Towards Efficient Satisfiability Checking for Boolean Algebra with Presburger Arithmetic

Viktor Kuncak and Martin Rinard

Abstract. Boolean Algebra with Presburger Arithmetic (BAPA) is a decidable logic that combines 1) Boolean algebras of sets of uninterpreted elements (BA) and 2) Presburger arithmetic operations (PA). BAPA can express relationships between integer variables and cardinalities of unbounded sets. In combination with other decision procedures and theorem provers, BAPA is useful for verifying quantitative properties of data structures. Motivated by the observation that many queries in program analysis and verification are quantifier-free formulas, this paper examines QFBAPA, the quantifier-free fragment of BAPA. The computational complexity of QFBAPA satisfiability was previously unknown. Previous QFBAPA algorithms have non-deterministic exponential time complexity due to an explosion in the number of introduced integer variables.

This paper shows, for the first time, how to avoid such exponential explosion. We present an algorithm for checking satisfiability of QFBAPA formulas by reducing them to formulas of quantifier-free Presburger arithmetic, with only O(n log(n)) increase in formula size. We prove the correctness of our algorithm using a theorem about sparse solutions of integer linear programming problems. This proves that QFBAPA satisfiability is in NP and therefore NP-complete. We implemented our algorithm; we describe its initial deployment in the Jahob verification system and discuss its performance.

1 Introduction

This paper considers the satisfiability problem for a logic that allows reasoning about sets and their cardinalities. We call this logic quantifier-free Boolean Algebra with Presburger Arithmetic and denote it QFBAPA. Our motivation for QFBAPA is proving the validity of formulas arising from program verification [15, 16, 17], but QFBAPA constraints also occur in mechanized set theory [9], constraint data bases [28,29], as a fragment of other logics [23, 25, 1] and in the semantic analysis of natural language [20]. Figure 1 shows the syntax of QFBAPA. The logic contains 1) arbitrary boolean algebra (BA) expressions denoting sets, supporting operations such as union, intersection and complement, 2) arbitrary quantifier-free Presburger arithmetic (PA) expressions, supporting addition of integers and multiplication by constants, and 3) a cardinality operator $|B|$ for computing the the size of a BA expression B and treating it as a PA expression. The constant MAXC denotes the size of the finite universal set $\mathcal{U}$, so $|\mathcal{U}| = \mathsf{MAXC}$. The expression $K \,\mathsf{dvd}\, T$ means that an integer constant K divides an integer expression T, whereas B^c denotes the complement of the set B.

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

Fig. 1. Quantifier-Free Boolean Algebra with Presburger Arithmetic (QFBAPA)

1.1 Using QFBAPA in Software Verification

We implemented the algorithm described in this paper in the Jahob data structure verification system [15]. Figure 2 shows some of the verification conditions expressible in QFBAPA that we encountered and proved using our decision procedure. (For more examples, see [17, Chapters 2 and 7].) The formulas in Figure 2 are in HOL syntax, where cardinality of a set is denoted by `card`. Jahob soundly maps such formulas into stronger BAPA, using a simple syntactic translation that represents individual variables as singleton sets and approximates constructs unsupported by BAPA.

Section 7 describes our preliminary experience, which suggests that, for the more complex examples, our new QFBAPA decision procedure generates smaller formulas than the original BAPA decision procedure. When proving formula validity, this reduction does not yet translate to smaller running times. However, our decision procedure seems to have advantage for finding counterexamples of large formulas. We believe that the underlying results are interesting in their own right, and, given the exponential reduction in formula size for larger formulas, they can be viewed as a first step towards future scalable decision procedures for QFBAPA.

	verification condition	property being checked
1	$x \notin \mathtt{content} \land \mathtt{size} = \mathtt{card}\,\mathtt{content} \longrightarrow$ $(\mathtt{size} = 0 \leftrightarrow \mathtt{content} = \emptyset)$	using invariant on size to prove correctness of an efficient emptiness check
2	$x \notin \mathtt{content} \land \mathtt{size} = \mathtt{card}\,\mathtt{content} \longrightarrow$ $\mathtt{size} + 1 = \mathtt{card}(\{x\} \cup \mathtt{content})$	maintaining correct size when inserting fresh element
3	$\mathtt{size} = \mathtt{card}\,\mathtt{content} \land$ $\mathtt{size1} = \mathtt{card}(\{x\} \cup \mathtt{content}) \longrightarrow$ $\mathtt{size} \le \mathtt{size1} \land \mathtt{size1} \le \mathtt{size} + 1$	maintaining size after inserting any element
4	$\mathtt{content} \subseteq \mathtt{alloc} \land$ $x_1 \notin \mathtt{alloc} \land$ $x_2 \notin \mathtt{alloc} \cup \{x_1\} \land$ $x_3 \notin \mathtt{alloc} \cup \{x_1\} \cup \{x_2\} \longrightarrow$ $\mathtt{card}\,(\mathtt{content} \cup \{x_1\} \cup \{x_2\} \cup \{x_3\}) =$ $\mathtt{card}\,\mathtt{content} + 3$	allocating and inserting three objects into a countainer
5	$\mathtt{content} \subseteq \mathtt{alloc0} \land x_1 \notin \mathtt{alloc} \land$ $\mathtt{alloc0} \cup \{x_1\} \subseteq \mathtt{alloc1} \land x_2 \notin \mathtt{alloc1} \land$ $\mathtt{alloc1} \cup \{x_2\} \subseteq \mathtt{alloc2} \land x_3 \notin \mathtt{alloc2} \longrightarrow$ $\mathtt{card}\,(\mathtt{content} \cup \{x_1\} \cup \{x_2\} \cup \{x_3\}) =$ $\mathtt{card}\,\mathtt{content} + 3$	allocating and inserting three objects into a countainer while potentially performing other allocations
6	$x \in C \land C_1 = (C \setminus \{x\}) \land$ $\mathtt{card}(\mathtt{alloc1} \setminus \mathtt{alloc0}) \le 1 \land$ $\mathtt{card}(\mathtt{alloc2} \setminus \mathtt{alloc1}) \le \mathtt{card}\,C_1 \longrightarrow$ $\mathtt{card}\,(\mathtt{alloc2} \setminus \mathtt{alloc0}) \le \mathtt{card}\,C$	bound on the number of allocated objects in a recursive function that incorporates container C into another container

Fig. 2. Example verification conditions proved using the QFBAPA decision procedure

1.2 QFBAPA and BAPA

The logic QFBAPA is the quantifier-free fragment of Boolean Algebra with Presburger Arithmetic (BAPA). In addition to the constructs in Figure 1, full BAPA supports arbitrary set and integer quantifiers. Feferman and Vaught [12, Section 8, Page 90] showed the decidability of a variant of BAPA and used it to show the decidability of generalized products of first-order structures. In [16, 17] we formalize a decision procedure for BAPA and show that BAPA has the same complexity as PA, namely alternating doubly exponential time with a linear number of alternations, denoted $\mathsf{STA}(*, 2^{2^{n^{O(1)}}}, n)$ in [4], [14, Lecture 24].

BAPA admits quantifier elimination, which implies that QFBAPA formulas define the same class of relations on sets and integers as BAPA formulas, so they essentially have the same expressive power. Quantifier elimination also makes BAPA interesting as a potential shared language for combining multiple reasoning procedures [13]. In general, QFBAPA formulas may be exponentially larger than the equivalent quantified BAPA formulas with same free variables. However, it is often the case that the proof obligation (or other problem of interest) is already expressed in quantifier-free form. It is therefore interesting to consider the complexity of the satisfiability problem for QFBAPA.

