Towards Efficient Satisfiability Checking for Boolean Algebra with Presburger Arithmetic

Viktor Kuncak[1] and Martin Rinard[2]

[1] Ecole Politechnique Fédérale de Lausanne, Lausanne, VD, Switzerland
[2] Massachusetts Institute of Technology, Cambridge, MA, USA

Abstract. Boolean Algebra with Presburger Arithmetic (BAPA) is a decidable logic that combines 1) Boolean algebra of sets of uninterpreted elements (BA) and 2) Presburger arithmetic (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 automatically verifying quantitative properties of data structures. This paper examines QFBAPA, the quantifier-free fragment of BAPA. The computational complexity of QFBAPA satisfiability was previously unknown; previous QFBAPA algorithms have nondeterministic 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 PA, 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 is the first proof that QFBAPA satisfiability is in NP and therefore NP-complete. We implemented our algorithm in the context of the Jahob verification system. Our preliminary experiments suggest that our algorithm, although not necessarily better for proving formula unsatisfiability, is more effective in detecting formula satisfiability than previous approaches.

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 [11,12,13], but QFBAPA constraints also occur in mechanized set theory [7], constraint data bases [23,24], as a fragment of other logics [18,20,1] and in the semantic analysis of natural language [15]. 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)

Using QFBAPA in software verification. We implemented the algorithm described in this paper in the Jahob data structure verification system [11]. Figure 2 shows some of the verification conditions expressible in QFBAPA that we encountered and proved using our decision procedure. In these verification conditions, sets such as `content`, C, and C_1 represent the contents of dynamically allocated data structures. (For more examples, see [13, 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 5 describes our preliminary experience with using our algorithm on formulas such as those in Figure 2, showing that the new algorithm is promising for detecting formula satisfiability.

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 [8, 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 [12,13] we formalize a decision procedure for BAPA and show that BAPA has the same complexity as the complexity of Presburger arithmetic (PA), namely alternating doubly exponential time with a linear number of alternations, denoted $\mathsf{STA}(*, 2^{2^{n^{O(1)}}}, n)$ in [4], [10, 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 [9].

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

Fig. 2. Examples proved using our QFBAPA decision procedure

1.1 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:

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

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 requires 100 bits for each set to indicate whether each element is in the set. Such

certificate is therefore exponential in the size of the formula (we assume that 100 is represented using $\log_2 100$ 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 [19]; supporting them is necessary for verifying symbolic cardinality bounds and constraints such as $|x \cap y| = |z|$.

Abstraction using sizes of partitions. An alternative 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 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 these integer variables. This approach is widely known [18], [7, Chapter 11] and is often used to illustrate the very idea of Venn diagrams. 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 integer variables that is exponential in the number of sets. This approach is the essence of previous algorithms for QFBAPA [28,23,18] and appears as a special case of our algorithm for quantified BAPA [12,13]. All these algorithms would yield exponentially large certificates for satisfiability of QFBAPA, specifying the values of exponentially many integer variables.

1.2 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, which has 56 equations, we would introduce $N = g(56) = 502$ generic integer variables $l_{p_1^i,\ldots,p_{10}^i}$ for $1 \leq i \leq N$ that are a function of propositional variables $(p_1^i, \ldots, p_{10}^i) \in \{0,1\}^{10}$ for $1 \leq i \leq N$. The polynomially bounded function g is given by the equation (6) below. We assume that the remaining $2^{10} - g(56)$ Venn regions are all empty, which allows us to express any set expression b as a sum of those of the N integer variables $l_{p_1^i,\ldots,p_{10}^i}$ whose indices $p_1^i, \ldots, p_{10}^i$ identify Venn regions that belong to b.
2. The computation of a sufficient polynomial value for N is the second contribution of this paper. We start with the result [**?**] 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 [?] gives the membership of QFBAPA in NP, but, given the NP-hardness of satisfiability of the generated formulas, it is desirable to obtain as tight a bound on $N(d)$ as possible. We make the following steps towards the computation of a precise bound: 1) we compute the exact bound $N(d) = d$ for $d \leq 3$; 2) we identify a lower bound $N(d) \geq d + \lfloor \frac{d}{4} \rfloor$ for $d \geq 4$; 3) we provide several equivalent characterizations of vectors that achieve the optimal bound for any d; 4) 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 describe our implementation of the algorithm in the context of the Jahob verification system and present preliminary experiments on the examples of Figure 2 and their variations.

