Ordered Sets in the Calculus of Data Structures

Viktor Kuncak, Ruzica Piskac, and Philippe Suter *

firstname.lastname@epfl.ch
Swiss Federal Institute of Technology Lausanne (EPFL)

Abstract. Our goal is to identify families of relations that are useful for reasoning about software. We describe such families using decidable quantifier-free classes of logical constraints with a rich set of operations. A key challenge is to define such classes of constraints in a modular way, by combining multiple decidable classes. Working with quantifier-free combinations of constraints makes the combination agenda more realistic and the resulting logics more likely to be tractable than in the presence of quantifiers.

Our approach to combination is based on reducing decidable fragments to a common class, Boolean Algebra with Presburger Arithmetic (BAPA). This logic was introduced by Feferman and Vaught in 1959 and can express properties of uninterpreted sets of elements, with set algebra operations and equicardinality relation (consequently, it can also express Presburger arithmetic constraints on cardinalities of sets). Combination by reduction to BAPA allows us to obtain decidable quantifier-free combinations of decidable logics that share BAPA operations. We use the term *Calculus of Data Structures* to denote a family of decidable constraints that reduce to BAPA. This class includes, for example, combinations of formulas in BAPA, weak monadic second-order logic of k-successors, two-variable logic with counting, and term algebras with certain homomorphisms. The approach of reduction to BAPA generalizes the Nelson-Oppen combination that forms the foundation of constraint solvers used in software verification. BAPA is convenient as a target for reductions because it admits quantifier elimination and its quantifier-free fragment is NP-complete.

We describe a new member of the Calculus of Data Structures: a quantifier-free fragment that supports 1) boolean algebra of finite and infinite sets of real numbers, 2) linear arithmetic over real numbers, 3) formulas that can restrict chosen set or element variables to range over integers (providing, among others, the power of mixed integer arithmetic and sets of integers), 4) the cardinality operators, stating whether a given set has a given finite cardinality or is infinite, 5) infimum and supremum operators on sets. Among the applications of this logic are reasoning about the externally observable behavior of data structures such as sorted lists and priority queues, and specifying witness functions for the BAPA synthesis problem. We describe an abstract reduction to BAPA for our logic, proving that the satisfiability of the logic is in NP and that it can be combined with the other fragments of the Calculus of Data Structures.

* This research was supported in part by the Swiss NSF Grant #120433.

1 Introduction

Many useful decidable constraints involve a notion of sets. To combine such constraints, we need a method that in addition to equality and propositional operations allows the constraints from different classes to have common operations on sets. Calculus of Data Structures [KPSW10] is an approach to combine multiple classes of constraints that allow sharing of set operations, by reducing each class to constraints of sets with a cardinality operation.

Constraints on sets arise in a variety of tasks, from software verification to interactive theorem proving. In addition to operators that combine collections into new ones (such as union or intersection), these formulas often involve the cardinality operator computing the number of elements in collections. Several decision procedures for sets and multisets with cardinality operator have been described recently [KNR06,KR07,PK08c,PK08a,PK08b]. Among these results is the NP-completeness of the theory of sets and multisets with the cardinality operator [PK08c]. In addition to their use in verification, these decision procedures can be used to synthesize code from specifications [KMPS10]. The applicability of these decision procedures can be increased by combining them with other decision procedures and theorem provers [KPSW10]. The existing decision procedures for collections with cardinality bounds do not consider operations that couple the collection operations with the operations on the elements.

In terms of quantified constraints, weak second-order theory of sets of totally ordered elements *without* cardinality operator is known to be decidable [Lae68], also as a consequence of S2S decidability [Rab69]. The restriction to quantification over finite sets is essential; the second-order theory of total order is undecidable, as is the theory with quantification over countable sets, or sets of rational or real numbers [She75].

The logic we consider is quantifier free. It tightly interconnects cardinalities, sets, and the underlying total order. Our operations of infimum and supremum only make sense when the underlying theory contains a complete lattice and are specific to theories that contain sets. Moreover, there is no natural way to e.g. remove the cardinality operator from a theory of sets without eliminating the notion of the set altogether. Therefore, previous general results on combinations of theories [RRZ05, SS07, Ghi05] do not appear to simplify the proofs that we present here, and have certainly not been used to show the present result.

We start from the NP-completeness result for sets with the cardinality operator [PK08c] and extend it to collections of totally ordered elements. The key challenge in considering such a logic is to avoid NEXPTIME hardness which easily arises in the presence of sets (see e.g. the addition of relation images to QFBAPA [YPK10]).

As a concrete and natural example of an ordered domain into which other orders can be embedded, we consider the set of real numbers (we do prove theorems that show that the automated reasoning can be performed in terms of rational numbers that denote bounds of certain intervals). To support discrete orders, we introduce the set of integers as a specific subset. Therefore, we can specify that a given set contains only integers, whenever this is desirable. Given a collection variable C, our formulas support computing the number of elements $|C|$, but also the minimum $\min(C)$ and the maximum $\max(C)$ of all elements of C (these operations are partial when the sets can be unbounded; our logic support appropriate predicates to check boundedness and finiteness). We can also define in our language the function $\mathsf{take}(k, C)$ that computes the least k elements from the collection C, where k is an integer variable. More generally, one can compute $\mathsf{lrange}(i, j, C)$ the collection of elements from position i to position j in the ordered collection, counting from the minimum element. A special case of this definable operation is extracting the i^{th} smallest or i^{th} biggest element of a sorted collection.

There is a number of areas in which we believe our constraints are useful.