1.3 Challenges in checking QFBAPA satisfiability

QFBAPA satisfiability is clearly NP-hard, because QFBAPA supports arbitrary propositional operators. Moreover, QFBAPA contains Boolean algebra of sets, which has its own propositional structure, so even the satisfiability of individual atomic formulas is NP-hard. The challenge is therefore proving the membership in NP. Membership in NP means that there are short certificates for satisfiability of QFBAPA formulas, or, dually, that invalid QFBAPA formulas have short counterexamples. Despite the widespread occurrence of QFBAPA constraints, this result was not known until now. To understand why existing approaches fail to establish membership in NP, consider the following example QFBAPA formula:

$$|\mathcal{U}| = 100 \; \wedge \bigwedge_{0 \le i < j \le 10} |x_i \cup x_j| = 30 \wedge \bigwedge_{0 \le i \le 10} |x_i| = 20 \qquad (E)$$

Explicitly specifying set contents. The formula (E) has 10 set variables. Each of these variables represents a subset of the universe of 100 elements. Therefore, a straightforward certificate of satisfiability of this QFBAPA formula would require 100 bits indicating whether each element is in the set, which is a certificate exponential in the size of the formula because we assume that 100 is represented using $\log 100$ bits. Such certificates therefore yield merely a membership of QFBAPA in NEXPTIME. Note that, even if we restrict the constants K in QFBAPA language to be 0 and 1, Presburger arithmetic expressions such as $k_1 = 1$, $k_{i+1} = k_i + k_i$ can efficiently encode large constants. Fundamentally, the reason we are interested in large set cardinalities is because they arise from small model theorem for Presburger arithmetic; supporting them is necessary for verifying symbolic cardinality bounds and constraints such as $|x \cap y| = |z|$.

Abstraction using sizes of partitions. An alternative approach to examining set interpretations up to a certain size is to consider a complete partitioning of sets into disjoint Venn regions $x_1^c \cap \ldots \cap x_{10}^c$, $x_1^c \cap \ldots \cap x_{10}$, $\ldots$, $x_1 \cap \ldots \cap x_{10}$, and introduce one non-negative integer variable for the size each of these partitions, yielding 2^{10} variables $l_{0,\ldots,0}, l_{0,\ldots,1}, \ldots, l_{1,\ldots,1}$. We can then represent cardinality of any set expression as a sum of finitely many of of these integer variables. This approach is widely known [23], [9, Chapter 11] and is often used to illustrate the very idea of Venn diagrams [34]. It has the advantage of not being exponential in the cardinalities of sets, because it reasons about these cardinalities symbolically. It also naturally integrates with the PA structure of QFBAPA and allows reducing QFBAPA to quantifier-free PA, as we explain below. Unfortunately, its direct use introduces a number of non-negative integer variables that is exponential in the number of sets. This approach is the essence of previous algorithms for for QFBAPA [36, 28, 23] and appears as a special case of our algorithm for quantified BAPA [16, 17]. All these algorithms would yield exponentially large certificates for satisfiability of QFBAPA, specifying the values of exponentially many non-negative integer variables.

1.4 Our Results

We can summarize the results of this paper as follows:

1. The key contribution of this paper is an encoding of QFBAPA formulas into polynomially-sized quantifier-free PA formulas. Instead of using exponentially many Venn region cardinality variables $l_{0,\ldots,0}, l_{0,\ldots,1}, \ldots, l_{1,\ldots,1}$, we use polynomially many "generic" variables along with polynomially many indices that determine the region that each generic variable represents. In the example (E) above, we would use introduce $N = 502$ of generic integer variables $l_{p^i_1,\ldots,p^i_{10}}$ for $1 \leq i \leq N$ that are a function of propositional variables $(p^i_1, \ldots, p^i_{10}) \in \{0,1\}^{10}$ for $1 \leq i \leq N$. We assume that the remaining Venn regions are all empty, which allows us to express any set expression b as a sum of those of the N integer variables $l_{p^i_1,\ldots,p^i_{10}}$ whose indices $p^i_1, \ldots, p^i_{10}$ identify Venn regions that belong to b.
2. The computation of a sufficient polynomial value for N is the second contributions of this paper. We start with the result [11] that if an element is in an integer cone generated by a set of vectors X of dimension d, then it is also in an integer cone generated by a "small" subset of X of size $N(d)$. This result implies that a system of equations with bounded coefficients, if satisfiable, has a *sparse solution* with only polynomially many non-zero variables, even if the number of variables in the system is exponential. As a consequence, every satisfiable QFBAPA formula has a witness of polynomial size, which indicates the values of integer variables in the original QFBAPA formula, lists the Venn regions that are non-empty, and indicates the cardinalities of these non-empty regions.

   This application of [11] gives the memership of QFBAPA in NP, but, given the NP-hardness of satisfiability of the generated formulas, it is desirable to obtain as tight bound on $N(d)$ as possible. We make the following steps towards the computation of a precise bound:
   (a) we compute the exact bound $N(d) = d$ for $d \leq 3$;
   (b) we identify a lower bound $N(d) \geq d + \left\lfloor \frac{d}{4} \right\rfloor$ for $d \geq 4$;
   (c) we provide several equivalent characterizations of vectors that achieve the optimal bound for any d, introducing the notion of a "non-redundant integer code generator" (NICG);
   (d) we provide a more precise bound in the presence of cardinality constraints of the form $|b| \leq c$ and $|b| = c$ for a small constant c.
3. We also describe the implemention of our algorithm in the context of the Jahob verification system. We evaluate the algorithm on the examples in Figure 2 and their variations.

Our previously reported results. We suggested the possibility of the existence of sparse solutions in the final version of [17], where we also established the complexity of quantified BAPA. In a previous technical report [21] we identified a PSPACE algorithm for QFBAPA, but the techniques used there are different and not needed for the results of this paper. We became aware of the result [11] in November 2006. A preliminary version of the current result is described in [15, Section 7.9].

2 Constructing Small Presburger Arithmetic Formulas

Given a QFBAPA formula, this section shows how to construct an associated polynomially larger quantifier-free PA formula. Section 3 then proves that the constructed formula is equisatisfiable with the original one.

Consider an arbitrary QFBAPA formula in the syntax of Figure 1. To analyze the problem, we first separate PA and BA parts of the formula by replacing $b_1 = b_2$ with $b_1 \subseteq b_2 \wedge b_2 \subseteq b_1$, replacing $b_1 \subseteq b_2$ with $|b_1 \cap b_2^c| = 0$, and then introducing integer variables k_i for all cardinality expressions $|b_i|$ occurring in the formula. With a constant increase in size, we obtain an equisatisfiable QFBAPA formula of the form $G \wedge F$ where G is a quantifier-free PA formula and F is of the form

$$\bigwedge_{i=0}^{p} |b_i| = k_i \tag{1}$$

We assume $b_0 = \mathcal{U}$ and $k_0 = \mathsf{MAXC}$, i.e., the first constraint is $|\mathcal{U}| = \mathsf{MAXC}$.

