F\left(P_{2}\right)\left[\bar{x}:=\bar{x}_{2}^{\prime \prime}\right] \wedge \varphi
$$

where $\varphi$ is a QFBAPA formula mentioning only variables in $\bar{x}_{1}^{\prime \prime}, \bar{x}_{2}^{\prime \prime}$.
By induction hypothesis, $F\left(P_{1}\right)\left[\bar{x}^{\prime}:=\bar{x}_{1}^{\prime \prime}\right]$ and $F\left(P_{2}\right)\left[\bar{x}:=\bar{x}_{2}^{\prime \prime}\right]$ are formulae of the fragment. By the closure properties, $F\left(P_{1}\right)\left[\bar{x}^{\prime}:=\right.$ $\left.\bar{x}_{1}^{\prime \prime}\right] \wedge F\left(P_{2}\right)\left[\bar{x}:=\bar{x}_{2}^{\prime \prime}\right]$ is in the fragment too. Let us call this formula $\psi$ so that the resulting formula is $\exists \bar{x}_{1}^{\prime \prime}, \bar{x}_{2}^{\prime \prime} . \psi \wedge \varphi$. Now, $\psi$ and $\varphi$ share free variables but it is straightforward that their conjunction is again a formula of the fragment since $\varphi$ only adds terms to the QFBAPA part and to the set interpretations.

Thus, we can rewrite $\exists \bar{x}_{1}^{\prime \prime}, \bar{x}_{2}^{\prime \prime} \cdot \psi \wedge \varphi$ into

$$
\begin{aligned}
\exists \bar{k}_{1}^{\prime \prime}, \bar{t}_{1}^{\prime \prime}, \bar{a}_{1}^{\prime \prime}, \bar{k}_{2}^{\prime \prime}, \bar{t}_{2}^{\prime \prime}, \bar{a}_{2}^{\prime \prime} \cdot \exists \bar{k}^{\prime}, \bar{t}^{\prime}, \bar{a}^{\prime} & \\
F\left(\bar{S}, \bar{k}, \bar{k}^{\prime}\right) & \wedge\left(\bar{t}, \bar{t}^{\prime}\right) \in R\left(l_{1}, \ldots, l_{s}\right) \\
& \wedge \bigwedge_{i} S_{i}=\left\{n \in \mathbb{N} \mid \psi_{i}\left(\bar{t}(n), \bar{t}^{\prime}(n)\right)\right\} \\
& \wedge \bigwedge_{j} S_{j}=\left\{n \in \mathbb{N} \mid \varphi_{j}\left(\bar{a}(n), \bar{a}^{\prime}(n), \bar{c}\right)\right\}
\end{aligned}
$$

which is again in the fragment.

- Closure under non-deterministic composition.

The formula associated with $P_{1} \square P_{2}$ is

$$
R\left(P_{1} \square P_{2}\right)=\left\{\left(\bar{i}, \bar{e}, \bar{a}, \bar{i}^{\prime}, \bar{e}^{\prime}, \bar{a}^{\prime}\right) \mid F\left(P_{1}\right) \vee F\left(P_{2}\right)\right\}
$$

where $F(P)$ is the formula associated to program $P$. Since the fragment is closed under disjunction, the resulting formula is in the fragment too.

| $S_{3}$ | 0 | $\cdots$ | 0 | 1 |  |
| :--- | :--- | :--- | :--- | :--- | :--- |
| $S_{4}$ | 0 | $\cdots$ | 0 | 1 |  |
| $S_{5}$ | 0 | $\cdots$ | 0 | 1 | 1 |
|  |  |  |  |  |  |

Figure 11: Table configuration imposed by the formula corresponding to a memory write.

### 7.3. Closure under Hoare Logic Triples

Given a Hoare triple $\{P\} c\{Q\}$, where $P, Q$ are formulae of the power fragment without existential quantifiers and $c$ is a command of the simple loop-free programming language of Figure 10, we know that its semantics corresponds to the formula $\left(P \wedge F_{c}\right) \Longrightarrow Q$ where $F_{c}$ is the formula derived in the previous section for command $c$. Equivalently, we need to check validity of $\neg\left(P \wedge F_{c}\right) \vee Q$. This is the same as checking unsatisfiability of $P \wedge F_{c} \wedge \neg Q$. Since we assume that $Q$ is in the power fragment without quantifiers, $\neg Q$ is again a formula in the fragment by Proposition 31. Since the power fragment is closed under conjunction, it follows that $\{P\} c\{Q\}$ can be written in it.