Our previously reported results. We suggested the possibility of the existence of sparse solutions in the final version of [13], where we also established the complexity of quantified BAPA. In a previous technical report [16] we identified a PSPACE algorithm for QFBAPA, but the techniques used there are different and not needed for the results of this paper. A preliminary version of the current result is described in [11, 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 linear 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 [22].) 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 perform the separation into $G \wedge F$ in practice. We obtain the following simple summary of our algorithm: substitute each expression $|b_i|$ 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 $N_0 + 1 \leq j \leq N$. This suggests an iterative algorithm of Figure 3 that starts with $N = 0$ and increases N until a counterexample is found or a provably sufficient bound is reached. 'break_symmetry' is a symmetry breaking predicate that imposes a lexicographical order on propositional variables β_j. 'set_expressions_into_card' transforms all boolean algebra expressions into form $|b_i| = 0$.

let findN(f : QFBAPA) : bool =
  let d = #atomic_formulas(f)
  let s_0 = #formulas_with_0_rhs(f)
  let s_1 = #formulas_with_1_rhs(f)
  let $d_1 = d - s_0 - s_1$
  let N_1 = if $(d_1 \leq 3)$ d_1
    else $\max\{n \mid 2^n \leq (n+1)^{d_1}\}$
  return $N_1 + s_1$

let makePA(f : QFBAPA, N : int) : QFPA =
  let $f_1 = f[|b_i| \mapsto \sum_{j=1}^{N} \texttt{ite}([\![b_i]\!]_{\beta_j}, l_j, 0)]_i$
  return
   $((\bigwedge_j l_j \geq 0) \wedge \text{break\_symmetry}) \rightarrow f_1$

let valid(f_0 : QFBAPA) : bool =
  let f = negation_normal_form(
    set_expressions_into_card(f_0))
  let N_0 = findN(f);
  $N := 0$;
  while($N \leq N_0$) do
    let f_{PA} = makePA(f, N)
    if $\neg$validPA(f_{PA}) return false;
    else $N := N + 1$;
  return true;

Fig. 3. Our algorithm for deciding QFBAPA formulas

3 Upper Bound on the Number of Non-Zero 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 $\sum_{y \in X} y$.

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

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

The following result is established as Theorem 1(ii) in [?].

Fact 1 (Eisenbrand, Shmonin (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 $|\tilde{X}| \leq 2d \log_2(4dM)$.*

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

$$x_\beta = (\llbracket b_0 \rrbracket_\beta, \llbracket b_1 \rrbracket_\beta, \ldots, \llbracket b_e \rrbracket_\beta).$$

Fact 1 implies is that if $(k_0, k_1, \ldots, k_p) \in \text{int\_cone}(X)$ where k_i are as in formula (2), then $(k_0, k_1, \ldots, k_p) \in \text{int\_cone}(\tilde{X})$ where $|\tilde{X}| = 2d \log_2(4d)$ (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 $N = 2d \log_2(4d)$ 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 Bounds and Nonredundant Integer Cone Generators

Definition 2. *Let $X \subseteq \mathbb{Z}^d$. We say that X is a* nonredundant integer cone generator *for b, and write NICG(X, b), if both 1) $b \in \text{int\_cone}(X)$, and 2) $b \notin \text{int\_cone}(X \setminus \{y\})$ for every $y \in X$.*

In the sequel we consider only vectors of *non-negative* integers, so $X \subseteq \mathbb{Z}^d_{\geq 0}$.

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 $\text{int\_cone}(X)$.

Lemma 1. *Suppose NICG(X, b) for $X \subseteq \mathbb{Z}^d_{\geq 0}$. If $\lambda_1, \lambda_2 : X \to \mathbb{Z}_{\geq 0}$ 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). Then $(0, \ldots, 0) \notin X$. Let $\lambda_1, \lambda_2 : X \to \mathbb{Z}_{\geq 0}$ 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 (otherwise the left-hand side would be zero vector and the right-hand side a vector with a strictly positive coordinate since $(0, \ldots, 0) \notin X$). By $b \in \text{int\_cone}(X)$, let $\lambda : X \to \mathbb{Z}_{\geq 0}$ such that $b = \sum_{x \in X} \lambda(x)x$. Let x_0 be such that $\lambda'_1(x_0) = 1$ and $\lambda(x_0) = \min\{\lambda(x) \mid \lambda'_1(x) = 1\}$. 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. Because all coefficients are non-negative, we conclude $b \in \text{int\_cone}(X \setminus \{x_0\})$, contradicting $\text{NICG}(X, b)$. ∎

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

Theorem 1. *Let $X \subseteq \mathbb{Z}^d_{\geq 0}$. 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) → 2): This is Lemma 1.