Let $y_1, \ldots, y_e$ be the set variables in $b_1, \ldots, b_p$. If we view each Boolean algebra formula b_i as a propositional formula, then for $\beta = (p_1, \ldots, p_e)$ where $p_i \in \{0,1\}$ let $[\![b_i]\!]_\beta \in \{0,1\}$ denote the truth value of b_i under the propositional valuation assigning the truth value p_i to the variable y_i. Let further s_β denote the Venn region associated with β, given by $s_\beta = \cap_{j=1}^{e} y_j^{p_j}$ where $y_j^0 = y_j^c$ is set complement and $y_j^1 = y_j$. Because b_i is a disjoint union of its corresponding Venn regions, we have $|b_i| = \sum_{\beta \models b_i} |s_\beta|$. For the sake of analysis, for each $\beta \in \{0,1\}^e$ introduce a non-negative integer variable l_β denoting $|s_\beta|$. Then (1) is equisatisfiable with the exponentially larger PA formula

$$\bigwedge_{i=0}^{p} \sum \left\{ l_\beta \mid \beta \in \{0,1\}^e \wedge [\![b_i]\!]_\beta{=}1 \right\} = k_i \tag{2}$$

Instead of this exponentially large formula where β ranges over all 2^e propositional assignments, the idea of our paper is to check the satisfiability of an asymptotically smaller formula

$$\bigwedge_{i=0}^{p} \sum \left\{ l_\beta \mid \beta \in \{\beta_1, \ldots, \beta_N\} \wedge [\![b_i]\!]_\beta{=}1 \right\} = k_i \tag{3}$$

where β ranges over a set of N assignments $\beta_1, \ldots, \beta_N$ for $\beta_i = (p_{i1}, \ldots, p_{ie})$ and p_{ij} are fresh free variables ranging over $\{0,1\}$. Let $d = p + 1$. We are interested in the best upper bound $N(d)$ on the number of non-zero Venn regions over all possible systems of equations. In the sequel we show that $N(d)$ is polynomial in d and therefore polynomial in the size of the original QFBAPA formula. This result implies that QFBAPA is in NP and gives an effective bound on how to construct a quantifier-free PA formula for checking the satisfiability of a given QFBAPA formula.

Encoding generic cardinality variables in PA. Formula (3) uses some PA constructs along with some meta-notation. We next explain how to write (3) as a polynomially large quantifier-free PA formula. Because there are only N distinct assignments β_j considered, we introduce one variable l_j for each $1 \leq j \leq N$, for a total of N integer variables. Let $c_{ij} = [\![b_i]\!]_{\beta_j}$ for $1 \leq i \leq p$ and $1 \leq j \leq N$. Then each conjunct of (3) becomes $\sum_{j=1}^{N} c_{ij} l_j = k_i$. It therefore suffices to show how to efficiently express sums with boolean variable (as opposed to constant) coefficients. For this we can use the standard conditional expression $\texttt{ite}(p, t_1, t_2)$, where p is a propositional formula and t_1, t_2 are integer terms. The $\texttt{ite}(p, t_1, t_2)$ expression evaluates to t_1 when p evaluates to true, and evaluates to t_2 when p evaluates to false. It can be efficiently eliminated by flattening the formula to contain no nested terms and then replacing $t = \texttt{ite}(p, t_1, t_2)$ with the formula $(p \rightarrow t = t_1) \wedge (\neg p \rightarrow t = t_2)$. (It is also directly available in the SMT-LIB format [27] and in the UCLID [19] decision procedure.) Using ite, we can express $c_{ij} l_j$ as $\texttt{ite}(c_{ij}, l_j, 0)$. Then (3) becomes $\bigwedge_{i=0}^{p} \sum_{j=1}^{N} \texttt{ite}([\![b_i]\!]_{\beta_j}, l_j, 0) = k_i$. Note that we can substitute the values k_i back into the original PA formula G, so there is no need to peform the separation into $G \wedge F$ in practice. We obtain the following simple summary of our algorithm:

$$\textbf{substitute each expression } |b_i| \textbf{ with } \sum_{j=1}^{N} \texttt{ite}([\![b_i]\!]_{\beta_j}, l_j, 0)$$

Note that this translation of QFBAPA into PA is parameterized by N. Sufficiently large values of N guarantee soundness and are the subject of the following sections, which show that a polynomial value suffices. However, any value of N can be used to try to prove the existence of a satisfying assignment for QFBAPA formulas, because a satisfying assignment for N_0 implies the existence of satisfying assignments for all $N \geq N_0$, letting $l_j = 0$ for $N_0 + 1 \leq j \leq N$.

3 Upper Bound on the Number of Non-Zero Venn Regions

We next prove that the number $N(d)$ of non-zero Venn regions can be assumed to be polynomial in d. Let $\mathbb{Z}$ denote the set of integers and $\mathbb{Z}_{\geq 0}$ denote the set of non-negative integers. We write $\sum X$ for $\sum\limits_{y \in X} y$.

Definition 1. *For $X \subseteq \mathbb{Z}^d$ a set of integer vectors, let*

$$\text{int\_cone}(X) = \{\lambda_1 x_1 + \ldots + \lambda_t x_t \mid t \geq 0 \wedge x_1, \ldots, x_t \in X \wedge \lambda_1, \ldots, \lambda_n \in \mathbb{Z}_{\geq 0}\}$$

denote the set of all non-negative integer linear combination of vectors from X.

To prove the bound on the number N of non-empty Venn regions from Section 2, we use a variation of the following result, established as Theorem 1(ii) in [11].

Fact 1 (Eisenbrand, Shmonina (2005)) *Let $X \subseteq \mathbb{Z}^d$ be a finite set of integer vectors and $M = \max\{(\max_{i=1}^{d} |x_j^i|) \mid (x_j^1, \ldots, x_j^d) \in X\}$ be the bound on the coordinates of vectors in X. If $b \in \text{int\_cone}(X)$, then there exists a subset $\tilde{X} \subseteq X$ such that $b \in \text{int\_cone}(\tilde{X})$ and $|\tilde{X}| \leq 2d \log(4dM)$.*

To apply Fact 1 to formula (2), let $X = \{x_\beta \mid \beta \in \{0,1\}^e\}$ where $x_\beta \in \{0,1\}^e$ is given by

$$x_\beta = ([\![b_0]\!]_\beta, [\![b_1]\!]_\beta, \ldots, [\![b_e]\!]_\beta).$$

Fact 1 implies is that if $(k_0, k_1, \ldots, k_p) \in \mathrm{int\_cone}(X)$ where k_i are as in formula (2), then $(k_0, k_1, \ldots, k_p) \in \mathrm{int\_cone}(\tilde{X})$ where $|\tilde{X}| = 2d \log(4d)$ (note that $M = 1$ because x_β are $\{0, 1\}$-vectors). The subset $\tilde{X}$ corresponds to selecting a polynomial subset of N Venn region cardinality variables l_β and assuming that the remaining ones are zero. This implies that formulas (2) and (3) are equisatisfiable.

A direct application of Fact 1 yields $N = 2d \log(4d)$ bound, which is sufficient to prove that QFBAPA is in NP. However, because this bound is not tight, in the sequel we prove results that slightly strengthen the bound and provide additional insight into the problem.

4 Nonredundant Integer Cone Generators and Upper Bound

Definition 2. *Let X be a set of integer vectors. We say that X is a* nonredundant integer cone generator *for b, and write NICG(X, b), if $b \in \mathrm{int\_cone}(X)$, and for every $y \in X$, $b \notin \mathrm{int\_cone}(X \setminus \{y\})$.*

Lemma 1 says that if NICG(X, b) for some b, then the sums of vectors $\sum Y$ for $Y \subseteq X$ are uniquely generated elements of $\mathrm{int\_cone}(X)$.

