Reasoning with Sets and Sums of Sets

Markus Bender
Universität Koblenz-Landau, Koblenz, Germany
mbender@uni-koblenz.de

Abstract

In this paper we introduce a decision procedure for the combined theory of sets, arithmetic and sums of sets, that can be used as abstraction for problems in verification.

This theory and the developed decision procedures is also the first step in our path to develop decision procedures for more theories that can be used as abstraction in more situations, for example with the additional bridging functions min, max or measures.

1 Introduction

Verification of real world systems often incorporates decision procedures for different data structures. To reason with these structures there are mainly two different approaches: one can extend an already existing calculus and appropriate tools to deal with these structures in a native way, or one can try to find an abstraction for these structures where there are already well established calculi and good tool support for the chosen abstraction. Using the combined theory of sets, arithmetic and cardinalities has proven to be a useful abstraction in many verification areas and there are multiple decision procedures based on it for example by Ohlbach ([8]) and Kuncak et al. ([6, 10, 9, 12, 7]). We extend this theory in our approach.

Related Work There are many contributions in the area of reasoning with sets and cardinalities. In addition to the work by Ohlbach ([8]) and by Kuncak and his collaborators ([6, 10, 9, 12, 7]) we want to mention the following: In [4] the authors offer a different way of reasoning with sets, cardinalities and arithmetic: They consider reasoning techniques for unions of non-disjoint theories where at least one of the theories is *$\mathscr{P}$-gentle*. This property allows to use the Nelson-Oppen procedure for reasoning in combinations of non-disjoint theories. The authors identify the Löwenheim and Bernays-Schönfinkel-Ramsey classes to be $\mathscr{P}$-gentle which allows reasoning with sets and cardinalities.

In [11] Calogero G. Zarba introduces the language *2LSC* (two-level syllogistic with cardinality) that allows to reason about problems with sets, cardinality of sets, the elements of the sets and cardinal numbers. A decision procedure for quantifier-free formulae in 2LSC is presented. It is inspired by the Nelson-Oppen method and uses decision procedures for cardinal linear arithmetic and for the theory of the elements of the sets as black boxes. As with the Nelson-Oppen method, the theories must be stably-infinite, but there is no need for disjoint signatures, making the introduced method usable for 2LSC.

The authors of [1] identify two fragments of Presburger arithmetic with counting quantifiers and uninterpreted function symbols, namely *extended flat* formulae and *simply flat* formulae. They show that both fragments are decidable and establish bounds on the complexity of deciding the satisfiability of formulae in these fragments.

Contributions In this paper we extend the results in [6] and propose a method that allows us to reason about sets, arithmetic, cardinalities and sums of sets, which offers the possibility to be used as abstraction in more cases. The shown method is also a starting point for an extension that gives even more flexibility by introducing additional bridging functions like the minimal or maximal element of a set or measures. These are current examples in our goal of finding properties and methods that allow us to construct a framework for reasoning with sets, arithmetic and other bridging functions. Our method terminates and has the same worst case complexity as the approach by Kuncak et al. [6].

Structure In Section 2 we introduce the preliminaries needed for our method. Section 3 then shows our method and gives information about its properties. We conclude this paper in Section 4 by elaborating the different kinds of extensions that are possible for this method and by summarizing the paper.

2 Preliminaries

2.1 Syntax

Let $\mathsf{S} = \{\mathsf{nat}, \mathsf{num}, \mathsf{set}\}$ be a set of sorts, where nat is the sort of natural numbers, num is the so called *element sort* and set is the sort of sets with elements of sort num. The sort nat is essentially only introduced to express cardinalities of sets. Thus, we do not distinguish between nat and num for most of our considerations. The following sets of function and predicate symbols

$$\begin{array}{lcl}
\Omega_{\mathsf{c_{num}}} & := & \{K \mid K \text{ a constant of sort nat or num with fixed semantics}\} \\
\Omega_{\mathsf{c_{set}}} & := & \{\emptyset,\ \mathscr{U}\} \\
\Omega_{\mathsf{num}} & := & \Omega_{\mathsf{c_{num}}} \cup \{+ : \mathsf{num} \times \mathsf{num} \to \mathsf{num},\ \cdot : \mathsf{num} \times \mathsf{num} \to \mathsf{num}\} \\
\Omega_{\mathsf{set}} & := & \Omega_{\mathsf{c_{set}}} \cup \{\cup : \mathsf{set} \times \mathsf{set} \to \mathsf{set},\ \cap : \mathsf{set} \times \mathsf{set} \to \mathsf{set},\ \complement : \mathsf{set} \to \mathsf{set}\} \\
\Omega_{\mathsf{sum}} & := & \{\mathsf{card} : \mathsf{set} \to \mathsf{nat}\} \cup \{\mathsf{sum} : \mathsf{set} \to \mathsf{num}\} \\
\Pi_{\mathsf{num}} & := & \{\approx_{\mathsf{num}} : \mathsf{num} \times \mathsf{num},\ < : \mathsf{num} \times \mathsf{num}\} \\
\Pi_{\mathsf{set}} & := & \{\approx_{\mathsf{set}} : \mathsf{set} \times \mathsf{set},\ \subseteq : \mathsf{set} \times \mathsf{set}\}
\end{array}$$

are used to define the following four signatures that are used throughout this paper:

$$\begin{array}{lclll}
\Sigma_{\mathsf{sum}} & := & (\{\mathsf{nat}, \mathsf{num}, \mathsf{set}\}, & \Omega_{\mathsf{num}} \cup \Omega_{\mathsf{set}} \cup \Omega_{\mathsf{sum}}, & \Pi_{\mathsf{num}} \cup \Pi_{\mathsf{set}}\ ) \\
\Sigma_{\mathsf{arith}} & := & (\{\mathsf{nat}, \mathsf{num}\}, & \Omega_{\mathsf{num}}, & \Pi_{\mathsf{num}}\ )
\end{array}$$

The symbols in $\Omega_{\mathsf{c_{num}}}$ denote numbers and for the symbols in $\Omega_{\mathsf{c_{set}}}$, $\emptyset$ denotes the empty set and $\mathscr{U}$ denotes the universal set. We sometimes use $\bar{x}$ instead of $\complement x$ to denote the complement of a set and in cases where the meaning is unambiguous, $\approx$ instead of $\approx_{\mathsf{num}}$, respectively $\approx_{\mathsf{set}}$. We introduce the following constants with the given meaning as syntactic sugar: $\mathsf{MAXC} := \mathsf{card}(\mathscr{U})$ of sort nat and $\mathsf{MAXS} := \mathsf{sum}(\mathscr{U})$ of sort num.