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

4) → 1): Take $b = \sum X$. ∎

Corollary 1 is used in [?] to establish the bound on the size of X with $\text{NICG}(X)$. 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 $\sum Y_1 \neq \sum Y_2$.*

Every set contains a NICG subset that generates a given element. To establish the existence of sparse solutions, it therefore suffices to establish bounds on the cardinality of X such that $\text{NICG}(X)$.

Lemma 2. *If $b \in \text{int\_cone}(X)$, then there exists $\tilde{X} \subseteq X$ with $b \in \text{int\_cone}(\tilde{X})$ and NICG$(\tilde{X}, b)$.*

Proof. If $b \in \text{int\_cone}(X)$ then by definition $b \in \text{int\_cone}(X_0)$ for a finite $X_0 \subseteq X$. If not $\text{NICG}(X_0, b)$, then $b \in \text{int\_cone}(X_1)$ where X_1 is a proper subset of X_0. Continuing in this fashion we obtain a finite maximal sequence $X_0 \supset X_1 \supset \ldots \supset X_k$ where $\text{NICG}(X_k, b)$, so we let $\tilde{X} = X_k$. ∎

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

Define

$$g(d) = \max\{n \mid 2^n \leq (n+1)^d\} \tag{6}$$

Theorem 2. *Let $X \subseteq \{0,1\}^d$, NICG(X), and $N = |X|$. Then for $d \geq 2$,*

$$N \leq g(d) \leq (1 + \varepsilon(d))(d \log_2 d) \tag{7}$$

where $\varepsilon(d) \leq 1$ and $\lim_{d\to\infty} \varepsilon(d) = 0$.

Proof. Let $X \subseteq \{0,1\}^d$, $\text{NICG}(X)$ and $N = |X|$. We prove $2^N \leq (N+1)^d$. Suppose that, on the contrary, $2^N > (N+1)^d$. If $\sum Y = (x^1, \ldots, x^d)$ for $Y \subseteq X$, then $0 \leq x^j \leq N$ because $Y \subseteq \{0,1\}^d$ and $|Y| \leq N$. 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, $2^N \leq (N+1)^d$. We next show that any n for which $2^n \leq (n+1)^d$ is bounded by $(1 + \varepsilon(d))(d \log_2 d)$. Using elementary reasoning, from $2^n \leq (n+1)$ we obtain $n \leq 2d \log_2(2n)$ (see [?], [11, Section 7.9.3] for details). Substituting this bound on n back into $n \leq d \log_2(n+1)$ we obtain

$$\begin{aligned} n &\leq d \log_2(n+1) \leq d \log_2(2d \log_2(2d) + 1) = d \log_2(2d(\log_2(2d) + \tfrac{1}{2d})) \\ &= d(1 + \log_2 d + \log_2(\log_2(2d) + \tfrac{1}{2d})) = d \log_2 d(1 + \tfrac{1+\log_2(\log_2(2d)+\frac{1}{2d})}{\log_2 d}) \end{aligned}$$

so we can let $\varepsilon(d) = (1 + \log_2(\log_2 d + 1 + \frac{1}{2d}))/\log_2 d$. ∎

Define $N(d) = \max\{|X| \mid X \subseteq \{0,1\}^d \wedge \text{NICG}(X)\}$. We have shown $N(d) \leq g(d)$. Thanks to the monotonicity of g, we can compute $g(d)$ efficiently using binary search.

4.1 Lower Bounds

Although we currently do not have tight bounds for $N(d)$, in this section we show several observations about lower bounds for $N(d)$.

