Negative Boolean Constraints*

Kim Marriott
Department of Computer Science
Monash University
Clayton, Vic 3168 AUSTRALIA.

Martin Odersky
Institut für Programmstrukturen
und Datenorganisation,
Universität Karlsruhe
76128 Karlsruhe, GERMANY.

Abstract

Systems of Boolean constraints which allow negative constraints such as $f \not\subseteq g$ are investigated. The results form a basis for algorithms to determine satisfiability, validity, implication, equivalence and variable elimination for such systems. These algorithms have applications in spatial query decomposition, machine reasoning, and constraint logic programming. Proofs of the results rely on independence of inequations, which enables results for systems with a single inequation to be lifted to systems with many inequations.

1 Introduction

Since Boole [2], systems (or conjunctions) of positive constraints $f \subseteq g$ over a Boolean algebra have been extensively studied. Here, we introduce and study a more general notion of Boolean constraint system in which *negative* Boolean constraints $f \not\subseteq g$ are also allowed. Systems of positive and negative constraints have not yet been widely studied in their own right. This may be because in the case of two-valued Boolean algebras, negative constraints add no power since the constraint $x \not\subseteq y$ is equivalent to $x = 1 \wedge y = \emptyset$. For more general Boolean algebras, however, systems of general Boolean constraints are strictly more powerful than systems of positive constraints; for instance, they allow inequality and strict containment to be expressed.

Our main technical results are in two areas. The first is determining satisfiability. The problem whether a Boolean equation is satisfiable is well known to be NP-complete. We show that deciding satisfiability of propositional formula over Boolean equations is also NP-complete. This implies as special cases NP-completeness of testing satisfiability for general Boolean constraints and co-NP-completeness of testing validity, implication and equivalence. We also show that the height of the Boolean algebra exactly characterizes the propositional formula which are satisfiable in it.

The second area is variable elimination. Systems of positive Boolean constraints S are closed under existential quantification, that is, $\exists x.S$ can always be expressed as a system of positive Boolean constraints. Thus, variable x can be eliminated from S. This ceases to be true if negative constraints are added. However, we show that general systems of constraints are closed under existential quantification for a class of reasonable Boolean algebras, namely the *atomless* algebras. Further we give a simple formula to compute the equivalent unquantified system.

Positive Boolean constraints have many applications in computer science. Negative constraints over general Boolean algebras also arise naturally in several areas, in particular in applications involving sets.

One such area is spatial query languages with application to geographic information systems, CAD systems, VLSI design rule checkers, or to visual language parsing. Here, general Boolean constraints allow us to express overlap and strict containment queries on regions in addition to the non-strict containment queries which are expressed by just positive constraints. Using the results given here, arbitrary multivariate spatial queries can be decomposed into sequences of univariate queries. Previously, spatial query languages were restricted to queries with acyclic variable dependencies in order to make query decomposition feasible [20]. This has been investigated in more detail in [9, 10].

*Part of this work was done while both authors were at I.B.M. Thomas J. Watson Research Center

Another application is in machine reasoning as simple Boolean inequations suffice to complete all possible syllogistic moods, and thus complete Aristotelian logic (see Chapter 10 of [17]).

A final application is in programming and database query languages. Recently there has been interest in constraint logic programming languages [11] which extend logic programming languages and in constraint query languages [12] which extend relational database query languages by allowing different constraint domains. In particular, systems such as CHIP [18] and Prolog-III [6] are extensions of Prolog which provide positive Boolean constraints. The results given here allow such languages to be further extended to handle negative Boolean constraints without increasing the worst-case complexity of the constraint solving algorithm.

The rest of this note is organized as follows: In Section 2, properties of positive Boolean constraints are reviewed. Section 3 investigates systems with a single inequation. Section 4 investigates independence of negative constraints. Sections 5 and 6 use these independence results to lift results of Section 3 to systems with more than one inequation. Section 7 discusses related work.

2 Preliminaries: Boolean Algebras and Positive Boolean Constraints

Boolean algebras and positive Boolean constraints were first introduced by Boole [2] in an effort to automate reasoning. Since that time they have been extensively studied, and have proved fundamental in numerous application areas. In this section we introduce our terminology and review properties of positive Boolean constraints that we shall make use of in the sequel. We assume that the reader has an elementary knowledge of Boolean algebras and Boolean equations. Suitable references are [3] and [13].