Let $\mathscr{X} = (\mathscr{X}_{\mathsf{nat}}, \mathscr{X}_{\mathsf{num}}, \mathscr{X}_{\mathsf{set}})$ be a countably infinite many-sorted set of variables, where $\mathscr{X}_{\mathsf{nat}}$, $\mathscr{X}_{\mathsf{num}}$, and $\mathscr{X}_{\mathsf{set}}$ are sets of variables of sort nat, num and set respectively. If not stated otherwise, x, x_i, y denote variables of sort set, called *set variables*, k, k_i, l, l_i, s, s_i denote variables of sort nat or num, called *arithmetical variables*, v, v_i denote set and arithmetical variables, and A, A_i denote any set expression, where i is any index. We use $\leq, \geq, >, \subset, \supseteq, \supset$ and $\not\approx$ with the usual meaning.

Please note that this syntax explicitly allows to express constraints between terms of sort nat and num like $\forall x\ \mathsf{sum}(x) < 2\mathsf{card}(x)$ or, a bit more subtle, $\forall x_1\ \mathsf{sum}(x_0) < 0 \wedge ((x_1 \subseteq x_0 \wedge \mathsf{card}(x_1) \approx 1) \to \mathsf{sum}(x_1) > 0)$.

2.2 Semantics

We consider Σ_{sum}-structures of the form $\mathscr{A} := (\mathbb{N}, \mathbb{F}, \Omega_{\mathscr{A}}, \Pi_{\mathscr{A}})$, where

- $\mathbb{N}$ is the set of natural numbers that is only used as the codomain for the function card.
- $\mathbb{F} \subseteq \mathbb{R}$ is a set of numbers that is closed under addition and multiplication, called *element support (of the structure)*. It is used as domain for the symbols in Π_{num}, as domain and codomain for the symbols in Ω_{num}, and as the codomain for sum. Arithmetical variables are assigned values in $\mathbb{F}$.

  The finite power set of $\mathbb{F}$, $\mathscr{P}_f(\mathbb{F})$, is used as domain for card, sum and for the symbols in Π_{set}. Additionally it is used as domain and codomain for the symbols in Ω_{set} and set variables are assigned values in $\mathscr{P}_f(\mathbb{F})$.
- For every function symbol $f \in (\Omega_{\mathsf{num}} \cup \Omega_{\mathsf{set}} \cup \Omega_{\mathsf{sum}} \cup \Omega_{\mathsf{min}})$, $\Omega_{\mathscr{A}}$ contains a function $f_{\mathscr{A}}$ that defines the semantics of the function symbol. The semantics for the symbols in Ω_{num} and Ω_{set} are as expected. The semantics for the symbols in $\Omega_{\mathsf{sum}} \cup \Omega_{\mathsf{min}}$ are defined below.
- $\Pi_{\mathscr{A}}$ defines the semantics for the predicate symbols in $\Pi_{\mathsf{num}} \cup \Pi_{\mathsf{set}}$, which is as expected.

As with the sort nat, the support $\mathbb{N}$ is essentially only introduced to make sure, that the codomain of card is a non-negative integer. Thus, we do not distinguish between $\mathbb{N}$ and $\mathbb{F}$ for most of our considerations. The fact, that we consider $\mathscr{P}_f(\mathbb{F})$ as domain for sets implies that we are only considering finite sets. If not stated otherwise, c, c_i denote elements of $\mathbb{N}$, d, d_i, e, e_i denote elements of $\mathbb{F}$ and o, o_i denote elements of $\mathscr{P}_f(\mathbb{F})$, where i is any index.

The constant $\mathscr{U}$ has a special meaning for our approach as its interpretation, $\mathscr{U}_{\mathscr{A}}$, is a finite subset of $\mathbb{F}$ that defines the scope of interest. Thus we first introduce the following semantics considering $\mathscr{U}$ and afterwards the general semantics of the bridging functions for $o \in \mathscr{P}_f(\mathbb{F})$:

- $\mathscr{U}_{\mathscr{A}} \in \mathscr{P}_f(\mathbb{F})$,
- $\mathsf{MAXC}_{\mathscr{A}} = \mathsf{card}_{\mathscr{A}}(\mathscr{U}_{\mathscr{A}}) := |\mathscr{U}_{\mathscr{A}}|$, the number of elements in $\mathscr{U}_{\mathscr{A}}$,
- $\mathsf{MAXS}_{\mathscr{A}} = \mathsf{sum}_{\mathscr{A}}(\mathscr{U}_{\mathscr{A}}) := \sum\limits_{e \in \mathscr{U}_{\mathscr{A}}} e$,
- $\mathsf{card}_{\mathscr{A}}(o) := |o|$, the number of elements in the set o,
- $\mathsf{sum}_{\mathscr{A}}(o) := \sum\limits_{e \in o} e$.