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

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

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

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 $N(d+1) \ge N(d)+1$. ∎

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

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

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

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

Proof. By Corollary 2, if $N(d+1) = d+1$, then $N(d)+1 \le d+1$ so $N(d) \le d$. Therefore, $N(3) = 3$ 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 $|X_0| = d+1$ 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 $b \in$ int_cone(X). By previous argument on real-valued relaxation, $N_R(3) = 3$, so b is in convex cone of some three vectors from X, say $b \in$ cone($\{x_1, x_2, x_3\}$). On the other hand, $b \notin$ int_cone($\{x_1, x_2, x_3\}$). 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 the absolute value of the determinant of the matrix $[x_1, x_2; x_3]$ is greater 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). ∎

***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 $d \ge 4$.

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.)

$$\begin{aligned} \lambda_1 + \lambda_2 + \lambda_3 &= 3 \\ \lambda_2 + \lambda_3 + \lambda_4 &= 3 \\ \lambda_1 + \lambda_3 + \lambda_4 + \lambda_5 &= 4 \\ \lambda_1 + \lambda_2 + \lambda_4 + \lambda_5 &= 4 \end{aligned} \tag{8}$$

It is easy to see that the system has $(1,1,1,1,1)$ as *the only solution* in the space of non-negative integers. Note that all variables are non-zero in this solution. The five columns of the system (8) 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 [27], there are maximal NICG sets of different cardinality. Nevertheless, Lemma 2 and Lemma 3 show that some of the properties of independent sets do hold for vectors in X where NICG(X).

Note also that X_5 is not a Hilbert basis [25]. 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 $d \geq 4$.

4.2 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 are 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 assignment 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 $k_i < c_i$. Given the bounds c_i, we have the following inequality that generalizes the one in Theorem 2:

$$2^N \leq \prod_{i=1}^{d} (1 + \min(c_i, N)) \tag{9}$$

The reasoning follows the proof of Theorem 2.

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 (9) 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, since for them $c_i = 0$. The bound $c_i = 1$ also frequently occurs in our examples because we encode elements as singleton sets.

5 Preliminary experiments

Figure 4 shows formula sizes and running times for the original BAPA algorithm and our new QFBAPA algorithm (Figure 3) on formulas of Figure 2 and their variations.[3]

The examples $2a$, $3a$, $6a$ are more realistic versions of examples 2, 3, 6 because they contain some unnecessary assumptions that would normally appear in an automatically generated verification condition. Syntactically determining which assumptions are useful is a difficult problem [6], so it is reasonable to leave this task to the the decision procedure. Formulas $1-6$, $2a$, $3a$, $6a$ are all valid. Formulas $2b$, $3b$, $4b$, $5b$, $6b$, $6c$ are obtained from $2a$, $3a$, 4, 5, $6a$, $6a$, respectively, by dropping one of the necessary assumptions or changing the relation between integers to make the formula invalid. All invalid formulas are marked by $*$ in Figure 4. In addition to the running time, the figure shows the number of abstract syntax tree nodes in the generated quantifier-free Presburger arithmetic formulas. For the QFBAPA algorithm, the "iteration of N" column indicates the number of non-empty Venn regions for which a counterexample was found, in the case when the formula is invalid. For valid formulas this column indicates the bound that was computed as being sufficient to establish formula validity; this bound is actually explored whenever validity checking terminates in the given timeout. In any case, the size of the generated PA formula corresponds to this value of N. The running time for QFBAPA is the sum of running times over all iterations, corresponding to the overall running time of the algorithm. We ran the experiments on 3GHz, 1MB cache, 2GB RAM workstation. As a decision procedure for quantifier-free PA we used CVC3 version 20070217 [3].

[3] The examples are available from http://lara.epfl.ch/~kuncak/cade07examples

VC# (∗=invalid)	BAPA		QFBAPA		
	PA size(nodes)	total time(s)	PA size(nodes)	iteration of N	total time(s)
1	39	< 0.1	190	3	< 0.1
2	57	< 0.1	220	4	0.1
$2a$	1049	1.8	840	5	15.4
$*2b$	946	1.4	87	1	< 0.1
3	51	< 0.1	131	3	< 0.1
$3a$	532	0.4	688	5	7.3
$*3b$	532	0.4	92	1	< 0.1
4	546	0.5	1328	8	> 100.0
$*4b$	554	0.5	284	2	0.1
5	2386	13.6	1750	8	> 100.0
$*5b$	2318	13.4	570	3	0.4
6	442	0.4	2613	18	> 100.0
$6a$	10822	> 100.0	8401	23	> 100.0
$*6b$	10822	> 100.0	1021	3	0.8
$*6c$	10563	> 100.0	990	3	0.9

Fig. 4. Results for variations of formulas in Figure 2

Discussion. These results suggest that our new algorithm is more effective than the previous algorithm for finding counterexamples of invalid formulas. On large valid formulas our algorithm generates more compact quantifier-free PA formulas than the introduction of exponentially many variables, but the complexity of generated formulas makes them difficult to solve, leading to worse overall performance. Nevertheless, on small formulas our new algorithm terminates, reaching the computed upper bound on N and thus establishing formula validity.