Lemma 1. *Suppose NICG(X, b). If $\lambda_1, \lambda_2 : X \to \mathbb{Z}_{\geq 0}$ are non-negative integer coefficients for vectors in X such that*

$$\sum_{x \in X} \lambda_1(x)x = \sum_{x \in X} \lambda_2(x)x \tag{4}$$

and $\lambda_1(x) \in \{0, 1\}$ for all $x \in X$, then $\lambda_2 = \lambda_1$.

Proof. Suppose NICG(X, b), $\lambda_1, \lambda_2 : X \to \mathbb{Z}_{\geq 0}$ are such that (4) holds and $\lambda_1(x) \in \{0, 1\}$ for all $x \in X$, but $\lambda_2 \neq \lambda_1$. If there are vectors x on the left-hand side of (4) that also appear on the right-hand side, we can cancel them. We obtain an equality of the form (4) for distinct λ'_1, λ'_2 with the additional property that $\lambda'_1(x) = 1$ implies $\lambda'_2(x) = 0$. Moreover, not all $\lambda'_1(x)$ are equal to zero. By $b \in \mathrm{int\_cone}(X)$, let $\lambda : X \to \mathbb{Z}_{\geq 0}$ be such that $b = \sum_{x \in X} \lambda(x)x$. Let x_0 be such that $\lambda'_1(x_0) = \min\{\lambda(x) \mid \lambda'_1(x) = 1\}$. By construction, $\lambda'_1(x_0) = 1$ and $\lambda'_2(x_0) = 0$. We then have, with x in sums ranging over X:

$$\begin{aligned} b &= \sum_{\lambda'_1(x)=1} \lambda(x)x + \sum_{\lambda'_1(x)=0} \lambda(x)x \\ &= \sum_{\lambda'_1(x)=1} (\lambda(x) - \lambda(x_0))x + \lambda(x_0) \sum_{\lambda'_1(x)=1} x + \sum_{\lambda'_1(x)=0} \lambda(x)x \\ &= \sum_{\lambda'_1(x)=1} (\lambda(x) - \lambda(x_0))x + \lambda(x_0) \sum \lambda'_2(x)x + \sum_{\lambda'_1(x)=0} \lambda(x)x \end{aligned}$$

In the last sum, the coefficient next to x_0 is zero in all three terms. We conclude $b \in \mathrm{int\_cone}(X \setminus \{x_0\})$, contradicting NICG$(X, b)$. ■

We write NICG(X) as a shorthand for NICG$(X, \sum X)$. Theorem 1 gives several equivalent characterizations of NICG(X). The equivalence of 1) and 4) is interesting because it justifies the use of NICG(X) independently of the generated vector b.

Theorem 1. *Let $X \subseteq \{0, 1\}^d$. The following statements are equivalent:*

1) *there exists a vector $b \in \mathbb{Z}^d_{\geq 0}$ such that NICG(X, b);*

2) *If $\lambda_1, \lambda_2 : X \to \mathbb{Z}_{\geq 0}$ are non-negative integer coefficients for vectors in X such that*

$$\sum_{x \in X} \lambda_1(x)x = \sum_{x \in X} \lambda_2(x)x$$

and $\lambda_1(x) \in \{0, 1\}$ for all $x \in X$, then $\lambda_2 = \lambda_1$.

3) *For $\{x_1, \ldots, x_n\} = X$ (for $x_1, \ldots, x_n$ distinct), the system of d equations expressed in vector form as*

$$\lambda(x_1)x_1 + \ldots + \lambda(x_n)x_n = \sum X \tag{5}$$

has $(\lambda(x_1), \ldots, \lambda(x_n)) = (1, \ldots, 1)$ as the unique solution in $\mathbb{Z}^n_{\geq 0}$.

4) *NICG(X).*

Proof. $1) \to 2)$: This is Lemma 1.

$2) \to 3)$: Assume $2)$ and let $\lambda_1(x_i) = 1$ for $1 \leq i \leq n$. For any solution λ_2 we then have $\sum_{x \in X} \lambda_1(x)x = \sum_{x \in X} \lambda_2(x)x$, so $\lambda_2 = \lambda_1$. Therefore, λ_1 is the unique solution.

$3) \to 4)$: Assume $3)$. Clearly $\sum X \in \mathrm{int\_cone}(X)$; it remains to prove that X is minimal. Let $y \in X$. For the sake of contradiction, suppose $\sum X \in \mathrm{int\_cone}(X \setminus \{y\})$. Then there exists a solution $\lambda(x)$ for (5) with $\lambda(y) = 0 \neq 1$, a contradiction with the uniqueness of the solution.

$4) \to 1)$: Take $b = \sum X$. ∎

Corollary 1 is used in [11] to establish the bound on the size of X with NICG(X). We obtain it directly from Lemma 1 taking $\lambda_2(x) \in \{0, 1\}$.

Corollary 1. *If NICG(X) then for $Y_1, Y_2 \subseteq X$, $Y_1 \neq Y_2$ we have $\sum Y_1 \neq \sum Y_2$.*

The following lemma says that it suffices to establish bounds on the cardinality of X such that NICG(X), because they give bounds on all X.

Lemma 2. *If $b \in \mathrm{int\_cone}(X)$, then there exists a subset $\tilde{X} \subseteq X$ such that $b \in \mathrm{int\_cone}(\tilde{X})$ and NICG$(\tilde{X}, b)$.*

Proof. If $b \in \mathrm{int\_cone}(X)$ then by definition $b \in \mathrm{int\_cone}(X_0)$ for a finite $X_0 \subseteq X$. If not NICG(X_0, b), then $b \in \mathrm{int\_cone}(X_1)$ where X_1 is a proper subset of X_0. Continuing in this fashion we obtain a sequence $X_0 \supset X_1 \supset \ldots \supset X_k$ where $k \leq |X_0|$. The last element X_k satisfies NICG(X_k, b). ∎

Moreover, the property NICG(X) is hereditary, i.e. it applies to all subsets of a set that has it. (The reader familiar with matroids [35] might be interested to know that, for $d \geq 4$, the family of sets $\{X \subseteq \{0, 1\}^d \mid \text{NICG}(X)\}$ is not a matroid, because it contains multiple subset-maximal elements of different cardinality.)

Lemma 3. *If NICG(X) and $Y \subseteq X$, then NICG(Y).*

Proof. Suppose that NICG(X) and $Y \subseteq X$ but not NICG$(Y, \sum Y)$. Because $\sum Y \in \mathrm{int\_cone}(X)$, there is $z \in Y$ such that $\sum Y \in \mathrm{int\_cone}(Y \setminus \{z\})$. Then also $\sum Y \in \mathrm{int\_cone}(X \setminus \{z\})$, contradicting Lemma 1. ∎

The following theorem gives our bounds on $|X|$. As in [11], we only use Corollary 1 instead of the stronger Lemma 1, suggesting that the bound is not tight.

Theorem 2. *Let $X \subseteq \{0,1\}^d$ and NICG(X). Then $2^N \le (N+1)^d$, and, consequently,*

$$|X| \le (1+\varepsilon(d))(d \log d) \tag{6}$$

where $\varepsilon(d) \le 1$ for all $d \ge 1$, and $\lim\limits_{d\to\infty} \varepsilon(d) = 0$.