Let $o \in \mathscr{P}_f(\mathbb{F})$, then $\complement_{\mathscr{A}} o := \{e \mid e \in \mathscr{U}_{\mathscr{A}} \text{ and } e \notin o\}$. In cases where no ambiguity can arise, we slightly abuse our notation by using $\mathscr{A}$ to denote the structure as well as the sorted supports of the structure. Let $\beta \rightarrow \mathscr{A}$ be a function assigning values to variables according of their sort that consists of $\beta_{\mathsf{nat}} : \mathscr{X}_{\mathsf{nat}} \rightarrow \mathbb{N}$, $\beta_{\mathsf{num}} : \mathscr{X}_{\mathsf{num}} \rightarrow \mathbb{F}$ and $\beta_{\mathsf{set}} : \mathscr{X}_{\mathsf{set}} \rightarrow \mathscr{P}_f(\mathbb{F})$. For β_{set} we impose the additional condition, that its codomain is $\mathscr{P}(\mathscr{U}_{\mathscr{A}})$.

To reflect that $\mathscr{U}_{\mathscr{A}}$ is our scope of interest we impose the following: In the case of universal quantification of a set variable, $\forall x\ p(x)$, is true, if and only if $p_{\mathscr{A}}(o)$ is true for all $o \in \mathscr{P}(\mathscr{U}_{\mathscr{A}})$ instead of all $o \in \mathscr{P}_f(\mathbb{F})$. In the case of existential quantification of a set variable, $\exists x\ p(x)$, is true, if and only if $p_{\mathscr{A}}(o)$ is true for at least one $o \in \mathscr{P}(\mathscr{U}_{\mathscr{A}})$ instead of at least one $o \in \mathscr{P}_f(\mathbb{F})$.

The semantics for the predicate and function symbols are fixed like this throughout the whole document. Therefore, only $\mathbb{F}$ may differ between different Σ_{sum}-structures.

Σ_{arith}-structures are defined accordingly. To extend a Σ_{arith}-structure to a Σ_{sum}-structure, the fixed semantics for the additional symbols need to be added.

2.3 Theories

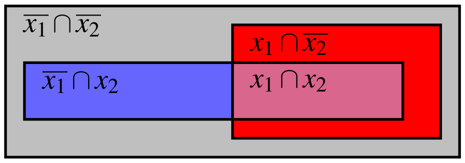
We use the well known theories of linear arithmetic and sets. The theory of sums of sets is defined by the axioms in Section 3 and the theory we consider is the union of these three theories. Whenever we talk about satisfiability, we mean it in the sense of "satisfiable modulo the according background theory".

2.4 Atomic Decompositions

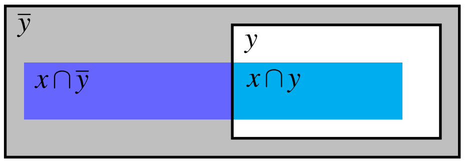
The concept of *atomic sets* and *atomic decompositions* was introduced by Ohlbach in [8] and used under the name of *cubes* by Kuncak et al. in [5]. Instead of giving a formal definition, we just give an intuition with help of Example 2.1. For n set variables, there are 2^n mutually disjoint regions in the Venn diagram that can be described as an intersection with n participating expressions, where each of this expressions is either a set variable or the complement of a set variable. These regions are called *atomic sets*. Every set variable can be described as union of atomic sets. This union is called *atomic decomposition*.

If not stated otherwise, $\mathscr{S}, \mathscr{S}_i$ denote atomic sets, where i is any index.

If all set expressions in a formula are represented as atomic decompositions, the formula is called *set-atomic* . Let $\mathbf{S}$ be the set of all atomic sets in a given formula and let $\mathbf{T}$ be a set such that $\emptyset \subset \mathbf{T} \subseteq \mathbf{S}$.


$\overline{x_1} \cap \overline{x_2}$
$x_1 \cap \overline{x_2}$
$\overline{x_1} \cap x_2$
$x_1 \cap x_2$

(a) The four atomic sets and the atomic decompositions for the two sets x_1 and x_2 of Example 2.1.



(b) The general intuition of Lemma 2.2.

Figure 1

Example 2.1. For a formula with exactly two set variables, x_1 and x_2 there are the following atomic sets: $\mathscr{S}_{00} := \overline{x_1} \cap \overline{x_2}$, $\mathscr{S}_{01} := \overline{x_1} \cap x_2$, $\mathscr{S}_{10} := x_1 \cap \overline{x_2}$, $\mathscr{S}_{11} := x_1 \cap x_2$ and these atomic decompositions: $x_1 := \mathscr{S}_{10} \cup \mathscr{S}_{11} = (x_1 \cap \overline{x_2}) \cup (x_1 \cap x_2)$, $x_2 := \mathscr{S}_{01} \cup \mathscr{S}_{11} = (\overline{x_1} \cap x_2) \cup (x_1 \cap x_2)$. This is depicted in Figure 1a.

2.5 Boolean Algebra and Presburger Arithmetic (BAPA)

As our approach is based on the work of Kuncak et al. [6] we give a short overview of their method. In Kuncak et al.'s approach [6] for solving constraints on sets and constraints on cardinalities a sentence in BAPA is given and then transformed to an equisatisfiable sentence containing only Presburger arithmetic. This transformation can roughly be summarized as follows:

1. Represent set atoms as arithmetical atoms by first replacing $\mathsf{A}_1 \approx_{\mathsf{set}} \mathsf{A}_2$ with $\mathsf{A}_1 \subseteq \mathsf{A}_2 \wedge \mathsf{A}_2 \subseteq \mathsf{A}_1$ and then replacing $\mathsf{A}_1 \subseteq \mathsf{A}_2$ with $\mathsf{card}\left(\mathsf{A}_1 \cap \overline{\mathsf{A}_2}\right) \approx 0$ for all set expressions A_1, A_2.
2. Represent every set expression as an atomic decomposition,
3. Use Lemma 2.2 for removing set quantifiers.