6 Related Work

We have presented the the first 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 representation 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 [?]. 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. Previous results, such as [25], consider matroids and Hilbert bases. As we remark in Section 4.1, the sets of vectors X with NICG(X) do not form a matroid, and maximal NICG(X) need not be a Hilbert basis. The equations generated from QFBAPA problems are more difficult than set packing and set partitioning problems [2], because our partition cardinality variables are not restricted to $\{0, 1\}$.

Relationship to counting SAT. Although similarly looking, QFBAPA satisfiability is different from the #SAT problem [26]. Solving QFBAPA formula differs from counting the number of satisfying assignments of propositional formulas because set partitions may possibly be empty. An immediate consequence of our results is that there is no QFBAPA formula of size polynomial in n that would express the property "all 2^n partitions of n sets are non-empty".

Reasoning about sets. The quantifier-free fragment of BA is shown NP-complete in [17]; see [14] for a generalization of this result using the parameterized complexity of the Bernays-Schönfinkel-Ramsey class of first-order logic [5, Page 258]. The decision procedure for quantifier-free fragment with cardinalities in [7, 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 [21]. 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 [21] in 0.1 seconds whereas SPASS+T is reported to time out in [21]).

Acknowledgements. Alexandr Andoni suggested using binary search instead of an analytical expression to compute the inverse of $N/\log(N+1)$. Stefan Andrei and Bruno Marnette made remarks about result [16]. Emina Torlak used her Kodkod constraint solver to search for counterexamples of the conjecture $N(3) = 3$ (Section 4.1 shows that this conjecture is true). We thank Zoran Džunić, Michael Sipser, and the anonymous reviewers for useful feedback.

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 *CAV*, 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. Egon Börger, Erich Grädel, and Yuri Gurevich. *The Classical Decision Problem*. Springer-Verlag, 1997.
6. 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.
7. Domenico Cantone, Eugenio Omodeo, and Alberto Policriti. *Set Theory for Computing*. Springer, 2001.
8. S. Feferman and R. L. Vaught. The first order properties of products of algebraic systems. *Fundamenta Mathematicae*, 47:57–103, 1959.

9. Silvio Ghilardi. Model theoretic methods in combined constraint satisfiability. *Journal of Automated Reasoning*, 33(3-4):221–249, 2005.
10. Dexter Kozen. *Theory of Computation*. Springer, 2006.
11. Viktor Kuncak. *Modular Data Structure Verification*. PhD thesis, EECS Department, Massachusetts Institute of Technology, February 2007.
12. 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.
13. 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.
14. 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.
15. Iddo Lev. Precise understanding of natural language. Stanford Univeristy PhD dissertation draft, February 2007.
16. Bruno Marnette, Viktor Kuncak, and Martin Rinard. On algorithms and complexity for sets with cardinality constraints. Technical report, MIT CSAIL, August 2005.
17. Kim Marriott and Martin Odersky. Negative boolean constraints. Technical Report 94/203, Monash University, August 1994.
18. 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. Kluwer, 1998.
19. Christos H. Papadimitriou. On the complexity of integer programming. *J. ACM*, 28(4):765–768, 1981.
20. Ian Pratt-Hartmann. Complexity of the two-variable fragment with counting quantifiers. *Journal of Logic, Language and Information*, 14(3):369–395, 2005.
21. Virgile Prevosto and Uwe Waldmann. SPASS+T. In *ESCoR: Empirically Successful Computerized Reasoning*, volume 192, 2006.
22. 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`.
23. 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.
24. Peter Z. Revesz. The expressivity of constraint query languages with boolean algebra linear cardinality constraints. In *ADBIS*, pages 167–182, 2005.
25. 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.
26. S. Toda. PP is as hard as the polynomial-time hierarchy. *SIAM Journal on Computing*, 20(5):865–877, 1991.
27. H. Whitney. On the abstract properties of linear independence. *American Journal of Mathematics*, 57:509–533, 1935.
28. Calogero G. Zarba. Combining sets with cardinals. *J. of Automated Reasoning*, 34(1), 2005.