1. Our constraints can be used to model programs that manipulate data structures. Whereas previous decision procedures supported unsorted sets, our result allows us to additionally consider ordered sets. The presence of order means that we can define operations such as extracting the least element of a set, which gives us complete algebraic laws for the external behavior of priority queues and sorted lists or trees (without duplicates). Our language supports not only operations of insertion and removal but also merging and comparison of sets, as well as selecting subsets of given size and indexing elements.
2. We can define in our language a natural relationship $A < B$ on sets, denoting $\forall x \in A. \forall y \in B.\ x < y$, simply by $\max(A) < \min(B)$. This relationship is useful in specifying e.g. invariants of binary search trees and can be used to verify lookup operations on a binary search tree.
3. Using sets defined over total orders, we are able to remove non-determinism in program synthesis. In [KMPS10] we have developed a synthesizer that works for arbitrary QFBAPA formulas. The synthesizer invokes a quantifier elimination procedure and uses the test terms from quantifier elimination as the synthesized program. These test terms involve choosing k elements from a Venn region, where the value k is computed in the synthesized program. Despite many good closure properties of QFBAPA, we found no natural way to introduce such test terms as part of QFBAPA itself. With the addition of ordering, functions such as $\mathsf{take}(k, C)$ suffice to specify all test terms. The presence of ordering in the specification language means that the user of synthesis can write specifications that have a unique solution.

Our result is formulated as a BAPA reduction and can thus be combined with other logics using the non-disjoint combination framework of [WPK09].

2 Examples

2.1 Using Ordered Sets in Verification

As an example of the application of our decision procedure to program verification, consider first the program of Figure 1, which defines a functional binary search tree in the Scala programming language [OSV08] and introduces two functions. content computes the elements stored in the tree (and is used only for specification purposes). find checks whether a given element is contained in the tree and is a key data structure operation.

```
object BSTSet
  sealed abstract class Tree
  private case class Leaf() extends Tree
  private case class Node(left: Tree, value: Int, right: Tree) extends Tree
   @invariant(max(content(left)) < value && value < min(content(right))

  def content(t: Tree): Set[Int] = t match {
    case Leaf() ⇒ ∅
    case Node(l,e,r) ⇒ content(l) ∪ { e } ∪ content(r) }

  def find(e: Int, t: Tree): Boolean = (t match
    case Leaf() ⇒ false
    case Node(l,v,r) ⇒
      if (e < v) find(e, l)
      else if (e == v) true
      else find(e, r)
  ) ensuring (res ⇒ (res == (e ∈ content(t))))
```

Fig. 1. Looking up an element in a binary search tree.

Verifying the property specified for find requires taking into account the invariant on the sortedness of trees. The difficult case is showing that if the procedure does *not* find an element, then the element is indeed not in the tree. The proof uses the fact that, for each node with value v, all elements L in the left subtree are less than v, and all elements in the right subtree are larger than v. We can express this condition as $\max(L) < v < \min(R)$. By applying standard techniques to reduce functional programs to formulas, we obtain verification conditions for find such as the following:

$$(\max(L) < v < \min(R) \wedge e < v) \rightarrow (e \in (L \cup \{v\} \cup R) \leftrightarrow e \in L)$$

Such a formula belongs to our decidable class, and can be handled using the decision procedure that we present in the sequel.

Figure 2 shows an example of `lookup` operation that finds the element of a given rank in a binary search tree. The verification condition formulas for this example can be expressed in our logic and proved using our decision procedure; one example verification condition is

$$C = L \cup \{v\} \cup R \wedge \max(L) < v < \min(R) \rightarrow \mathsf{lrange}(C, \mathsf{card}(L), \mathsf{card}(L)+1) = \{v\}.$$

Figure 3 shows a partitioning function such as the one used as part of the quicksort algorithm. The function splits an unordered collection into a collection

```
object BSTSet {
  sealed abstract class Tree
  private case class Leaf() extends Tree
  private case class Node(left: Tree, value: Int, right: Tree, size : Int) extends Tree {
    @invariant(content(this.left).max < this.value
          && this.value < content(this.right).min)
          && size = content(this).size }

  def lookup(index : Int, t: Tree): Int =
  require (0 <= index && index < t.size)
    t match {
    case Leaf() ⇒ error "out of bounds"
    case Node(l,v,r,_) ⇒
      if (index < l.size) lookup(index, l)
      else if (index == l.size) v
      else lookup(index − l.size − 1, r)
    }
  ensuring (v ⇒ (content(t).lrange(index,index+1))= {v}) }
```

Fig. 2. Finding an element of a given rank in a binary search tree.

```
def partition(s: Set, pivot: Int): (Set,Set) =
  var remaining = s
  var below = {}
  var above = {}
  while (invariant below ∪ above ∪ remaining = s ∧
                  below = {} ∨ max(below) ≤ pivot ∧
                  above = {} ∨ pivot < min(above))
   (remaining != {}) {
    var e = chooseOne(remaining)
    remaining = remaining \ {e}
    if (e ≤ pivot)
      below = below ∪ {e}
    else
      above = above ∪ {e}
  }
}) ensuring ((below,above) ⇒ below ∪ above = s ∧
                           below = {} ∨ max(below) ≤ pivot ∧
                           above = {} ∨ pivot < min(above))
```

Fig. 3. Partitioning an unordered collection

of those elements that are less than equal to a given pivot element, and those elements that are greater than the pivot. Given the appropriate loop invariant (Figure 3), the following verification conditions can also be proved using our decision procedure:

$$(\mathsf{above} = \emptyset \lor \mathsf{pivot} < \min(\mathsf{above})) \land \neg(e \leq \mathsf{pivot}) \land \mathsf{above}' = \mathsf{above} \cup \{e\}$$
$$\rightarrow \mathsf{pivot} < \min(\mathsf{above}').$$

2.2 Using Ordered Sets in Program Synthesis

Synthesizing software from given specifications [MW80] should increase the productivity of a programmer and the chances of obtaining error-free software that entirely corresponds to its specification. The concept of ordering immediately yields a much larger number of definable functions, such as take and lrange. These functions are sufficient to express within the logic the Skolem functions of quantified formulas of BAPA. Consider, for example, the formula

$$\forall S. \forall k.\ \exists A. \exists B.\ (|S| = 2k \rightarrow S = A \cup B \land A \cap B = \emptyset \land |A| = |B|)$$

This formula has a witness function $f(S, k)$ computing the sets (A, B) by $f(S, k) = (\mathsf{take}(k/2, S), S \setminus \mathsf{take}(k/2, S))$, where $k/2$ denotes integer division by 2, which is definable in integer linear arithmetic. Using ordering on sets, we can define such a computable witness function for every valid BAPA formula with a $\forall^\star\exists^\star$ prefix, thus ensuring a stronger form of quantifier elimination. We have used such witness functions as an output of a synthesis procedure for BAPA [KMPS10]; there the underlying programming language implementation used an implicit ordering on elements to compute f. Without an ordering on elements in the logic it was not clear how to specify a particular subset of a set of a given size. Using the ordering of elements (and the induced partial order on the sets) make the specification more precise and thus improves the predictability of the synthesized code.