Lemma 2.2 ([6])**.** *Let $x_1, \ldots, x_n$ be finite disjoint sets and $l_1, \ldots, l_n, k_1, \ldots, k_n$ be natural numbers. Then the following two statements are equivalent:*

1. $\exists y \left(\bigwedge_{i=1}^{n} \mathsf{card}(x_i \cap y) = k_i \wedge \mathsf{card}(x_i \cap \overline{y}) = l_i \right)$, *where y is a finite set.*

2. $\bigwedge_{i=1}^{n} \mathsf{card}(x_i) = k_i + l_i$.

Figure 1b depicts the idea of Lemma 2.2. Lemma 2.2 introduces only a way of removing existentially quantified set variables, but we can use $\forall x\, F \equiv \neg \exists x\, \neg F$ to remove universally quantified set variables.

We use the term *original formula* to refer to the input formula of the procedure and *transformed formula* to refer to the output of the procedure.

The satisfiability of the transformed formula, and therefore the satisfiability of the original formula, can then be decided by using a solver for Presburger arithmetic. This can always be done, as Kuncak et al. [6] are only interested in the cardinality of sets, and do not impose additional constraints on the elements. Thus, if the cardinalities are known, the assignment for the atomic sets can be done in a rather simple way: Select an arbitrary element from the support, which has not yet been used, and assign it to the set. Repeat selecting and assigning elements until the number of elements in the set is equal to the value for the cardinality of the set. Proceed in this way until the assignment for all atomic sets is fixed. This way of deciding BAPA formulae is sound and complete and the result of the transformation is illustrated for a simple input in Example 2.3

Example 2.3. The sentence F' is the result of transforming the BAPA-sentence $F = \forall x\, \forall y\, \exists k\, (x \subseteq y \rightarrow \mathsf{card}(x) + k \approx \mathsf{card}(y))$ to an equisatisfiable formula that is pure arithmetic using the method by [6].

$$F' = \forall l_0\, \forall l_1 \Big(\mathsf{MAXC} \approx l_0 + l_1 \rightarrow \Big(\forall l_{00},\, l_{01},\, l_{10},\, l_{11} l_0 \approx l_{00} + l_{01} \wedge l_0 \approx l_{10} + l_{11} \rightarrow (\tag{1}$$

$$\exists k\, (l_{10} \approx 0 \rightarrow l_{10} + l_{11} + k \approx l_{01} + l_{11}))\Big)\Big) \tag{2}$$

3 Reasoning with Sums of Sets

We show how to extend the method in [6] to deal with the following two extensions:

1. allow formulae with an arbitrary combination of quantified and free variables,
2. introduce the new bridging function sum, returning the sum of all elements of a set.

With the additional bridging function, our goal is to still use a prover for arithmetic, that has no knowledge of the background theories of the bridging functions to decide satisfiability for a given formula. Thus we extend Kuncak et al.'s algorithm [6] in such a way, that is is able to transform a given formula that contains sum, into a formula of pure arithmetic. In contrast to [6] we do not necessarily end up with Presburger arithmetic, but depending on the element support of the chosen structure in different arithmetical fragments. However, we keep BAPA as part of the name for our approach, namely we talk about the theory $\mathsf{BAPA_S}$.

To encode some information of the background theory in the resulting arithmetical formula and therefore prevent the generation of models for the transformed formula that cannot be used to generate a model for the original formula, we add the following axioms during the transformation: To model the property that $\mathsf{sum}(\emptyset) = 0$, we enrich the original formula by adding

$$\mathsf{card}(\mathscr{S}) \approx 0 \rightarrow \mathsf{sum}(\mathscr{S}) \approx 0 \tag{SumEmpty}$$

for all $\mathscr{S} \in \mathbf{S}$. Additionally, we add

$$\mathsf{card}(\mathscr{S}_1) \approx 1 \wedge \mathsf{card}(\mathscr{S}_2) \approx 1 \rightarrow \mathsf{sum}(\mathscr{S}_1) \not\approx \mathsf{sum}(\mathscr{S}_2) \tag{UniqueSingleton}$$

for all $\mathscr{S}_1, \mathscr{S}_2 \in \mathbf{S}$ with $\mathscr{S}_1 \neq \mathscr{S}_2$.

Let F be a formula in $\mathsf{BAPA_S}$, then $\mathsf{Ax_S}(F)$ is the conjunction of the instances of SumEmpty for all $\mathscr{S} \in \mathbf{S}$ and the instances of UniqueSingleton for all $\mathscr{S}_1, \mathscr{S}_2 \in \mathbf{S}$ with $\mathscr{S}_1 \neq \mathscr{S}_2$.

Although the enrichment of the formula is at the end of the transformation, i.e. the axioms are added as pure arithmetical formulae, we show the axioms in their original form, i.e. containing sets and bridging functions, as this presentation is clearer and easier to comprehend.

Depending on the choice of the element support $\mathbb{F}$, the original formula and the transformed formula are not always equisatisfiable, even with the additional axioms. This is implied by the fact that sum and its background theory not only impose constraints on the number of elements of the sets, but on the elements as well, i.e. it is not sufficient to use arbitrary distinct elements of the domain as elements for the atomic sets, but we need to find elements that have certain properties. We identified a property of the element support which we name sum constructive (see Definition 3.3) for which we can show that the original formula and the transformed formula are always equisatisfiable. Intuitively it states that the element support $\mathbb{F}$ has the following property: There are infinitely many distinct ways to represent each element in $\mathbb{F}$ as sum of two distinct elements in $\mathbb{F}$.