A *Boolean formula* is a variable, a constant $\emptyset$ or 1, the complement of a formula, a disjunction of formulas, or a conjunction of formulas. A formula is *atomic* if it is a variable or a constant. A *literal* is an atomic formula or its complement. A *term* is a conjunction of literals. A *Boolean function* is a function that can be described by a Boolean formula. A *positive* Boolean constraint is of the form $f \subseteq g$ where f and g are Boolean formulas.

Boole showed that any system of positive Boolean constraints can be rewritten to an equivalent Boolean equation of the form $f = \emptyset$ where f is a Boolean formula. Boole's "fundamental theorem of Boolean algebra" allows us to rewrite a Boolean formula f into a form in which any given variable x in f is isolated. Letting $f_x(a)$ denote the formula obtained by replacing all occurrences of x in f by a, we have that:

Theorem 2.1 (Boole) $f = x \cdot f_x(1) + \overline{x} \cdot f_x(\emptyset)$.

Applying Theorem 2.1 to all variables in a Boolean formula f yields f's *(extended) disjunctive normal form, dnf(f)*. Note that each term in the extended disjunctive normal form contains all variables in the system.

Theorem 2.2 (Boole) Let $x^{-1} = \overline{x}$ and $x^1 = x$. For every Boolean formula f in variables $x_1, ..., x_n$:

$$f = \sum_{(a_1,...,a_n)\in\{-1,1\}^n} f(a_1, ..., a_n) \cdot x_1^{a_1} \cdot ... \cdot x_n^{a_n}.$$

It follows that positive constraints are closed under existential quantification:

Theorem 2.3 $\exists x \,.\, f = \emptyset \Leftrightarrow f_x(\emptyset) \cdot f_x(1) = \emptyset$.

Using Schröder's theorem we can rewrite an equality $f_x = \emptyset$ into an equivalent range constraint over variable x:

Theorem 2.4 (Schröder) $f = \emptyset \Leftrightarrow f_x(\emptyset) \subseteq x \subseteq \overline{f_x(1)}$.

One important example of a Boolean algebra is the power set $\wp X$ of any set X, where set union, intersection and complement are the disjunction, conjunction and complement operators respectively. Another example are the propositional formula.

A *field of sets* is a subset of a power set that is closed under complements and finite unions and intersections. Fields of sets are important to the study of Boolean algebras because of Stone's Representation Theorem:

Theorem 2.5 (Stone) Every Boolean algebra is isomorphic to a field of sets.

A useful corollary is that every finite Boolean algebra is isomorphic to a finite power set. Another useful consequence (see [13], Proposition 2.19) is that:

Proposition 2.6 A system of positive Boolean constraints is satisfiable in some Boolean algebra iff it is satisfiable in all Boolean algebras.

Definition. The *height* of an element x of a Boolean algebra, denoted by $h(x)$, is the least upper bound of the lengths of all chains between $\emptyset$ and x. The *height* of a Boolean algebra is the height of the top element 1 in this algebra. A Boolean algebra is *infinite* if it has infinite height.
For instance, the height of $x \in \wp X$ is the cardinality of x.

Definition. A non-empty element x of a Boolean algebra M is *atomic* iff there exists no element y in M such that $\emptyset \subset y \subset x$. M is *atomless* iff it contains no atomic elements.
An example of an atomless Boolean algebra is the set of (equivalence classes of) measurable subsets of $\Re^k$, in which two sets are considered equivalent when they are identical "almost everywhere". This Boolean algebra corresponds to the data model in spatial databases in which regions are not arranged on a grid.

Here we investigate an extension of Boolean constraints in which negative constraints are allowed. A *negative* Boolean constraint is of the form $f \not\subseteq g$ where f and g are Boolean formulas. Systems with negative and positive Boolean constraints not only provide containment, equality and non-containment, but also provide inequality and strict containment, as

$$\begin{array}{lcl} x \neq y & \Leftrightarrow & x \cdot \overline{y} + \overline{x} \cdot y \not\subseteq \emptyset, \\ x \subset y & \Leftrightarrow & x \subseteq y \wedge y \not\subseteq x. \end{array}$$

3 Systems with a Single Inequation

We have seen that any system of positive Boolean constraints can be rewritten to an equivalent Boolean equation. Thus, any system of Boolean constraints is equivalent to a system of the form:

$$f = \emptyset \wedge g_1 \neq \emptyset \wedge \ldots \wedge g_n \neq \emptyset$$

where f and the g_i's are Boolean formulas.