## 8. Classification of Array Theories

[41] studied first a theory of arrays with an extensionality axiom.
[15] extended this by adding pointwise functions and relations to the supported language. [15] showed how to encode sets and multisets in the language of CAL but noted that cardinalities of those could not be expressed. In [35], we showed that CAL constraints can be encoded in QFBAPAI with a log-quadratic increase in the formula size. Thus QFBAPAI can be used to encode CAL constraints as well as cardinalities over sets and multisets.
[1] studied a fragment of the theory of arrays that supported cardinalities of interpreted sets, which they called simple flat subfragment. A similar language was studied in [21]. These works were driven by applications in verification of distributed systems. This may be the reason why the logic used in the interpreted sets is Presburger arithmetic. Our logic QFBAPAI admits any NP-decidable theory instead. None of these works seem to have recognized the connection with [15].

The model-theoretic approach to array theories that we propose, had the virtue of suggesting the further generalisation of including an ordering relation in the index set since this is done for instance in [17]. Inspired by the result of [14], we formulated the fragment RegExp-QFBAPA-Power which still has the advantage of being independent of the component theory.


Figure 12: Treated theories of arrays sorted by expressivity.

## 9. Conclusion

We have obtained results on the computational complexity of several logical theories that are relevant in software verification. In order to do so, we used a basic insight on the decomposition of the theory of power structures in terms of the index theory and the theory of the elements, as well as the combination of theories through sets and cardinalities.

Our approach allows to make definability arguments so that it is clear what can be expressed with the proposed specifications. For instance, in RegExp-QFBAPA-Power, the definable relations on the indices of the power structure are precisely those definable in weak monadic second-order arithmetic of one successor. It also provides means for classifying existing syntactic theories. As shown in Figure 10, QFBAPAI subsumes at least four existing theories in the literature.

There are two interesting directions of future work. First, it would be interesting to investigate the trade-offs between the expressivity offered by RegExp-QFBAPA-Power and its perfomance in practice. This could have an impact on SMT solvers such as Z3 or CVC which implement the CAL subfragment of RegExp-QFBAPA-Power. Experimental work could be combined with theoretical analysis as the one suggested in [39]. Second, the semantic methodology of this paper could be applied to other array theories such as the classical [8]. It would be also interesting to apply the semantic methodology to other domains in software verification. This has been advocated by some of the leading experts in the field [33, Lecture 19].

## Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

## Data availability

No data was used for the research described in the article.

## 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.
[2] Francesco Alberti, Silvio Ghilardi, and Natasha Sharygina. Booster: An Acceleration-Based Verification Framework for Array Programs. In