The following steps are executed to transform a given formula in $\mathsf{BAPA_S}$ to a formula in pure arithmetic:

1. Eliminate $\approx_{\mathsf{set}}$ by replacing $\mathsf{A}_1 \approx_{\mathsf{set}} \mathsf{A}_2$ with $\mathsf{A}_1 \subseteq \mathsf{A}_2 \wedge \mathsf{A}_2 \subseteq \mathsf{A}_1$.
2. Eliminate $\subseteq$ by replacing $\mathsf{A}_1 \subseteq \mathsf{A}_2$ with $\mathsf{card}\big(\mathsf{A}_1 \cap \overline{\mathsf{A}_2}\big) \approx 0$.
3. Represent every set expression as an atomic decomposition.
4. Distribute card by replacing instances whose argument is a union of atomic sets with the sum of the appropriate atomic instances i.e. $\mathsf{card}(\bigcup_{\mathscr{S} \in \mathbf{T}} \mathscr{S}) \rightsquigarrow \sum_{\mathscr{S} \in \mathbf{T}} \mathsf{card}(\mathscr{S})$, and purify by introducing new nat-variables for instances of card.
5. Distribute sum by replacing instances whose argument is a union of atomic sets with the sum of the appropriate atomic instances i.e. $\mathsf{sum}(\bigcup_{\mathscr{S} \in \mathbf{T}} \mathscr{S}) \rightsquigarrow \sum_{\mathscr{S} \in \mathbf{T}} \mathsf{sum}(\mathscr{S})$, and purify by introducing new num-variables for the atomic instances of sum.
6. Eliminate quantifiers from inside out and add the instances of the axioms SumEmpty for all $\mathscr{S} \in \mathbf{S}$ and UniqueSingleton $\mathscr{S}_1, \mathscr{S}_2 \in \mathbf{S}$ with $\mathscr{S}_1 \neq \mathscr{S}_2$.

The method that transforms a formula applying steps 1 to 4 is called α_{pre} and $\alpha_{\mathsf{pre}}(F)$ denotes the result of applying α_{pre} on a formula F. α_{pre} is identical to the first part of the approach described in [6]. Please note that after α_{pre} is applied to a formula F, $\alpha_{\mathsf{pre}}(F)$ is set-atomic and set expressions appear only as argument of bridging functions. Additionally we know that all instances of card are atomic instances and due to the purification only occur as kind of prefix of the original formula. The method that transforms a formula applying steps 1 to 6 is called α_{S}. $\alpha_{\mathsf{S}}(F)$ denotes the result of applying α_{S} on a formula F.

Step 6, from now on denoted $\alpha_{\mathsf{S}}^{\mathsf{Q}}$ depends on the total number of set variables and the number and type of quantification of quantified set variables. Its core component is the iterative removal of existential and universal set quantifiers by using an extension of Lemma 2.2. In the following, we do not explicitly use the duality $\forall x\, F \equiv \neg\exists x\, \neg F$ to rewrite the formula accordingly, but we do an additional rewrite step to remove the negation symbol. Thus the type of the set quantifier under consideration just determines which junctor and which type of quantifier is used while building the resulting formulae. Some details on $\alpha_{\mathsf{S}}^{\mathsf{Q}}$ are given in Definition 3.1 and Example 3.2 illustrates its use.

Definition 3.1. $\alpha_{\mathsf{S}}^{\mathsf{Q}}$ is defined as follows: Let F' be a formula in prenex normal form in $\mathsf{BAPA}_{\mathsf{S}}$. The formula F is derived from F' by applying α_{pre}, and then distribute sum and purify according to step 5. Further let n denote the number of distinct set variables in F', q the number of set quantifiers in F' and r the number of all quantifiers in F'. Further, let $m = 2^n$.

Due to the fact that F is the result of applying first α_{pre} and then step 5 of α_{S}, we know that F has the following form

$$F = Q_1\, v_1 \ldots Q_r\, v_r\ G, \text{where}$$
$$G = \exists l_{n,1} \ldots l_{n,m} \quad \exists s_{n,1} \ldots s_{n,m} \bigwedge_{i=1}^{m} \mathsf{card}(\mathscr{S}_{n,i}) \approx l_{n,i} \wedge \bigwedge_{i=1}^{m} \mathsf{sum}(\mathscr{S}_{n,i}) \approx s_{n,i} \wedge H$$

and that H does not contain any set expressions.

1. If $q = 0$, i.e. F contains only free set variables:

   This implies that there are no quantified set variable. We then add the axioms to the formula constructed by putting the quantifiers in front of H. We return $\mathsf{Ax}_{\mathsf{S}}(F) \wedge Q_1\, v_1\ \ldots Q_r\, v_r\, H$
2. If $q > 0$, i.e. F contains quantified set variables:

   We return the result of $\alpha_{\mathsf{S}}^{\mathsf{E}}([Q_1\, v_1, \ldots, Q_r\, v_r],\ n,\ q,\ 0,\ G)$, where $\alpha_{\mathsf{S}}^{\mathsf{E}}$ is described below.

Instead of giving a formal definition of $\alpha_{\mathsf{S}}^{\mathsf{E}}$, we just give a general intuition on how it works and then present some of the details with the help of Example 3.2. $\alpha_{\mathsf{S}}^{\mathsf{E}}$ essentialy picks the quantifier $Q_r\, v_r$ out of the list $Q_1\, v_1, \ldots, Q_{r-1}\, v_{r-1}, Q_r\, v_r$ constructs a new formula G' based on G and some other constraints and then calls itself recursively with the shortened list $Q_1\, v_1, \ldots, Q_{r-1}\, v_{r-1}$ and the newly constructed G'. If the list of quantifiers is empty, the formula G is returned. If v_r is an arithmetical variable, $G' := Q_r\, v_r\ G$, otherwise G' is constructed from G by using Lemma 2.2, where we use it for constructing defining hierarchies of sums for card and sum. If not all set variables are quantified, the definition of the purification for card and sum is removed after the last quantifier on sets is removed.

Example 3.2. Let $F_0 := \forall x_1\ \ (x_0 \subseteq x_1 \rightarrow \mathsf{sum}(x_0) \approx \mathsf{sum}(x_0 \cap x_1))$ be a formula in $\mathsf{BAPA_S}$. We show how $\alpha_{\mathsf{S}}(F_0)$ is constructed.