3 Unordered, Possibly Infinite Sets with Cardinalities

As a background for our main result on sets of *ordered* elements, we establish the complexity for the satisfiability problem for a logic of *unordered* sets, shown in Figure 4. (Therefore, in this section, the elements should not be assumed to range over real numbers or integers, but over some arbitrary infinite set.) The grammar symbol F ranges over formulas, T over terms denoting real numbers, and S over terms denoting sets. Note that we allow mixed linear constraints on the cardinalities of sets, but the cardinalities of sets themselves are integers or infinite (unlike [PK08b] where cardinalities could be fractional).

The decidability of a quantified version of the logic in Figure 4, based on quantifier elimination and without fractional constraints on cardinalities, goes back to [FV59]. The elementary complexity for the quantified case was shown in [KNR06] (see Section 8.1 for the case of possibly infinite sets). Using quantifier elimination for mixed linear arithmetic [Wei99], we can also obtain the decidability of the quantified language in Figure 4. In this paper, we are interested in efficient bounds for the quantifier-free fragment.

$$F ::= F \wedge F \mid \neg F \mid S = S \mid T \leq T \mid \mathsf{card}(S){=}T \mid \mathsf{card}(S){\geq}\aleph_0 \mid T \in \mathbb{Z}$$
$$T ::= k \mid C \mid T + T \mid C * T$$
$$S ::= x \mid \emptyset \mid S \cap S \mid S^c$$

Fig. 4. The logic QFBAPA_∞ of sets with cardinalities. Here C denotes rational constants.

Using a sparse solution theorem for integer linear arithmetic [ES06] the NP membership (and, trivially, NP-hardness) for a simpler version of this logic was shown in [KR07]. However, previous statements of this complexity result considered interpretation where all sets are *finite*. The main claim of this section is that the NP membership remains even if we allow sets to be *infinite*, introduce a predicate to check finiteness, and allow the constraints on cardinalities to be embedded into constraints on linear real arithmetic.

As usual, by a Venn region over variables $x_1, \ldots, x_n$ we mean an intersection containing for each set x_i either x_i or its complement x_i^c. There are exactly 2^n Venn regions over n set variables.

Theorem 1. *Let F be a QFBAPA_∞ formula (Figure 4) with free set variables $x_1, \ldots, x_n$ containing at most d atomic formulas ($d \geq 2$). Then F is satisfiable iff there exist*

- *at most $O(d\log(d))$ Venn regions $r_1, \ldots, r_N$ over variables $x_1, \ldots, x_n$ and*
- *non-negative integers $k_1, \ldots, k_N$ and $a_1, \ldots, a_N \in \{0, 1\}$ whose number of bits is polynomial in the size of formula F*

such that F is true in some valuation in which for each i where $1 \leq i \leq N$ either $a_i = 0$ and $\mathsf{card}(r_i){=}k_i$, or $a_i = 1$ and $\mathsf{card}(r_i){\geq}\aleph_0$, and such that the valuation of all Venn regions other than $r_1, \ldots, r_N$ is the empty set.

The idea of the proof is to encode cardinalities of possibly infinite set with pairs of integer variables. We represent a finite set of size p by $(0, p)$. We represent an infinite set by pairs of the form $(1, p)$ where p is arbitrary. We then perform a similar but slightly more involved construction to the one in [KR07].

Proof of Theorem 1 Note that all set algebra equations can be encoded using cardinalities using the fact that $A = B$ iff $\mathsf{card}((A \cap B^c) \cup (A^c \cap B)){=}0$. We can thus assume that set variables occur only in atomic formulas of the form $\mathsf{card}(s){=}t$ and $\mathsf{card}(s){\geq}\aleph_0$. For each set expression s_i in F introduce fresh variables p_i and q_i. Consider the formula

$$\bigwedge_i \big((q_i = 0 \wedge \mathsf{card}(s_i){=}p_i) \vee (q_i = 1 \wedge \mathsf{card}(s_i){\geq}\aleph_0)\big) \qquad (**)$$

Replace each $\mathsf{card}(s_i){=}t$ in F with $(q_i = 0 \wedge p_i = t)$ and replace each $\mathsf{card}(s_i){\geq}\aleph_0$ with $q_i = 1$. Then conjoin the result with $(**)$; we obtain a formula of the form $P \wedge (**)$ where P is a mixed linear arithmetic formula. The original input formula is equivalent to $\exists (p_i)_i (q_i)_i (P \wedge (**))$. Indeed, given the values of sets in the original formula, we assign $q_i = 0$ if s_i is finite and $q_i = 1$ if it is infinite. If s_i is finite, we assign p_i to its size, otherwise we assign p_i to 0. Then $\mathsf{card}(s_i){=}t$ and $(q_i = 0 \wedge p_i = t)$ evaluate to the same truth value, and so do $\mathsf{card}(s_i){\geq}\aleph_0$ and $q_i = 1$. Conversely, if $(**)$ holds then $q_i = 0$ if s_i is finite and $q_i = 1$ if s_i is infinite, so the original and new atomic formulas again evaluate to the same truth values. Note that for s_i infinite, both $\mathsf{card}(s_i){=}p_i$ and $(q_i = 0 \wedge p_i = t)$ are false, regardless of the value of p_i. Thus, given an assignment for which $q_i = 1$, if we change the value of p_i the truth value of $(**)$ or P remains the same.