Proof. Let $X \subseteq \{0,1\}^d$, NICG(X) and $N = |X|$. We prove $2^N \le (N+1)^d$. Suppose that, on the contrary, $2^N > (N+1)^d$. If $\sum Y = (x^1, \ldots, x^d)$ for $Y \subseteq X$, then $0 \le x^j \le N$ because $Y \subseteq \{0,1\}^d$ and $|Y| \le N$. Therefore, there are only $(N+1)^d$ possible sums $\sum Y$. Because there are 2^N subsets $Y \subseteq X$, there exist two distinct subsets $U, V \in 2^X$ such that $\sum U = \sum V$. This contradicts Corollary 1. Therefore, $2^N \le (N+1)^d$, so $N \le d \log(N+1)$. From here we use elementary reasoning with inequalities to obtain $N \le 2d \log(2d)$ (see [11] or [17, Section 7.9.3] for details). Substituting this bound on N back into $N \le d\log(N+1)$ we obtain

$$\begin{aligned} N &\le d\log(N+1) \le d\log(2d\log(2d)+1) = d\log(2d(\log(2d)+\tfrac{1}{2d})) \\ &= d(1+\log d + \log(\log(2d)+\tfrac{1}{2d})) = d\log d(1+\tfrac{1+\log(\log(2d)+\frac{1}{2d})}{\log d}) \end{aligned}$$

so we can let $\varepsilon(d) = (1+\log(\log d + 1 + \frac{1}{2d}))/\log d$. ■

We can now define the function whose bounds we are interested in computing.

Definition 3. $N(d) = \max\{|X| \mid X \subseteq \{0,1\}^d \wedge \text{NICG}(X)\}$

Theorem 2 implies $N(d) \le (1+\varepsilon(d))(d \log d)$. However, because the function $N/\log(N+1)$ on integers is monotonic, we can efficiently compute its exact inverse by binary search.

5 Lower Bounds and Reals

Although we currently do not have tight bounds for $N(d)$, in this section we show, in sequence, the following observations about lower bounds for $N(d)$:

1. $d \le N(d)$ for all d;
2. $N_R(d) = d$ if we use real variables instead of integer variables;
3. $N(d) = d$ for $d \in \{1,2,3\}$;
4. for $d + \lfloor \frac{d}{4} \rfloor \le N(d)$ for $4 \le d$;

We first show $d \le N(d)$.

Lemma 4. *Let $X = \{(x_i^1, \ldots, x_i^d) \mid 1 \le i \le n\}$ and*

$$X^+ = \{(x_i^1, \ldots, x_i^d, 0) \mid 1 \le i \le n\} \cup \{(0, \ldots, 0, 1)\}$$

Then NICG(X) if and only if NICG(X^+).

Corollary 2. *$N(d) + 1 \le N(d+1)$ for all $d \ge 1$.*

Proof. Let $X \subseteq \{0,1\}^d$, NICG(X), and $|X| = N(d)$. Then NICG(X^+) by Lemma 4 and $|X^+| = N(d)+1$, which implies $N(d+1) \ge N(d)+1$. ■

Note that we have $N(1) = 1$ because there is only one non-zero $\{0, 1\}$ vector in one dimension. From Corollary 2 we obtain our lower bound, with standard basis as NICG.

Lemma 5. $d \leq N(d)$*. Specifically, NICG*$(\{e_1, \ldots, e_d\})$ *where* e_i *are unit vectors.*

Note that for $X = \{e_1, \ldots, e_d\}$ we have $\mathrm{int\_cone}(X) = \mathbb{Z}^d_{\geq 0}$, which implies that X is a *maximal* NICG, in the sense that no proper superset $W \supset X$ has the property NICG(W).

Real-valued relaxation of QFBAPA**.** In Appendix B we show that, if we use real or rational measure of set size instead of integer arithmetic, we obtain a nicer-behaved problem and the bound $N'(d) = d$ follows using well-known results in linear programming. We can use this technique as a sound (but incomplete) method for proving the absence of solutions of a QFBAPA formula. This approach is attractive both because the bound $N'(d) = d$ is smaller than the bound for integers, and because the decision procedure for real linear arithmetic is more efficient than for quantifier-free PA.

$\boldsymbol{N(d) = d}$ **for** $\boldsymbol{d \in \{1, 2, 3\}}$**.** We next show that for $d \in \{1, 2, 3\}$ not only $d \leq N(d)$ but also $N(d) \leq d$.

Lemma 6. $N(d) = d$ *for* $d \in \{1, 2, 3\}$.

Proof. By Corollary 2, if $N(d + 1) = d + 1$, then $N(d) + 1 \leq d + 1$ so $N(d) \leq n$. Therefore, $N(d) = 3$ implies $N(2) = 2$ as well, so we can take $d = 3$.

If $N(d) > d$, then there exists a set X with NICG(X) and $|X| > d$. From Lemma 3, a subset $X_0 \subseteq X$ with $|X| = d + 1$ also satisfies NICG(X_0). Therefore, $N(3) = 3$ is equivalent to showing that there is no set $X \subseteq \{0, 1\}^3$ with NICG(X) and $|X| = 4$.

Consider a possible counterexample $X = \{x_1, x_2, x_3, x_4\} \subseteq \{0, 1\}^3$ with $b \in X$. By previous argument on real-value relaxation, $N'(3) = 3$, so b is in convex cone of some three vectors from X, say $b \in \mathrm{cone}(\{x_1, x_3, x_3\})$. On the other hand, $b \notin \mathrm{int\_cone}(\{x_1, x_3, x_3\})$. If we consider a system $\lambda_1 x_1 + \lambda_2 x_2 + \lambda_3 x_3 = b$ this implies that such system has solution over non-negative reals, but not over non-negative integers. This can only happen if in the process of Gaussian elimination we obtain coefficients whose absolute value is more than 1. The only set of three vectors for which this can occur is $X_1 = \{(0, 1, 1), (1, 0, 1), (1, 1, 0)\}$ We then consider all possibilities for the fourth vector in X, which, modulo permutations of coordinates, are $(0, 0, 0)$, $(1, 1, 1)$, $(1, 1, 0)$, and $(1, 0, 0)$. However, adding any of these vectors violates the uniqueness of the solution to $\lambda_1 x_1 + \lambda_2 x_2 + \lambda_3 x_3 + \lambda_4 x_4 = \sum X$, so NICG$(X)$ does not hold by Theorem 1, condition 3). ■

$\boldsymbol{N = \frac{5}{4}d - \frac{3}{4}}$ **lower bound.** We next show that there exists an example $X_5 \subseteq \{0, 1\}^4$ with NICG(X_5) and $|X_5| = 5$. From this it follows that $N(d) > d$ for all $d \geq 4$.