1. **Eliminate** $\approx_{\mathsf{set}}$ **, and eliminate** $\subseteq$**:**

   $\approx_{\mathsf{set}}$ does not appear in F_0. The atom $x_0 \subseteq x_1$ is rewritten to $\mathsf{card}(x_0 \cap \overline{x_1}) \approx 0$.

   Thus we get $F_1 := \forall x_1\ \ (\mathsf{card}(x_0 \cap \overline{x_1}) \approx 0 \rightarrow \mathsf{sum}(x_0) \approx \mathsf{sum}(x_0 \cap x_1))$.

2. **Introduce Atomic Decompositions:**

   The atomic sets in F_0 are: $\mathscr{S}_{00} := \overline{x_0} \cap \overline{x_1}$, $\mathscr{S}_{01} := \overline{x_0} \cap x_1$, $\mathscr{S}_{10} := x_0 \cap \overline{x_1}$, $\mathscr{S}_{11} := x_0 \cap x_1$. By making F_1 set-atomic, we get $F_2 := \forall x_1\ \ (\mathsf{card}(\mathscr{S}_{10}) \approx 0 \rightarrow \mathsf{sum}(\mathscr{S}_{10} \cup \mathscr{S}_{11}) \approx \mathsf{sum}(\mathscr{S}_{11}))$.

3. **Distribute** card**, purify** card **:**

   The only instance of card in F_2 is already an atomic instance, thus we just purify

   $$\begin{aligned}F_3 := \forall x_1\quad & \exists l_{00},\, l_{01},\, l_{10},\, l_{11}\\ & \mathsf{card}(\mathscr{S}_{00}) \approx l_{00} \wedge \cdots \wedge \mathsf{card}(\mathscr{S}_{11}) \approx l_{11} \wedge (l_{10} \approx 0 \rightarrow \mathsf{sum}(\mathscr{S}_{10} \cup \mathscr{S}_{11}) \approx \mathsf{sum}(\mathscr{S}_{11}))\end{aligned}$$

   After this step, α_{pre} is finished and the formula F_3 is set-atomic.

4. **Distribute** sum**, purify** sum**:**

   $\mathsf{sum}(\mathscr{S}_{10} \cup \mathscr{S}_{11}) \approx \mathsf{sum}(\mathscr{S}_{11})$ is rewritten to $\mathsf{sum}(\mathscr{S}_{10}) + \mathsf{sum}(\mathscr{S}_{11}) \approx \mathsf{sum}(\mathscr{S}_{11})$.

   $$\begin{aligned}F_4 := \forall x_1\quad & \exists l_{00},\, l_{01},\, l_{10},\, l_{11}\quad \exists s_{00},\, s_{01},\, s_{10},\, s_{11}\\ & \mathsf{card}(\mathscr{S}_{00}) \approx l_{00} \wedge \cdots \wedge \mathsf{card}(\mathscr{S}_{11}) \approx l_{11} \wedge \mathsf{sum}(\mathscr{S}_{00}) \approx s_{00} \wedge \cdots \wedge \mathsf{sum}(\mathscr{S}_{11}) \approx s_{11} \wedge\\ & (l_{10} \approx 0 \rightarrow s_{10} + s_{11} \approx s_{11})\end{aligned}$$

5. **Eliminate Quantifiers** (Step 6 in α_{S})**:**

   Step 1: **Call** $\alpha_{\mathsf{S}}^{\mathsf{Q}}(F_4)$:

   The number of distinct set variables in F_1 is $n = 2$, the number of set quantifiers in F_1 is $q = 1$ and the number of all quantifiers in F_1 is $r = 1$.
   As F_4 contains quantified variables, we call $\alpha_{\mathsf{S}}^{\mathsf{E}}([\forall x_1],\ 2,\ 1,\ 0,\ G)$, where

   $$\begin{aligned}G := & \exists l_{00},\, l_{01},\, l_{10},\, l_{11}\quad \exists s_{00},\, s_{01},\, s_{10},\, s_{11}\\ & \mathsf{card}(\mathscr{S}_{00}) \approx l_{00} \wedge \cdots \wedge \mathsf{card}(\mathscr{S}_{11}) \approx l_{11} \wedge \mathsf{sum}(\mathscr{S}_{00}) \approx s_{00} \wedge \cdots \wedge \mathsf{sum}(\mathscr{S}_{11}) \approx s_{11} \wedge\\ & (l_{10} \approx 0 \rightarrow s_{10} + s_{11} \approx s_{11})\end{aligned}$$

   Thus, we construct G' as follows and then return the result of $\alpha_{\mathsf{S}}^{\mathsf{E}}([],\ 2,\ 1,\ 1,\ G')$.

   $$\begin{aligned}G' := & \forall l_{00},\, l_{01},\, l_{10},\, l_{11}\quad \forall s_{00},\, s_{01},\, s_{10},\, s_{11}(\\ & \quad l_0 \approx l_{00} + l_{01} \wedge l_1 \approx l_{10} + l_{11} \wedge s_0 \approx s_{00} + s_{01} \wedge s_1 \approx s_{10} + s_{11} \wedge \mathsf{Ax_S}(F)\\ & \quad) \rightarrow (l_{10} \approx 0 \rightarrow s_{10} + s_{11} \approx s_{11})\end{aligned}$$

Some explanations on the construction of G':