We next observe that $(**)$ is equisatisfiable with a formula in quantifier-free Presburger arithmetic, extending the construction for finite cardinalities of [KR07]. For each Venn region r_i over the set variables occurring in $(**)$, we introduce two fresh non-negative integer variables a_i and k_i, analogously to q_i and p_i. The intended interpretation is again

$$\bigwedge_j \big((a_j = 0 \wedge \mathsf{card}(r_j){=}k_j) \vee (a_j = 1 \wedge \mathsf{card}(r_j){\geq}\aleph_0)\big) \qquad (V)$$

We next use the property that the cardinality of a finite union of disjoint sets is the sum of the cardinalities of sets. We obtain the following linear integer constraint

$$\bigwedge_i \big(\sum_j \gamma_{ij} a_j \geq q_i \;\wedge\; \sum_j \gamma_{ij} k_j = p_i \big) \qquad (I)$$

where $\gamma_{i,j}$ equals 1 if $r_j \subseteq s_i$ is valid and 0 otherwise.

We claim that $(**)$ and (I) have equivalent sets of solution vectors for variables q_j, p_j. Two solutions $(q_j, p_j)_j$ and $(q'_j, p'_j)_j$ are equivalent if $q_j = q'_j$ and if $q_j = 0$ then $p_j = p'_j$.

$(\Rightarrow)$: In one direction, given the values of set variables such that $(**)$ holds, define a_j and k_j according to (V). Consider an arbitrary conjunct number i in the conjunction (I). Consider first the case when s_i is a finite set. Then all Venn regions contained in s_i are finite, so for all j for which $\gamma_{ij} = 1$ we have $a_j = 0$ and p_j denotes the size of the finite Venn region r_j. We also have $q_i = 0$ so $\sum_j \gamma_{ij} a_j \geq q_i$ reduces to $0 \geq 0$. The condition $\gamma_{ij} k_j = p_i$ holds because the size of a union of disjoint sets is equal to sums of the sizes of the sets. In the second case, s_i is infinite. Then $q_i = 1$. Because s_i is a finite union of Venn regions, there must be a Venn region r_j that is infinite. Therefore, there exists j such that $\gamma_{ij} = 1$ and $a_j = 1$. This ensures that $\sum_j \gamma_{ij} a_j \geq q_i$ holds. Given that $q_i = 1$, we can find an equivalent solution with arbitrary values of p'_j, so we can make $\sum_j \gamma_{ij} k_j = p_i$ hold as well.

$(\Leftarrow)$: In the other direction, suppose we have the values of variables a_i and k_i. We then define as Venn regions disjoint sets such that (V) holds. The proof that $(**)$ holds is straightforward.

This completes the proof of equisatisfiability of $(**)$ and (I). We are left with checking the satisfiability of $P \wedge (I)$. Transform P into disjunctive normal form $\bigvee_j D_j \cdot (\boldsymbol{k}, \boldsymbol{a}, \boldsymbol{v}) \geq e_j$, where each disjunct is a mixed linear programming problem and all coefficients in D_j are integers. The formula is satisfiable iff one of the disjuncts is satisfiable, so consider one disjunct $D_j \cdot (\boldsymbol{k}, \boldsymbol{a}, \boldsymbol{v}) \geq e_j$. As in [PK08b], let $\boldsymbol{v} = \boldsymbol{l} + \boldsymbol{f}$ where $\boldsymbol{l}$ are fresh integer variables, and $\boldsymbol{f}$ are real variables restricted to $[0, 1)$. Splitting D_j into two groups of columns $[D'_j, D''_j]$ we obtain a problem of the form $D''_j \boldsymbol{f} \geq b - D'_j \cdot (\boldsymbol{k}, \boldsymbol{a}) - D''_j \boldsymbol{l}$. Note that the right-hand side is integer so we can replace the left-hand side with g where $g = \lfloor D''_j \boldsymbol{f} \rfloor$. Because $\boldsymbol{f}$ has components from $[0, 1)$, the vector g is bounded by the norm of the matrix D''_j, and the values of g are representable by polynomially many bits. For each value of g, the original problem decomposes into one polynomial sized mixed linear programming problem $D''_j \boldsymbol{f} \geq g$, and one integer linear programming problem

$$(I) \ \wedge \ (g \geq b - D'_j \cdot (\boldsymbol{k}, \boldsymbol{a}) - D''_j \boldsymbol{l}) \qquad (K)$$

Note that (I) is an integer linear programming problem with exponentially many variables (a_j, k_j). However, there are only polynomially many constraints in (I) (two for each expression s_i in the input formula) and therefore polynomially many in (K). Moreover, also the coefficients γ_{ij} are bounded. After introducing polynomially many slack variables for the inequations in (I), we obtain (K') that has the form arising in [KR07] from constraints over finite sets, and has sparse solutions with only $O(d \log d)$ variables a_i and k_i non-empty. The non-empty Venn regions $r_1, \ldots, r_N$ are precisely those Venn regions r_i for which $a_i \neq 0 \vee k_i \neq 0$. □

If we know that sparse solutions exist, it is not necessary to generate an exponentially large integer linear programming problem; we can encode in quantifier-free Presburger arithmetic both the guessing of which regions are non-zero, and checking whether the constraints on sets are satisfiable for this guess. We therefore obtain the extension of the result in [KR07] to infinite sets and to mixed linear constraints on cardinalities.

Corollary 1. *Satisfiability for* QFBAPA_∞ *(Figure 4) is NP-complete.*

4 Ordered, Possibly Infinite Sets with Cardinalities

$$\begin{aligned}
F ::= \; & F \wedge F \mid \neg F \mid S = S \mid T \leq T \mid S < S \mid T \in S \mid S \subseteq \mathbb{Z} \mid T \in \mathbb{Z} \\
& \mathsf{card}(S) \geq \aleph_0 \mid \mathsf{card}(S) = T \mid \mathsf{inf}(S) = -\infty \mid \mathsf{inf}(S) = T \mid \mathsf{sup}(S) = \infty \mid \mathsf{sup}(S) = T \\
T ::= \; & k \mid C \mid T + T \mid C * T \\
S ::= \; & x \mid \emptyset \mid S \cap S \mid S^c \mid \mathsf{take}(T, S) \mid \mathsf{lrange}(T, T, S)
\end{aligned}$$

Fig. 5. The logic $\mathsf{QFBAPA}^{<}_{\infty}$ of ordered sets with cardinalities, infima and suprema. Here C denotes rational constants.