Consider the following system of 4 equations with 5 variables, where all variable coefficients are in $\{0, 1\}$. (We found this example by narrowing down the search using the observations on minimal counterexamples in the proof of Lemma 6.)

$$\begin{array}{rcrcrcrcrcl}
\lambda_1 & + & \lambda_2 & + & \lambda_3 & & & & & = & 3 \\
 & & \lambda_2 & + & \lambda_3 & + & \lambda_4 & & & = & 3 \\
\lambda_1 & & & + & \lambda_3 & + & \lambda_4 & + & \lambda_5 & = & 4 \\
\lambda_1 & + & \lambda_2 & & & + & \lambda_4 & + & \lambda_5 & = & 4
\end{array} \qquad (7)$$

Performing Gaussian elimination yields an equivalent system

$$\begin{aligned}
\lambda_1 + \lambda_2 + \lambda_3 \qquad\qquad &= 3 \\
\lambda_2 + \lambda_3 \ \ + \lambda_4 \qquad &= 3 \\
\lambda_3 + 2\lambda_4 \ \ + \lambda_5 &= 4 \\
3\lambda_4 + 2\lambda_5 &= 5
\end{aligned}$$

From this form it easy to see that the system has $(\lambda_1, \lambda_2, \lambda_3, \lambda_4, \lambda_5) = (1, 1, 1, 1, 1)$ as *the only solution* in the space of non-negative integers. Note that all variables are non-zero in this solution. (In contrast, as discussed above, because the system is satisfiable, it must have a solution in non-negative reals where at most 4 coordinates are non-zero; an example of such solution is $(\lambda_1, \lambda_2, \lambda_3, \lambda_4, \lambda_5) = (0, 1.5, 1.5, 0, 2.5)$.) The five columns of the system (7) correspond to the set of vectors $X_5 = \{(1, 0, 1, 1), (1, 1, 0, 1), (1, 1, 1, 0), (0, 1, 1, 1), (0, 0, 1, 1)\}$ such that NICG(X_5). The set X_5 is also a maximal NICG, because adding any of the remaining 9 non-zero vectors in $\{0, 1\}^4 \setminus X_5$ results in a set that is not NICG.

This argument shows that there exist maximal NICG of size larger than d for $d \geq 4$. As we have remarked before, the set of d unit vectors is a maximal NICG for every d, which means that, unlike linearly independent sets of vectors over a field or other independent sets in a matroid [35], there are maximal NICG sets of different cardinality.

Note also that X_5 is not a Hilbert basis [31]. Namely, we have that $(1, 1, 1, 1) \in \text{cone}(X_5) \setminus \text{int\_cone}(X_5)$ because $(1, 1, 1, 1) = 1/3((1, 0, 1, 1) + (1, 1, 0, 1) + (1, 1, 1, 0) + (0, 1, 1, 1))$. This illustrates why previous results on Hilbert bases do not directly apply to the notion of NICG.

Using k identical copies of X_5 (with 4 equations in a group mentioning a disjoint set of 5 variables) we obtain systems of $4k$ equations with $5k$ variables such that the only solution is a vector $(1, \ldots, 1)$ of all ones. By adding p unit vector columns for $1 \leq p \leq 3$, we also obtain systems of $4k + p$ equations with $5k + p$ variables, with $N = \frac{5}{4}d - \frac{p}{4} = d + \lfloor \frac{d}{4} \rfloor \geq \frac{5}{4}d - \frac{3}{4}$, which, in particular, shows that $N = d$ upper bound is invalid for all $d \geq 4$.

6 Better Upper Bounds for Small Cardinalities

Consider a QFBAPA formula in separated form $G \wedge F$ as in Section 2, where G is a PA formula and F is given by (2). Our bounds on N so far a function of d alone. For many formulas arising in practice we can reduce N using bounds on the values that k_i can take, as explained in this section. In our experience, this improvement significantly reduced the overall running time of our algorithm.

Improved bound. Suppose that we can conclude that if the formula $F \wedge G$ is satisfiable, then there exists a satisfying assignments for variables where $0 \leq k_i \leq c_i$ (if we do not have a bound for some i, we let $c_i = \infty$). We can often obtain such a bound c_i by transforming G to negation-normal form and checking if k_i occurs in literals such as $k_i = 0$ or $k_i < c_i$. (It may also be possible to adapt the techniques for PA formulas from [32].) Given the bounds c_i, we have the following inequality that generalizes the one in Theorem 2:

$$2^N \leq \prod_{i=1}^{d} (1 + \min(c_i, N)) \tag{8}$$

The reasoning follows the proof of Theorem 2 and we sketch it below. By construction of c_i, consider a satisfying assignment with $k_i \leq c_i$, fix the values of k_i and consider a satisfying assignment for (2) that has a minimal number N of non-zero values for l_β, for some N. Consider a sum $\sum Y = (t_1, \ldots, t_d)$ of a subset Y of vectors X corresponding to N non-zero l_β. From $\sum X = (k_1, \ldots, k_d)$ and $Y \subseteq X$ we have $t_i \leq k_i \leq c_i$. On the other hand, $|Y| \leq N$, so $t_i \leq N$. Therefore, $t_i \leq \min(c_i, N)$. As in Theorem 2, the inequality (8) then follows from the requirement that all 2^N subsets Y have distinct sums with coordinates from 0 to $\min(c_i, N)$.

Consequences for common cases. Two common cases that we can easily take advantage of are bounds $c_i = 0$ and $c_i = 1$. Suppose that for $i \in I_0$ we have $c_i = 0$ and for $i \in I_1$ (where $I_1 \cap I_0 = \emptyset$) we have $c_i = 1$. Let $|I_0| = s_0$ and $|I_1| = s_1$. Letting $c_i = \infty$ for $i \notin I_0 \cup I_1$, from (8) we obtain $2^N \leq 2^{s_1}(N+1)^{d-s_0-s_1}$. For $c_i = 0$ and $c_i = 1$ we can in fact obtain a slightly stronger bound from the condition $2^N \leq 2^{s_1}(N - s_1 + 1)^{d-s_0-s_1}$, which can be justified as follows. Consider a satisfying assignment for $G \wedge F$. When $i \in I_0$, we can eliminate the equation $|b_i| = k_i$ in (2) and remove all l_β such that $[\![b_i]\!]_\beta{=}1$ from the remaining equations, while preserving the property that all vectors in the matrix corresponding to (2) are in $\{0, 1\}$. The bound on non-zero variables for the resulting system with $d - s_0$ equations therefore applies to the original system as well. Similarly, if $i \in I_1$ and the right-hand side $k_i = 1$, then we know that in the satisfying assignment there is exactly one β_1 such that $[\![b_i]\!]_{\beta_1}{=}1$, so we can remove the equation $|b_i| = 1$, and for all j such that $[\![b_j]\!]_{\beta_1}{=}1$ subtract 1 from k_j and remove l_{β_1}. The result is again a system with $\{0, 1\}$ coefficients, but one less equation. Increasing the bound for the resulting system by one (to account for $l_{\beta_1} = 1$) we obtain the bound for the original system, which proves our claim.

These observations are important in practice because they imply that pure boolean algebra expressions (such as $b_1 \subseteq b_2$ and $b_1 = b_2$) do not increase N when they occur positively. The bound $c_i = 1$ also frequently occurs in our examples because we encode elements as singleton sets; our result says that one such cardinality bound to a formula increases the number of needed integer variables only by one.

Note that, if all cardinality bounds are small constants, we obtain formulas simpler than QFBAPA and we can expect good results by encoding them with universally quantified formulas of first-order logic with equality without function symbols, as in [18, 7]. What we have shown in this section is that we can benefit from taking small constant cardinalities into account even if some of the cardinalities are large or symbolic.