Franck Cassez and Jean-François Raskin, editors, Automated Technology for Verification and Analysis, Lecture Notes in Computer Science, pages 18-23, Cham, 2014. Springer International Publishing.
[3] Francesco Alberti, Silvio Ghilardi, and Natasha Sharygina. Decision Procedures for Flat Array Properties. Journal of Automated Reasoning, 54(4):327-352, April 2015.
[4] Sanjeev Arora and Boaz Barak. Computational Complexity: A Modern Approach. Cambridge University Press, Cambridge ; New York, 2009.
[5] Haniel Barbosa, Clark Barrett, Martin Brain, Gereon Kremer, Hanna Lachnitt, Makai Mann, Abdalrhman Mohamed, Mudathir Mohamed, Aina Niemetz, Andres Nötzli, Alex Ozdemir, Mathias Preiner, Andrew Reynolds, Ying Sheng, Cesare Tinelli, and Yoni Zohar. cvc5: A Versatile and Industrial-Strength SMT Solver. In Dana Fisman and Grigore Rosu, editors, Tools and Algorithms for the Construction and Analysis of Systems, Lecture Notes in Computer Science, pages 415-442, Cham, 2022. Springer International Publishing.
[6] J. Barwise and S. Feferman, editors. Model-Theoretic Logics. Perspectives in Logic. Cambridge University Press, Cambridge, 2017.
[7] Aaron Bradley and Zohar Manna. Calculus of computation: decision procedures with applications to verification. Springer, Berlin, 2007. OCLC: ocn190764844.
[8] Aaron R. Bradley, Zohar Manna, and Henny B. Sipma. What's Decidable About Arrays? In E. Allen Emerson and Kedar S. Namjoshi, editors, Verification, Model Checking, and Abstract Interpretation, Lecture Notes in Computer Science, pages 427-442, Berlin, Heidelberg, 2006. Springer.
[9] Stanley N. Burris. Logic for Mathematics and Computer Science. Prentice Hall, Upper Saddle River, N.J, 1st edition edition, August 1997.
[10] J. R. Büchi and D. Siefkes. Decidable Theories: Vol. 2: The Monadic Second Order Theory of All Countable Ordinals. Springer, Berlin, 1973.
[11] J. Richard Büchi. Weak Second-Order Arithmetic and Finite Automata. Mathematical Logic Quarterly, 6(1-6):66-92, 1960.
[12] Chang Chia-Hsiang and Robert Paige. From regular expressions to DFA's using compressed NFA's. Theoretical Computer Science, 178(1):1-36, May 1997.
[13] Maximiliano Cristiá and Gianfranco Rossi. Integrating Cardinality Constraints into Constraint Logic Programming with Sets. Theory and Practice of Logic Programming, pages 1-33, November 2021. Publisher: Cambridge University Press.
[14] Przemysław Daca, Thomas A. Henzinger, and Andrey Kupriyanov. Array Folds Logic. In Swarat Chaudhuri and Azadeh Farzan, editors, Computer Aided Verification, Lecture Notes in Computer Science, pages 230-248, Cham, 2016. Springer International Publishing.
[15] 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.
[16] Leonardo de Moura and Nikolaj Bjørner. Z3: An Efficient SMT Solver. In Tools and Algorithms for the Construction and Analysis of Systems, volume 4963, pages 337-340, Berlin, Heidelberg, 2008. Springer Berlin Heidelberg. Series Title: Lecture Notes in Computer Science.
[17] S. Feferman and R. Vaught. The first order properties of products of algebraic systems. Fundamenta Mathematicae, 47(1):57-103, 1959.
[18] Jeanne Ferrante and Charles Rackoff. The Computational Complexity of Logical Theories. Springer Berlin, Heidelberg, 1979.
[19] Robert W. Floyd. Assigning Meanings to Programs. In Timothy R. Colburn, James H. Fetzer, and Terry L. Rankin, editors, Program Verification: Fundamental Issues in Computer Science, Studies in Cognitive Systems, pages 65-81. Springer Netherlands, Dordrecht, 1993.
[20] Silvio Ghilardi, Enrica Nicolini, Silvio Ranise, and Daniele Zucchelli. Decision procedures for extensions of the theory of arrays. Annals of Mathematics and Artificial Intelligence, 50(3):231-254, August 2007.
[21] 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.
[22] Peter Habermehl, Radu Iosif, and Tomáš Vojnar. What Else Is Decidable about Integer Arrays? In Foundations of Software Science and Computational Structures, Lecture Notes in Computer Science, pages 474-489, Berlin, Heidelberg, 2008. Springer.
[23] Wilfrid Hodges. Model Theory. Encyclopedia of Mathematics and its Applications. Cambridge University Press, Cambridge, 1993.
[24] Gérard Huet. Initiation à la Logique Mathématique, 1986.
[25] James Cornelius King. A program verifier. PhD thesis, Carnegie-Mellon University, Pittsburgh Pennsylvania USA, September 1969. Section: Technical Reports.
[26] S. C. Kleene. Representation of Events in Nerve Nets and Finite Automata. In Representation of Events in Nerve Nets and Finite Automata, pages 3-42. Princeton University Press, 1956.
[27] Daniel Kroening and Ofer Strichman. Decision Procedures. Texts in Theoretical Computer Science. An EATCS Series. Springer Berlin Heidelberg, Berlin, Heidelberg, 2 edition, 2016.
[28] Viktor Kuncak, Huu Hai Nguyen, and Martin Rinard. Deciding Boolean Algebra with Presburger Arithmetic. Journal of Automated Reasoning, 36(3):213-239, April 2006.
[29] Viktor Kuncak and Martin Rinard. Towards Efficient Satisfiability Checking for Boolean Algebra with Presburger Arithmetic. In Frank Pfenning, editor, Automated Deduction - CADE-21, Lecture Notes in Computer Science, pages 215-230, Berlin, Heidelberg, 2007. Springer.
[30] Rustan Leino. Dafny: An Automatic Program Verifier for Functional Correctness. In Logic for Programming, Artificial Intelligence, and Reasoning, pages 348-370, April 2010.
[31] J. A. Makowsky. Algorithmic uses of the Feferman-Vaught Theorem. Annals of Pure and Applied Logic, 126(1):159-213, April 2004.
[32] J. McCarthy. Towards a Mathematical Science of Computation. In Timothy R. Colburn, James H. Fetzer, and Terry L. Rankin, editors, Program Verification: Fundamental Issues in Computer Science, Studies in Cognitive Systems, pages 35-56. Springer Netherlands, Dordrecht, 1993.
[33] José Meseguer. Lecture Notes on Topics in Automated Reasoning. 2017.
[34] Andrzej Mostowski. On direct products of theories. The Journal of Symbolic Logic, 17(1):1-31, March 1952. Publisher: Cambridge University Press.
[35] Rodrigo Raya, Jad Hamza, and Viktor Kunčak. NP Decision Procedure for Monomial and Linear Integer Constraints. Technical Report arXiv:2208.02713, arXiv, October 2022. arXiv:2208.02713 [cs] type: article.
[36] Helmut Seidl, Thomas Schwentick, Anca Muscholl, and Peter Habermehl. Counting in Trees for Free. In Josep Díaz, Juhani Karhumäki, Arto Lepistö, and Donald Sannella, editors, Automata, Languages and Programming, Lecture Notes in Computer Science, pages 1136-1149, Berlin, Heidelberg, 2004. Springer.
[37] Joseph Shoenfield. Mathematical logic. Addison-Wesley Publishing Company, Inc., 1967.
[38] Michael Sipser. Introduction to the Theory of Computation. Cengage Learning, 3 edition, June 2012.
[39] R. E. Stearns and H.B. Hunt III. On the Complexity of the Satisfiability Problem and the Structure of NP. Technical Report 86-21, State University of New York at Albany, Computer Science Department, October 1986.
[40] 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.
[41] A. Stump, C.W. Barrett, D.L. Dill, and J. Levitt. A decision procedure for an extensional theory of arrays. In Proceedings 16th Annual IEEE