Figure 5 shows the syntax of $\mathsf{QFBAPA}^{<}_{\infty}$, our quantifier-free logic of sets of real numbers supporting integer sets and variables, linear arithmetic, the cardinality operator, infimum, and supremum. In this section we show our main result, which is a reduction of $\mathsf{QFBAPA}^{<}_{\infty}$ to QFBAPA_{∞}.

The predicates $T \in \mathbb{Z}$ and $A \subseteq \mathbb{Z}$ denote that T is an integer term and that A is a set containing only integers, respectively. $\mathsf{card}(A){\ge}\aleph_0$ means that A is an infinite set. The predicate $\mathsf{card}(A){=}T$ means that A is finite and its cardinality is the non-negative integer term T. The predicate $\mathsf{inf}(A){=}{-\infty}$ means that A has no lower bound, whereas $\mathsf{inf}(A){=}T$ means that the infimum of A is T. Analogously, $\mathsf{sup}(A){=}\infty$ means that A has no upper bound, whereas $\mathsf{sup}(A){=}T$ means that the supremum of A is T. As a special case, for finite non-empty sets the operations $\mathsf{card}, \mathsf{inf}, \mathsf{sup}$ correspond, respectively, to: the number of elements, the least element, and the greatest element. For the empty set, we define the infimum and supremum predicates to be false: we assume $\neg(\mathsf{inf}(\emptyset){=}{-\infty})$ and $\neg(\mathsf{sup}(\emptyset){=}\infty)$, as well as $\neg(\mathsf{inf}(\emptyset){=}v)$ and $\neg(\mathsf{sup}(\emptyset){=}T)$ for every term T. Note that the logic can define arbitrary propositional operations, the subset relation ($A \subseteq B$ is $A \cap B = A$), and all set algebra operations (using $\cap$ and the complement S^c). We define the partial order on finite sets, denoted $A{<}B$ by $\mathsf{sup}(A){=}k_A \wedge k_A < k_B \wedge \mathsf{inf}(B){=}k_B$, where k_A and k_B are fresh integer variables. Define $\mathsf{take}(T, S) = A$ as $\mathsf{card}(A){=}T \wedge A {<} (S \setminus A)$. We define $\mathsf{lrange}(T_1, T_2, S) = B$ as a shorthand for $B = A_2 \setminus A_1 \wedge A_2 = \mathsf{take}(T_2, S) \wedge A_1 = \mathsf{take}(T_1 - 1, S)$, where A_1, A_2 are fresh set variables.

4.1 A Decision Procedure for $\mathsf{QFBAPA}^{<}_{\infty}$

We next describe our procedure for reducing $\mathsf{QFBAPA}^{<}_{\infty}$ to BAPA. When we reduce fresh variables we assume that they are existentially quantified at the top level. The satisfiability of the resulting formula will thus reduce to satisfiability of a quantifier-free formula.

Rewritings. If k_{f} and A_{f} are fresh variables we replace $T \in S$ with

$$k_{\mathsf{f}} = T \wedge \mathsf{card}(A_{\mathsf{f}}){=}1 \wedge \mathsf{inf}(A_{\mathsf{f}}){=}k_{\mathsf{f}} \wedge \mathsf{sup}(A_{\mathsf{f}}){=}k_{\mathsf{f}} \wedge A_{\mathsf{f}} \subseteq S.$$

We also replace negative occurrences of the predicates $\mathsf{card}(S){\ge}\aleph_0$, $\mathsf{card}(S){=}T$, $\mathsf{inf}(S){=}{-\infty}$, $\mathsf{inf}(S){=}T$, $\mathsf{sup}(S){=}\infty$, $\mathsf{sup}(S){=}T$ with positive occurrences by introducing fresh variables and disjunctions. For example, we replace $\neg(\mathsf{inf}(S){=}{-\infty})$ with $(S = \emptyset \vee \mathsf{inf}(S){=}k)$ for k fresh.

In the sequel we present an algorithm for *conjunctions* F of literals; formulas of arbitrary boolean structure can be handled by e.g. rewriting them into disjunctive normal form or using an approach analogous to DPLL(T) [GHN+04]. Some of the transformations below introduce new disjunctions by "guessing"; we assume that the process of selecting one disjunct is applied implicitly.

Decomposition into integers and non-integers sets. For each set variable A, we introduce two fresh variables $A^{\mathbb{Z}}$ and $A^{\mathbb{R}\setminus\mathbb{Z}}$, and add the constraints that $A = A^{\mathbb{Z}} \cup A^{\mathbb{R}\setminus\mathbb{Z}} \wedge A^{\mathbb{Z}} \cap A^{\mathbb{R}\setminus\mathbb{Z}} = \emptyset$. If the constraint $A \subseteq \mathbb{Z}$ appears in F, we replace it with $A^{\mathbb{R}\setminus\mathbb{Z}} = \emptyset$. This step effectively partitions every set variable into an integer and a non-integer part. In our reduction to QFBAPA_{∞}, we will use this partitioning to encode the constraints on the integer and real parts of sets separately. In the following when we refer to "any set variable A", we refer to all set variables, including the ones introduced at this step.

Purification. We flatten all arguments to the predicates $\mathsf{card}(S){\geq}\aleph_0$, $\mathsf{card}(S){=}T$, $\mathsf{inf}(S){=}{-\infty}$, $\mathsf{inf}(S){=}T$, $\mathsf{sup}(S){=}\infty$, $\mathsf{sup}(S){=}T$, $A \subseteq \mathbb{Z}$ and $T \in \mathbb{Z}$ by introducing fresh variables and new equalities when an argument is not already a variable. Then, for each set variable A such that $\mathsf{inf}(A){=}{-\infty}$ or $\mathsf{sup}(A){=}\infty$ appears in the formula, we add the constraint $\mathsf{card}(A){\geq}\aleph_0$.