- As we are considering $\forall x_1$, the quantifiers $\exists l_{00}, \dots l_{11} \exists s_{00}, \dots s_{11}$ in G are changed to universal quantifiers and the core formula is connected with $\rightarrow$.
- As $\forall x_1$ is the first set quantifier to be treated, we add the appropriate axioms:

$$\begin{aligned}&\mathsf{Ax}_{\mathsf{S}}(F) := \\ &(l_{01} \approx 1 \wedge l_{00} \approx 1) \rightarrow (s_{01} \not\approx s_{00}) \wedge (l_{01} \approx 1 \wedge l_{10} \approx 1) \rightarrow (s_{01} \not\approx s_{10}) \wedge \\ &(l_{10} \approx 1 \wedge l_{00} \approx 1) \rightarrow (s_{10} \not\approx s_{00}) \wedge (l_{11} \approx 1 \wedge l_{00} \approx 1) \rightarrow (s_{11} \not\approx s_{00}) \wedge \\ &(l_{11} \approx 1 \wedge l_{01} \approx 1) \rightarrow (s_{11} \not\approx s_{01}) \wedge (l_{11} \approx 1 \wedge l_{10} \approx 1) \rightarrow (s_{11} \not\approx s_{10}) \wedge \\ &(l_{00} \approx 0 \rightarrow s_{00} \approx 0) \wedge (l_{01} \approx 0 \rightarrow s_{01} \approx 0) \wedge (l_{10} \approx 0 \rightarrow s_{10} \approx 0) \wedge (l_{11} \approx 0 \rightarrow s_{11} \approx 0)\end{aligned}$$

- We check if $\#_{used} = q - 1$: As $0 = 1 - 1 = 0$, i.e. we are considering the last set quantifier, we do not add the following definitions of the purification

$$\exists l_0, l_1 \exists s_0, s_1 \mathsf{card}(\overline{x_0}) \approx l_0 \wedge \mathsf{card}(x_0) \approx l_1 \wedge \mathsf{sum}(\overline{x_0}) \approx s_0 \wedge \mathsf{sum}(x_0) \approx s_1$$

  to construct G' from G.

Step 2: **Call $\alpha^{\mathsf{E}}_{\mathsf{S}}([],\ 2,\ 1,\ 1,\ G')$:** As the list of quantifiers is empty, $\alpha^{\mathsf{E}}_{\mathsf{S}}$ returns G'.

$$\begin{aligned}F_5 := \forall l_{00},\ l_{01},\ l_{10},\ l_{11}\ \forall s_{00},\ s_{01},\ s_{10},\ s_{11}(\qquad (3)\\ l_0 \approx l_{00} + l_{01} \wedge l_1 \approx l_{10} + l_{11} \wedge s_0 \approx s_{00} + s_{01} \wedge s_1 \approx s_{10} + s_{11} \wedge \qquad (4)\\ \mathsf{Ax}_{\mathsf{S}}(F)) \rightarrow (l_{10} \approx 0 \rightarrow s_{10} + s_{11} \approx s_{11}) \qquad (5)\end{aligned}$$

The formula F_5 does not contain any set expression and consists of arithmetical constraints only.

3.1 Properties

Thus with α_{S} we can transform a given formula in $\mathsf{BAPA}_{\mathsf{S}}$ to a formula in pure arithmetic. The process always terminates (Theorem 3.6). We identify a condition (sum constructive (Definition 3.3)) that guarantees that the original formula and transformed formula are equisatisfiable (Theorem 3.4): We show that we can construct a model for the original formula from a model for the transformed formula (Corollary 3.5). We also offer information on the complexity of our method in Theorem 3.7.

We only provide ideas of the following proofs. A document with full proofs is available online[1].

Definition 3.3 (sum constructive)**.** Let $\mathscr{A}$ be a Σ_{sum}-structure with element support $\mathbb{F}$. $\mathbb{F}$ is called *sum constructive* if and only if for all $c \in \mathbb{F}$ there exist infinitely many $a, b \in \mathbb{F}$, such that $a \neq b$ and $a + b = c$.

If the element support of a structure is sum constructive, we call the structure sum constructive.

A formula is called *sum constructive satisfiable* if it has a sum constructive model.

Please note that the property sum constructive is only linked to the element support of the structure, i.e. it does not rely on, or impose anything upon the interpretations of the functions and predicates in a structure. The property guarantees that it is always possible to construct the missing elements in the assignments for the set variables, as the transformed formula only provides information on the number of elements and the sum of all elements for each set. Examples of sets that are sum constructive are $\mathbb{Z}$, $\mathbb{Q}$ and $\mathbb{R}$. Examples of sets that are not sum constructive are $\mathbb{N}$, $\mathbb{R}_+$ and any finite set.

[1] `userpages.uni-koblenz.de/~mbender/sum_full.pdf`

Theorem 3.4 (Equisatisfiability for sum constructive structures)**.** *For all formulae F in* $\mathsf{BAPA_S}$*, F is sum constructive satisfiable if and only if $\alpha_\mathsf{S}(F)$ is sum constructive satisfiable.*

Proof. (Idea). The proof is based on how one can use sum constructive model for F to construct a sum constructive model for $\alpha_\mathsf{S}(F)$ and vice versa. Transforming a Σ_sum-structure to an Σ_arith-structure and vice versa can be done easily, therefore we focus on how to construct an assignment for the free variables.

- To use the assignments for sets in F to construct assignments for the arithmetical variables in $\alpha_\mathsf{S}(F)$ that correspond to cardinalities and sums of sets can be done straight forward. The property sum constructive is not really needed for this part of the proof.
- The other way around is only possible if the element sort is sum constructive as we rely on this property to construct information about the elements of the sets. As we know how many elements each set has and what the sum of all the elements in a set is, we can use the two values to construct all the elements of the sets and therefore construct an assignment for a sum constructive-model for F from a sum constructive-model of $\alpha_\mathsf{S}(F)$.