Symposium on Logic in Computer Science, pages 29-37, Boston, MA, USA, 2001. IEEE Comput. Soc.
[42] Nikhil Swamy, Juan Chen, and Ben Livshits. Verifying Higher-order Programs with the Dijkstra Monad. In Programming Language Design and Implementation, June 2013.
[43] A.M. Turing. Checking a large routine. In Report of a Conference on High Speed Automatic Calculating Machines, pages 67-69. Univ. Math. Lab., Cambridge, 1949.
[44] Niki Vazou, Eric L. Seidel, Ranjit Jhala, Dimitrios Vytiniotis, and Simon Peyton-Jones. Refinement types for Haskell. In Proceedings of the 19th ACM SIGPLAN international conference on Functional programming, ICFP '14, pages 269-282, New York, NY, USA, August 2014. Association for Computing Machinery.
[45] Nicolas Charles Yves Voirol. Verified Functional Programming. PhD thesis, EPFL, Lausanne, 2019.
[46] Thomas Wies, Ruzica Piskac, and Viktor Kuncak. Combining Theories with Shared Set Operations. In Silvio Ghilardi and Roberto Sebastiani, editors, Frontiers of Combining Systems, Lecture Notes in Computer Science, pages 366-382, Berlin, Heidelberg, 2009. Springer.
[47] Sheng Yu. Regular Languages. Handbook of Formal Languages: Volume 1 Word, Language, Grammar, pages 41-110, 1997.


[^0]:    *Corresponding author.
    Email-addresses: rodrigo.raya@epfl.ch, viktor.kuncak@epfl.ch

[^1]:    ${ }^{1}$ This decomposition can be seen as an "algorithmic technique" [31].

[^2]:    ${ }^{2}$ The observation of Büchi has been generalised. Some terminology uses the suffix "on words" to refer to this kind of interpretation of logical theories. It is also common to find the expression "MSOL on words" in this context since the proof of Büchi equating regular languages with WS1S is also applicable to the formalism of monadic second-order logic.

[^3]:    ${ }^{3}$ In general, it may be computationally difficult to compute the complement of a regular expression. Alternatively, one could ask the user of the verification system to provide a deterministic finite automata instead.

[^4]:    ${ }^{4}$ The proof in this section still holds in the case of having an arbitrary quantifier-free Presburger arithmetic formula on the right-hand side.