Guessing the empty and the infinite sets. For each set variable A, we guess whether A is empty, non-empty finite, or infinite. In each case, we add a constraint to F: If it is empty, we add the constraint $\mathsf{card}(A){=}0$. If it is infinite, we add the constraint $\mathsf{card}(A){\geq}\aleph_0$. If is it finite but non-empty, we add the constraint $\mathsf{card}(A){=}k_\mathsf{f} \wedge k_\mathsf{f} \geq 1$, where k_f is fresh.

Guessing the unbounded sets. For each set that we guessed was infinite, we now guess whether it admits an infimum and a supremum. Similarly to the previous step, we add the constraints $\mathsf{inf}(A){=}{-\infty}$, $\mathsf{sup}(A){=}\infty$, $\mathsf{inf}(A){=}k_\mathsf{f}$ and $\mathsf{sup}(A){=}k_\mathsf{f}$ to F as needed. We note that at the end of this step, there is a variable for the cardinality of each finite set appearing in the formula, as well as a variable for each bounded infimum and supremum, when they exist.

Guessing an ordering on bounds. We consider the set $\boldsymbol{B}$ of all numeric variables appearing in the predicates inf and sup as well as all integer and rational constants appearing in F. We guess an arrangement into (equality) equivalence classes of all variables and constants in $\boldsymbol{B}$. We then guess a total ordering between these equality classes. We number the classes in increasing order from 1 to n. We add to F equality constraints between members of the same equivalence class. For convenience, we also introduce a fresh variable b_i in each class. We finally add to F all constraints $b_i < b_{i+1}$.

Segmentation of the domain. For each of the equivalence classes, we create two fresh set variables $C_i^{\mathbb{Z}}$ and $C_i^{\mathbb{R}\setminus\mathbb{Z}}$. We then guess whether b_i is an integer or a non-integer value. In the first case, we add the constraint $b_i \in \mathbb{Z} \wedge \mathsf{card}(C_i^{\mathbb{Z}}){=}1 \wedge \mathsf{card}(C_i^{\mathbb{R}\setminus\mathbb{Z}}){=}0$ to F. Otherwise, we add the constraint $\mathsf{card}(C_i^{\mathbb{Z}}){=}0 \wedge \mathsf{card}(C_i^{\mathbb{R}\setminus\mathbb{Z}}){=}1$. We then create $2 \cdot (n+1)$ more fresh set variables $C_{n+1}^{\mathbb{Z}}$ to $C_{2n+1}^{\mathbb{Z}}$ and $C_{n+1}^{\mathbb{R}\setminus\mathbb{Z}}$ to $C_{2n+1}^{\mathbb{R}\setminus\mathbb{Z}}$. We make all the fresh set variables disjoint by adding to F the constraints $\mathsf{card}(C_i^{\star_1} \cap C_j^{\star_2}){=}0$ for $0 \leq i < j \leq 2n+1$ and $\star_1, \star_2 \in \{\mathbb{Z}, \mathbb{R}\setminus\mathbb{Z}\}$. In the following we interpret the fresh sets as points ($C_0^\star$ to $C_n^\star$) and intervals ($C_{n+1}^\star$ to $C_{2n+1}^\star$) on $\mathbb{R}$ (see Figure 6). As for our set variables previously, the points and intervals are split between integer and non-integer values. Note that in our interpretation, the introduced sets are all infinite, except for the sets of integers $C_{n+2}^{\mathbb{Z}}$ to $C_{2n}^{\mathbb{Z}}$. We therefore add to F the constraints

$$\begin{aligned}
\mathsf{card}(C_i^{\mathbb{Z}}){=}k_\mathsf{f}^i \wedge k_\mathsf{f}^i = \lceil b_i - b_{i-1} \rceil - 1 \quad & \text{for } n+2 \leq i \leq 2n \\
\mathsf{card}(C_i^{\mathbb{Z}}){\geq}\aleph_0 \quad & \text{for other values of } i \\
\mathsf{card}(C_i^{\mathbb{R}\setminus\mathbb{Z}}){\geq}\aleph_0 \quad & \text{for all values of } i
\end{aligned}$$

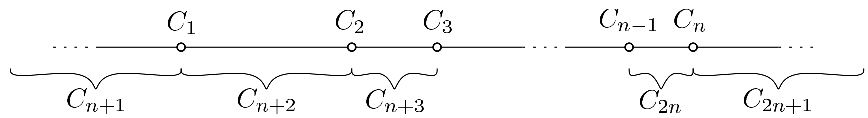

Fig. 6. Segmentation of $\mathbb{R}$. We use C_i here to denote $C_i^{\mathbb{Z}} \cup C_i^{\mathbb{R}\setminus\mathbb{Z}}$.

where k_{f}^i are fresh integer variables. (Note that we can encode the first equality without $\lceil \quad \rceil$ by using inequalities.) Following our interpretation, we express the set variables of F using the point and interval sets. For each non-empty set $A^\star$ ($\star \in \{\mathbb{Z}, \mathbb{Z} \setminus \mathbb{R}\}$), we introduce $2n+1$ fresh variables $A_i^\star$ representing the intersection of $A^\star$ with $C_i^\star$. $A^\star$ can then be expressed as the (disjoint) union of all variables $A_i^\star$:

$$\bigwedge_{0<i\leq 2n+1} A_i^\star = A^\star \cap C_i^\star \quad \wedge \quad A^\star = \bigcup_{0<i\leq 2n+1} A_i^\star$$

We now need to express that some of these intersections are empty. If we guessed that $A^\star$ admitted an infimum in the equivalence class p we add the constraints $\bigwedge_{n+1\leq i\leq n+p} A_i^\star \cap C_i^\star = \emptyset$, and similarly for sets that admit a supremum.