7 Preliminary Experiments

Figure 3 shows formula sizes and running times for the original BAPA algorithm and our new QFBAPA algorithm. As benchmarks we used the formulas in Figure 2, as well as two of their variations. Namely, formulas in Figure 2 are nice-looking partly because we generated them using Jahob annotations that specify relevant assumptions needed to establish an assertion. Without such assertions, verification conditions contain tens of additional useless assumptions. Syntactically determining which assumptions are useful is a difficult problem [8], so it is reasonable to leave this task to the the decision procedure. Therefore, in examples 2a and 6a we added back 3 of those original assumptions. Moreover, we made 6a an invalid formula by changing $\leq$ in the goal into $<$. For both the original and the new reduction to PA we report the size of the generated PA formula, the time needed for SMT solver to prove its validity, and the overall running time that includes the conversion to PA. The conversion to PA is very fast for the QFBAPA algorithm, but is exponential in worst case for the BAPA algorithm (see, for example, 2a benchmark). We used CVC Lite [3] as the solver for PA. The QFBAPA algorithm computes N by inverting the monotonic function $N/\log(N+1)$ and by taking into account the optimizations in Section 6. For propositional variables that encode assignments $\beta_1, \ldots, \beta_N$, our QFBAPA algorithm implementation generates a symmetry breaking predicate that imposes a lexicographical order on these assignments; we found that this predicate reduces the running time of the PA solver several times.

For these particular examples we can conclude that for small QFBAPA formulas both the previous BAPA and the current QFBAPA approach are fast. For proving validity of formulas with a larger number of variables (benchmarks 2a, 5), the QFBAPA approach can generate smaller PA formulas (as expected), but the PA solver often fails to prove their validity, most likely due to a large number of additional propositional variables and `ite` statements.[1] Finally, for finding counterexamples, the example 6a suggests that the QFBAPA algorithm is already better. The BAPA algorithm takes over 30 seconds even to generate the PA formula. The QFBAPA algorithm computes $N = 18$ and finishes in 13.1 seconds. In fact, the QFBAPA algorithm can find counterexample even for $N = 2$, taking less than 0.1 seconds, suggesting that iterative search from $N = 1$ to the bound that guarantees soundness could be veru productive for finding counterexamples.

vc	BAPA			QFBAPA		
	PA size	PA time(s)	total time(s)	PA size	PA time(s)	total time(s)
1	39	< 0.1	< 0.1	190	< 0.1	< 0.1
2	57	< 0.1	< 0.1	220	< 0.1	< 0.1
$2a$	1049	0.7	2.0	840	18.6	18.8
3	51	< 0.1	< 0.1	131	< 0.1	0.1
4	546	0.3	0.6	1328	∞	∞
5	2386	3.2	14.8	1750	∞	∞
6	442	0.2	0.4	2613	∞	∞
$6a$	4251	∞	∞	4687	12.1	13.1

Fig. 3. Formula sizes and running times for formulas in Figure 2

8 Related Work

To our knowledge, our result is the only decision procedure for a logic with sets and cardinality constraints that does not explicitly construct all set partitions. Using a new form of small model property, the "small number of non-zero variables property", we obtained a non-deterministic polynomial-time algorithm that can be solved by producing polynomially large quantifier-free Presburger arithmetic formulas. A polynomial bound sufficient for NP membership can be derived from [11]. In addition to improvements in the bounds that take into account small cardinalities, we introduced the notion of non-redundant integer cone generators and established their properties. Note that previous results, such as [31], consider matroids and Hilbert bases. Non-redundant integer cone generators that we introduced seem the most natural notion for determining the bounds for sparse solutions problem. As we remark in Section 5, the sets of vectors X with NICG(X) do not form a matroid, and maximal NICG(X) need not be a Hilbert basis. Note also that the equations generated from QFBAPA problems are more diffi-

[1] In our experiments, CVC Lite would run out of memory. We also tried using with Yices which did not run out of memory, but still failed to complete in a reasonable amount of time.

cult than set packing and set partitioning problems [2] because integer variables are not restricted to be $\{0,1\}$.

Relationship to counting SAT. Although similarly looking, it turns out that QFBAPA and #SAT problem [33] are quite different; see Appendix A for details.

Presburger arithmetic. The matching lower and upper bounds for PA were shown in [4]. Our reduction to PA along with small model property for PA that follows from [24] yields polynomial encoding of QFBAPA into SAT, whose efficiency in practice would be interesting to explore in the future.

Reasoning about Sets. The quantifier-free fragment of BA is shown NP-complete in [22]; see [18] for a generalization of this result using the parameterized complexity of the Bernays-Schönfinkel-Ramsey class of first-order logic [6, Page 258]. [9] gives an overview of several fragments of set theory including theories with quantifiers but no cardinality constraints and theories with cardinality constraints but no quantification over sets. The decision procedure for quantifier-free fragment with cardinalities in [9, Chapter 11] introduces exponentially many integer variables to reduce the problem to PA.

Using first-order provers. With appropriate axioms and decision procedures, first-order provers can also be used to reason about QFBAPA-like constraints, as shown, for example, by SPASS+T [26]. Our decision procedure by itself is not nearly as widely applicable as SPASS+T, but is complete for its domain (for example, it proves a formulation of problem number (73) from [26] in 0.1 seconds whereas SPASS+T is reported to time out in the particular experiments performed in [26]). Our decision procedure can therefore be useful as a component of such more general systems or as guidance for choosing axioms on cardinalities of collections along with strategies on when they should be applied.

References

1. Franz Baader, Diego Calvanese, Deborah McGuinness, Daniele Nardi, and Peter Patel-Schneider, editors. *The Description Logic Handbook: Theory, Implementation and Applications*. CUP, 2003.
2. Egon Balas and Manfred W. Padberg. Set partitioning: A survey. *SIAM Review*, 18(4):710–760, 1976.
3. Clark Barrett and Sergey Berezin. CVC Lite: A new implementation of the cooperating validity checker. In *Proc.* 16^{th} *Int. Conf. on Computer Aided Verification (CAV '04)*, volume 3114 of *Lecture Notes in Computer Science*, pages 515–518, 2004.
4. Leonard Berman. The complexity of logical theories. *Theoretical Computer Science*, 11(1):71–77, 1980.
5. Dimitris Bertsimas and John N. Tsitsiklis. *Introduction to Linear Optimization*. Athena Scientific, Belmont, Massachusetts, 1997.
6. Egon Börger, Erich Grädel, and Yuri Gurevich. *The Classical Decision Problem*. Springer-Verlag, 1997.
7. Charles Bouillaguet, Viktor Kuncak, Thomas Wies, Karen Zee, and Martin Rinard. On using first-order theorem provers in a data structure verification system. Technical Report MIT-CSAIL-TR-2006-072, MIT, November 2006. http://hdl.handle.net/1721.1/34874.
8. Charles Bouillaguet, Viktor Kuncak, Thomas Wies, Karen Zee, and Martin Rinard. Using first-order theorem provers in a data structure verification system. In *VMCAI'07*, November 2007.
9. Domenico Cantone, Eugenio Omodeo, and Alberto Policriti. *Set Theory for Computing*. Springer, 2001.
10. W. J. Cook, J. Fonlupt, and A. Schrijver. An integer analogue of Carathéodory's theorem. *Journal of Combinatorial Theory, Series B*, 40(63–70), 1986.

11. Friedrich Eisenbrand and Gennady Shmonina. 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`.

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

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

14. Dexter Kozen. *Theory of Computation*. Springer, 2006.

15. Viktor Kuncak. *Modular Data Structure Verification*. PhD thesis, EECS Department, Massachusetts Institute of Technology, February 2007.

16. Viktor Kuncak, Hai Huu Nguyen, and Martin Rinard. An algorithm for deciding BAPA: Boolean Algebra with Presburger Arithmetic. In *20th International Conference on Automated Deduction, CADE-20*, Tallinn, Estonia, July 2005.