□

Corollary 3.5 (Model Construction)**.** *For all formulae F in* $\mathsf{BAPA_S}$*, all sum constructive Σ_arith-structures $\mathscr{A}$ and all assignments $\beta : \mathscr{X} \to \mathscr{A}$, we can construct a Σ_sum-structure $\mathscr{A}'$ and an assignment $\beta' : \mathscr{X} \to \mathscr{A}'$ such that, if $\mathscr{A}, \beta \models \alpha_\mathsf{S}(F)$ then $\mathscr{A}', \beta' \models F$.*

Proof. (Idea). This follows from the proof of Theorem 3.4. □

Theorem 3.6 (Termination)**.** *Let F be a formula in* $\mathsf{BAPA_S}$ *and let us assume that we have a prover for arithmetic that always terminates. Therefore, checking if F is sum constructive satisfiable is decidable.*

Proof. (Idea). This follows immediately from the way that α_S works, as later steps do not influence previous steps and all steps are essentially rewriting operations with a bounded number of steps. □

Theorem 3.7 (Complexity)**.** *Let F be a formula in* $\mathsf{BAPA_S}$ *and let* $\mathsf{size}(F)$ *denote the size of the formula F. The following hold:*

1. *The size of $\alpha_\mathsf{S}(F)$ is bounded by $O(2^{2\mathsf{size}(F)})$.*
2. *The time for the transformation of F to $\alpha_\mathsf{S}(F)$ is bounded by $O(2^{2\mathsf{size}(F)})$.*
3. *The number of quantifier alternations in F and $\alpha_\mathsf{S}(F)$ are identical.*

Proof. (Idea). The proofs for these three properties can be done by following the procedure α_S and analyzing how it changes the given formula. □

4 Conclusion

We presented a method for reasoning in the combined theory of sets, cardinalities of sets, sums of sets and arithmetic with an arbitrary mix of quantified and free set variables. This theory is useful as it can be used as an abstraction for verification tasks. We have shown that our method is sound and complete and that its complexity is comparable to the approach in [6] which is the basis of our extension.

Additionally, this method offers a good starting point for further extensions. Preliminary results show that we can lift it to the theory $\mathsf{BAPA_M}$ which contains sets, arithmetic, cardinalities and the two bridging functions min and max that return the minimal, respectively the maximal element of a set. There is already work on sets with constraints on infimum and supremum of sets showing that this fragment can be useful [7]. $\mathsf{BAPA_S}$ and $\mathsf{BAPA_M}$ can be combined in a natural way leading to a broader variety of possible uses. With those three bridging functions, we hope to see similarities and properties that allow us to establish a more general method of reasoning with arithmetic, sets, cardinalities and more bridging functions, to get a language with a higher expressiveness that has established decision procedures and can be used as abstraction for verification and reasoning tasks.

On a different path, we expect that we can extend this framework to allow for measures as bridging functions from sets to arithmetic and then from intervals to arithmetic. We want to pursue this direction to get an decision procedure for intervals and measures. This is a step towards the target of developing a decision procedure for fragments of the duration calculus ([3, 2]).

The implementation of the presented method α_{S} is an ongoing task.

Acknowledgment: We thank Viorica Sofronie-Stokkermans for her helpful feedback and Matthias Horbach for many valuable discussions. We are thankful for the constructive comments of the anonymous reviewers, as well.

References

[1] F. Alberti, S. Ghilardi, and E. Pagani. Counting constraints in flat array fragments. *CoRR*, abs/1602.00458, 2016.

[2] Z. Chaochen and M. R. Hansen. *Duration calculus: a formal approach to real-time systems*. Springer, Berlin New York, 2004.

[3] Z. Chaochen, C. A. R. Hoare, and A. P. Ravn. A calculus of durations. *Inf. Process. Lett.*, 40(5):269–276, 1991.

[4] P. Chocron, P. Fontaine, and C. Ringeissen. A gentle non-disjoint combination of satisfiability procedures. In S. Demri, D. Kapur, and C. Weidenbach, editors, *IJCAR 2014, Proceedings*, volume 8562 of *LNCS*, pages 122–136. Springer, 2014.

[5] V. Kuncak, H. H. Nguyen, and M. C. Rinard. An algorithm for deciding BAPA: Boolean algebra with presburger arithmetic. In R. Nieuwenhuis, editor, *CADE-20, Proceedings*, volume 3632 of *LNCS*, pages 260–277. Springer, 2005.

[6] V. Kuncak, H. H. Nguyen, and M. C. Rinard. Deciding boolean algebra with presburger arithmetic. *J. Autom. Reasoning*, 36(3):213–239, 2006.

[7] V. Kuncak, R. Piskac, and P. Suter. Ordered sets in the calculus of data structures. In A. Dawar and H. Veith, editors, *CSL 2010, Proceedings*, volume 6247 of *Lecture Notes in Computer Science*, pages 34–48. Springer, 2010.

[8] H. J. Ohlbach. Set description languages and reasoning about numerical features of sets. In P. Lambrix, A. Borgida, M. Lenzerini, R. Möller, and P. F. Patel-Schneider, editors, *International Workshop on Description Logics (DL'99), Proceedings*, volume 22 of *CEUR Workshop Proceedings*. CEUR-WS.org, 1999.

[9] T. Wies, R. Piskac, and V. Kuncak. Combining theories with shared set operations. In S. Ghilardi and R. Sebastiani, editors, *FroCoS 2009, Proceedings*, volume 5749 of *LNCS*, pages 366–382. Springer, 2009.

[10] K. Yessenov, R. Piskac, and V. Kuncak. Collections, cardinalities, and relations. In G. Barthe and M. V. Hermenegildo, editors, *VMCAI 2010, Proceedings*, volume 5944 of *LNCS*, pages 380–395. Springer, 2010.

[11] C. G. Zarba. Combining sets with cardinals. *J. Autom. Reasoning*, 34(1):1–29, 2005.

[12] K. Zee, V. Kuncak, and M. C. Rinard. Full functional verification of linked data structures. In R. Gupta and S. P. Amarasinghe, editors, *Proceedings of the ACM SIGPLAN 2008 Conference on Programming Language Design and Implementation, Tucson, AZ, USA, June 7-13, 2008*, pages 349–361. ACM, 2008.