Solution of the QFBAPA_∞ constraints. As a final step, we remove from F all occurrences of the predicates inf and sup. We observe that the resulting formula is in QFBAPA_∞. We can use the results from Section 3 to determine its satisfiability. Our original formula is satisfiable if and only if F is satisfiable.

Theorem 2. *The decision procedure described above is sound and complete.*

Proof. **Soundness.** We first show that each of our reasoning steps results in a logically sound conclusion. The rewritings are correct by definition. Splitting each set into a partition is sound and the added constraints on the partition are consequences. The purification process introduces fresh variables that are constrained to be equal to the term they represent, so any model for the formula without the fresh variables can trivially be extended to the original variables. The constraints that we add after guesses are immediate consequences of these. It remains to show that when we introduce the fresh variables $C_i^\star$ and $A_i^\star$ and the constraints on them, we do not exclude any solution for the existing variables. To show this, consider a model for all non-fresh variables, and consider the ordering of equivalence classes following from the values of the infima and suprema of these sets. For any set $A^\star$, we can build the sets $A_i^\star$ as in the construction by splitting $A^\star$ into subsets with bounds defined by the values in the equivalence classes. We can then construct the sets $C_i^\star$ by taking the union of all sets $A_i^\star$ (each time for a fixed i). Note that all the sets $C_i^\star$ are disjoint.

Completeness. To show completeness, we need to show that we can build a model for the original formula from a model for our formula in QFBAPA_∞. The model for the reduced formula will contain values for all the bounds b_i as well as the cardinality of each set (and more generally of each Venn region), and we need to extend this model by populating the sets with elements from $\mathbb{R}$. We start by populating the singleton sets $C_i^\star$, for $0 < i \leq n$. If the value for b_i is an integer, we set in the extended model $C_i^{\mathbb{Z}} = \{b_i\}$, $C_i^{\mathbb{R}\setminus\mathbb{Z}} = \emptyset$, and the model for the case where b_i is not an integer is built similarly (note that the decision procedure will always return values that are in $\mathbb{Q}$). We then proceed to populate the sets C_{n+i}, for $0 < i \leq n+1$. We know the cardinality of each Venn region of such a set, and these regions are by definition disjoint. In the case of the integer sets $C_{n+i}^{\mathbb{Z}}$, we simply pick distinct integers for each Venn region. We know this is always possible because we encoded all the relevant cardinality constraints in the QFBAPA_∞ formula. We can pick integers in any order, because our construction ensures that no ordering constraints concern elements *within* a set $C_{n+i}^{\mathbb{Z}}$. As a result, $C_{n+i}^{\mathbb{Z}}$ will always contain all integers from b_{i-1} to b_i. For the Venn regions of the non-integer sets $C_{n+i}^{\mathbb{R}\setminus\mathbb{Z}}$, the construction is slightly more involved. Because $C_{n+i}^{\mathbb{R}\setminus\mathbb{Z}}$ is infinite, we know that at least one of its Venn regions is infinite as well, and the model for the QFBAPA_∞ formula will encode this. We name this region V_I. For a set $C_{n+i}^{\mathbb{R}\setminus\mathbb{Z}}$ with m distinct Venn regions, infimum b_{i-1} and supremum b_i, we can populate the j^{th} Venn region V_j (with $V_j \neq V_I$) as follows. If V_j has finite cardinality k, we then define

$$V_j = \bigcup_{1\leq l\leq\lceil\frac{k}{2}\rceil}\left\{b_i + \frac{\varepsilon}{m\cdot l + j}\right\} \cup \bigcup_{1\leq l\leq k-\lceil\frac{k}{2}\rceil}\left\{b_{i+1} - \frac{\varepsilon}{m\cdot l+j}\right\}$$

If the cardinality of V_j is required to be infinite, we define V_j as the corresponding countable set $V_j = \bigcup_{l\in\mathbb{N}}\left\{b_i + \frac{\varepsilon}{m\cdot l+j}\right\} \cup \bigcup_{l\in\mathbb{N}}\left\{b_{i+1} - \frac{\varepsilon}{m\cdot l+j}\right\}$. In both cases, we define ε as $\frac{1}{2}\cdot\min(1, \min_{0<i<n}(b_{i+1}, b_i))$ (ε is half the width of the smallest interval). Note that this generates distinct non-integer values for all the Venn regions because $m\cdot l + j \neq m\cdot l' + j'$ whenever $0 \leq j < j' < m$. For infinite sets, these values will converge to both ends of the interval represented by $C_i^{\mathbb{R}\setminus\mathbb{Z}}$. This means that constraints on infima and suprema outside of particular set will always be satisfied. We define the identified infinite region V_I to be $(b_{i-1}, b_i) \setminus \bigcup_{0<j\leq k, j\neq I} V_j$ (i.e. the open interval corresponding to $C_{n+i}^{\mathbb{R}\setminus\mathbb{Z}}$ minus all other Venn regions). This set has uncountably many elements and is also dense towards its extreme points. The construction for the four intervals that are open either to the left or to the right is similar. This concludes the construction of the elements of the $C_i^\star$ sets. The construction for all the other sets follows then from their definitions in terms of the $C_i^\star$ sets.

Complexity. QFBAPA_∞ is NP-hard because it subsumes propositional logic. The above reduction to QFBAPA_∞ runs in NP-time. By Corollary 1, the satisfiability problem for $\mathsf{QFBAPA}_\infty^<$ is NP-complete.

5 An Extension of the Calculus of Data Structures

We can now state an extension of Theorem 5 from [KPSW10] with the result on ordered collections.

Theorem 3. *There exist BAPA reductions for the following logics 1) WS2S [TW68], 2) two-variable logic with counting over finite models (C^2) [PH05, PST00], 3) Bernays-Schönfinkel-Ramsey over finite models [Ram30], 4) quantifier-free multisets with cardinality constraints [PK08a], 5) term algebras with the* content *function [SDK10], 6) the logic* $\mathsf{QFBAPA}^{<}_{\infty}$ *in Figure 5. Thus, quantifier-free set-sharing combination of all these logics is decidable.*

6 Conclusions

We had previously identified a number of uses for constraints on sets and cardinality bounds and established their optimal complexity. In this paper we generalized these results to: a) infinite sets b) the case of a total, possibly dense, ordering relation on collection elements. In particular, we have looked at collections of numerical elements: in this context, constraints on cardinalities are naturally combined with constraints on minimal and maximal elements. In each case, we have shown that the NP-completeness complexity of the decision problem was preserved in the extension.