17. Viktor Kuncak, Hai Huu Nguyen, and Martin Rinard. Deciding Boolean Algebra with Presburger Arithmetic. *J. of Automated Reasoning*, 2006. `http://dx.doi.org/10.1007/s10817-006-9042-1`.

18. Viktor Kuncak and Martin Rinard. Decision procedures for set-valued fields. In *1st International Workshop on Abstract Interpretation of Object-Oriented Languages (AIOOL 2005)*, 2005.

19. Shuvendu K. Lahiri and Sanjit A. Seshia. The UCLID decision procedure. In *CAV'04*, 2004.

20. Iddo Lev. Precise understanding of natural language. Stanford Univeristy PhD dissertation draft, February 2007.

21. Bruno Marnette, Viktor Kuncak, and Martin Rinard. On algorithms and complexity for sets with cardinality constraints. Technical report, MIT CSAIL, August 2005.

22. Kim Marriott and Martin Odersky. Negative boolean constraints. Technical Report 94/203, Monash University, August 1994.

23. Hans Jürgen Ohlbach and Jana Koehler. How to extend a formal system with a boolean algebra component. In W. Bibel P.H. Schmidt, editor, *Automated Deduction. A Basis for Applications*, volume III, pages 57–75. Kluwer Academic Publishers, 1998.

24. Christos H. Papadimitriou. On the complexity of integer programming. *J. ACM*, 28(4):765–768, 1981.

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

26. Virgile Prevosto and Uwe Waldmann. SPASS+T. In *ESCoR: Empirically Successful Computerized Reasoning*, volume 192, 2006.

27. Silvio Ranise and Cesare Tinelli. The SMT-LIB Standard: Version 1.2. Technical report, Department of Computer Science, The University of Iowa, 2006. Available at `www.SMT-LIB.org`.

28. Peter Revesz. Quantifier-elimination for the first-order theory of boolean algebras with linear cardinality constraints. In *Proc. Advances in Databases and Information Systems (ADBIS'04)*, 2004.

29. Peter Z. Revesz. The expressivity of constraint query languages with boolean algebra linear cardinality constraints. In *ADBIS*, pages 167–182, 2005.

30. Alexander Schrijver. *Theory of Linear and Integer Programming*. John Wiley & Sons, 1998.

31. András Sebö. Hilbert bases, Caratheodory's theorem and combinatorial optimization. In R. Kannan and W. Pulleyblank, editors, *Integer Programming and Combinatorial Optimization I*. University of Waterloo Press, 1990.

32. Sanjit A. Seshia and Randal E. Bryant. Deciding quantifier-free presburger formulas using parameterized solution bounds. In *19th IEEE LICS*, 2004.

33. S. Toda. PP is as hard as the polynomial-time hierarchy. *SIAM Journal on Computing*, 20(5):865–877, 1991.

34. John Venn. On the diagrammatic and mechanical representation of propositions and reasonings. *Dublin Philosophical Magazine and Journal of Science*, 9(59):1–18, 1880.

35. H. Whitney. On the abstract properties of linear independence. *American Journal of Mathematics*, 57:509–533, 1935.

36. Calogero G. Zarba. Combining sets with cardinals. *J. of Automated Reasoning*, 34(1), 2005.

Appendix

A Counting SAT

The complexity of QFBAPA is perhaps even more interesting in the light of the similarity of QFBAPA to the #SAT problem of computing the number of satisfying assignments of a propositional formula. The #SAT problem generalizes propositional satisfiability (SAT) and is possibly more difficult [33]. Given a propositional formula a with propositional variables $p_1, \ldots, p_n$, consider instead the corresponding boolean algebra formula b with corresponding set variables $x_1, \ldots, x_n$, with $\wedge, \vee, \neg$ replaced by $\cap, \cup, {}^c$. Then checking $|\mathcal{U}| = 2^n \wedge |b| > 2^{n-1} - 1$ satisfiability corresponds to testing the most significant bit of the number of solutions of the propositional formula a, but only under the following condition $(*)$: *all* 2^n *partitions with variables* $x_1, \ldots, x_n$ *are non-empty*. If there was a formula that would encode the condition $(*)$ while being of size polynomial in n, then the formula $C \wedge |\mathcal{U}| = 2^n \wedge |b| > 2^{n-1} - 1$ would encode (the query version of) #SAT, so the existence of short satisfiability certificates for QFBAPA would be intimately tied to fundamental questions in complexity theory. The results of this paper imply that there is no such polynomially large formula C, and that there are, in fact, QFBAPA satisfiability certificates of polynomial size, independently of the relationship between #SAT and SAT.

B Quantifier-Free Boolean Algebra with (Real) Linear Arithmetic

It is interesting to observe that, for a variation of the QFBAPA problem over *real numbers*, which we call QFBALA (Quantifier-Free Boolean Algebra with Linear Arithmetic), we have $N'(d) = d$ as a lower *and upper* bound for every d.

We define QFBALA similarly as QFBAPA, but we use real (or rational) linear arithmetic instead of integer linear arithmetic and we interpret $|A|$ is some real-valued measure of the set A. A possible application of QFBALA are generalizations of probability consistency problems such as [5, Page 385, Example 8.3]. Set algebra operations then correspond to the σ-algebra of events, and the measure of the set is the probability of the event. Another model of QFBALA is to interpret sets as finite disjoint unions of half-open intervals $[a, b)$ contained in $[0, 1)$, and let $|A|$ be the sum of the lengths of the disjoint intervals making up A.

The conditions we are using on the models are 1) for two disjoint sets A, B, we have $|A \cup B| = |A| + |B|$, 2) if $|C| = p$ and $0 \leq q \leq p$, then there exists $B \subseteq C$ such that $|B| = q$, and 3) (for simplicity) if $A \neq \emptyset$, then $|A| > 0$.

We can reduce the satisfiability of QFBALA to the satisfiability of a conjunction of a quantifier-free linear arithmetic formula over reals and a formula of the form (2) but with l_β non-negative real values instead of non-negative integer values. We then reduce formula (2) to a formula of the form (3). The question is then, what can we use as the bound $N'(d)$ for QFBALA problems? This question reduces to following. Define convex cone generated by a set of vectors by

$$\operatorname{cone}(X) = \{\lambda_1 x_1 + \ldots + \lambda_t x_t \mid t \geq 0 \wedge x_1, \ldots, x_t \in X \wedge \lambda_1, \ldots, \lambda_n \geq 0\}$$

where $\lambda_1, \ldots, \lambda_n \in \mathbb{R}$ are non-negative real coefficients. If $b \in \operatorname{cone}(X)$, what bound can we put on the cardinality of a subset $\tilde{X} \subseteq X$ such that $X \in \operatorname{cone}(\tilde{X})$? Note that d is a lower bound, using the same example of unit vectors as X. In the case of real numbers, Carathéodory's theorem [10] states that d is an upper bound as well: $b \in \operatorname{cone}(\tilde{X})$ for some $\tilde{X}$ of cardinality at most d.

We can also explain that $N'(d) = d$ using the terminology of linear programming [30]. The equations (2) along with $l_\beta \geq 0$ for $\beta \in \{0,1\}^e$ determine a polytope in $\mathbb{R}^{2^e}$, so if they have a solution, they have a solution that is a vertex of the polytope. The vertex in $\mathbb{R}^{2^e}$ is the intersection of 2^e hyperplanes, of which at most d are given by (2), so the remaining ones must be hyperplanes of the form $l_\beta = 0$. This implies that at least $2^e - d$ coordinates of the vertex are zero and at most d of them can be non-zero.