We have shown that these steps beyond uninterpreted elements provide important benefits: using this new expressive power, we were able to precisely specify the contracts of functions manipulating ordered data structures. We have also shown that the added expressiveness promises to make synthesis specifications more precise and the synthesized code more predictable. Finally, in addition to the uses of the presented decision procedure alone, the fact that the decision procedure works as a reduction to BAPA [WPK09] means that they can be naturally combined with a number of other logics such as WS1S [TW68] two-variable logic with counting [PST00], BAPA extensions [YPK10], and certain recursive functions over algebraic data types [SDK10]. Therefore, it presents another building block towards a rich decidable language useful in verification, synthesis, and automated reasoning.

Acknowledgements. We thank Yuri Gurevich for providing in 2008 helpful references on the decidability of the theories of total orders as well as the IJCAR and CSL reviewers for their feedback. We thank Robin Steiger and Utkarsh Upadhyay who have, in the meantime, implemented a decision procedure for finite sets of integers with the cardinality operator and made it more efficient.

References

ES06. Friedrich Eisenbrand and Gennady Shmonin. Carathéodory bounds for integer cones. *Operations Research Letters*, 34(5):564–568, September 2006. `http://dx.doi.org/10.1016/j.orl.2005.09.008`.

FV59. S. Feferman and R. L. Vaught. The first order properties of products of algebraic systems. *Fundamenta Mathematicae*, 47:57–103, 1959.

Ghi05. Silvio Ghilardi. Model theoretic methods in combined constraint satisfiability. *Journal of Automated Reasoning*, 33(3-4):221–249, 2005.

GHN+04. Harald Ganzinger, George Hagen, Robert Nieuwenhuis, Albert Oliveras, and Cesare Tinelli. DPLL(T): Fast decision procedures. In *CAV*, 2004.

KMPS10. Viktor Kuncak, Mikaël Mayer, Ruzica Piskac, and Philippe Suter. Complete functional synthesis. In *PLDI*, 2010.

KNR06. Viktor Kuncak, Hai Huu Nguyen, and Martin Rinard. Deciding Boolean Algebra with Presburger Arithmetic. *J. of Automated Reasoning*, 2006.

KPSW10. Viktor Kuncak, Ruzica Piskac, Philippe Suter, and Thomas Wies. Building a calculus of data structures. In *Verification, Model Checking, and Abstract Interpretation (VMCAI)*, 2010.

KR07. Viktor Kuncak and Martin Rinard. Towards efficient satisfiability checking for Boolean Algebra with Presburger Arithmetic. In *CADE-21*, 2007.

Lae68. H. Laeuchli. A decision procedure for the weak second order theory of linear order. *Studies in Logic and the Foundat. of Math.*, 50:189–197, 1968.

MW80. Zohar Manna and Richard Waldinger. A deductive approach to program synthesis. *ACM Trans. Program. Lang. Syst.*, 2(1):90–121, 1980.

OSV08. Martin Odersky, Lex Spoon, and Bill Venners. *Programming in Scala: a comprehensive step-by-step guide*. Artima Press, 2008.

PH05. Ian Pratt-Hartmann. Complexity of the two-variable fragment with counting quantifiers. *Journal of Logic, Language and Information*, 14(3):369–395, 2005.

PK08a. Ruzica Piskac and Viktor Kuncak. Decision procedures for multisets with cardinality constraints. In *VMCAI*, number 4905 in LNCS, 2008.

PK08b. Ruzica Piskac and Viktor Kuncak. Fractional collections with cardinality bounds. In *Computer Science Logic (CSL)*, 2008.

PK08c. Ruzica Piskac and Viktor Kuncak. Linear arithmetic with stars. In *CAV*, 2008.

PST00. Leszek Pacholski, Wieslaw Szwast, and Lidia Tendera. Complexity results for first-order two-variable logic with counting. *SIAM J. on Computing*, 29(4):1083–1117, 2000.

Rab69. Michael O. Rabin. Decidability of second-order theories and automata on infinite trees. *Trans. Amer. Math. Soc.*, 141:1–35, 1969.

Ram30. F. P. Ramsey. On a problem of formal logic. *Proc. London Math. Soc.*, s2-30:264–286, 1930. doi:10.1112/plms/s2-30.1.264.

RRZ05. Silvio Ranise, Christophe Ringeissen, and Calogero G. Zarba. Combining data structures with nonstably infinite theories using many-sorted logic. In *FroCoS*, 2005.

SDK10. Philippe Suter, Mirco Dotta, and Viktor Kuncak. Decision procedures for algebraic data types with abstractions. In *POPL*, 2010.

She75. Saharon Shelah. The monadic theory of order. *The Annals of Mathematics of Mathematics*, 102(3):379–419, Nov 1975.

SS07. Viorica Sofronie-Stokkermans. Hierarchical and modular reasoning in complex theories: The case of local theory extensions. In *FroCoS '07*, 2007.

TW68. J. W. Thatcher and J. B. Wright. Generalized finite automata theory with an application to a decision problem of second-order logic. *Mathematical Systems Theory*, 2(1):57–81, August 1968.

Wei99. Volker Weispfenning. Mixed real-integer linear quantifier elimination. In *ISSAC*, pages 129–136, 1999.

WPK09. Thomas Wies, Ruzica Piskac, and Viktor Kuncak. Combining theories with shared set operations. In *Frontiers in Combining Systems*, 2009.

YPK10. Kuat Yessenov, Ruzica Piskac, and Viktor Kuncak. Collections, cardinalities, and relations. In *VMCAI*, 2010.