Before studying the general case, we will look at the "simple" case when the system has a single inequation. We shall see that they behave very much like positive systems.

Definition. A system of Boolean constraints is *simple* if it has the form $f = \emptyset \wedge g \neq \emptyset$.

Simple systems have a straightforward test for satisfiability. In a rewording of Proposition 10.1 in Rudeanu [17] we have:

Proposition 3.1 Let S be the simple system $f = \emptyset \wedge g \neq \emptyset$. S is satisfiable iff $g \not\subseteq f$. ■

As proven in [9], simple systems admit quantifier elimination:

Lemma 3.2 (Double Diamond) For arbitrary elements a, b, c, d:

$$\exists x \,.\, a \subseteq x \subseteq b \wedge \neg (c \subseteq x \subseteq d) \Leftrightarrow a \subseteq b \wedge \neg (b \subseteq d \wedge c \subseteq a).$$

Proof: First consider the "$\Rightarrow$" direction. Clearly, $a \subseteq b$ follows from the antecedent. Assume that the second part of the consequent does not hold, then $b \subseteq d \wedge c \subseteq a$. Together with $a \subseteq b$ this implies $c \subseteq a \subseteq b \subseteq d$, which contradicts the antecedent.
Now consider the "$\Leftarrow$" direction. If we assume $a \subseteq b \wedge b \not\subseteq d$, the consequent holds with $x = b$. On the other hand, assuming $a \subseteq b \wedge c \not\subseteq a$, the consequent holds with $x = a$. ■

Proposition 3.3 Let S be the simple system $f = \emptyset \wedge g \neq \emptyset$. Then

$$\exists x \,.\, S \Leftrightarrow f_x(\emptyset) \cdot f_x(1) = \emptyset \;\wedge\; \overline{f_x(1)} \cdot g_x(1) + \overline{f_x(\emptyset)} \cdot g_x(\emptyset) \neq \emptyset.$$

Proof: Let A be $f_x(\emptyset)$, B be $f_x(1)$, C be $g_x(\emptyset)$ and D be $g_x(1)$.

$$\begin{array}{rcll} \exists x \,.\, S & \Leftrightarrow & \exists x \,.\, A \subseteq x \subseteq \overline{B} \wedge \neg (C \subseteq x \subseteq \overline{D}) & \text{(from Theorem 2.4)} \\ & \Leftrightarrow & A \subseteq \overline{B} \wedge \neg (\overline{B} \subseteq \overline{D} \wedge C \subseteq A) & \text{(from Lemma 3.2)} \\ & \Leftrightarrow & A \subseteq \overline{B} \wedge (\overline{B} \not\subseteq \overline{D} \vee C \not\subseteq A) & \\ & \Leftrightarrow & A \cdot B = \emptyset \wedge (\overline{B} \cdot D \neq \emptyset \vee C \cdot \overline{A} \neq \emptyset) & \\ & \Leftrightarrow & A \cdot B = \emptyset \wedge \overline{B} \cdot D + C \cdot \overline{A} \neq \emptyset. \quad \blacksquare & \end{array}$$

In the sequel we will extend these results to the general case. We do this by finding sufficient conditions for "independence" of inequations to hold. We distinguish two types of independence.

Definition. *Weak independence* (of inequations) holds for a Boolean algebra M iff for any constraint system S, of the form $f = \emptyset \wedge g_1 \neq \emptyset \wedge \ldots \wedge g_n \neq \emptyset$ say,

$$M \models \exists \,.\, S \Leftrightarrow \exists \,.\, (f = \emptyset \wedge g_1 \neq \emptyset) \wedge \ldots \wedge \exists \,.\, (f = \emptyset \wedge g_n \neq \emptyset)$$

where $\exists \,.\, S$ denotes the existential closure of system S.
Strong independence (of inequations) holds for M iff for any variable x and constraint system S, of the form $f = \emptyset \wedge g_1 \neq \emptyset \wedge \ldots \wedge g_n \neq \emptyset$ say,

$$M \models \exists x \,.\, S \Leftrightarrow \exists x \,.\, (f = \emptyset \wedge g_1 \neq \emptyset) \wedge \ldots \wedge \exists x \,.\, (f = \emptyset \wedge g_n \neq \emptyset).$$

Clearly, strong independence implies weak. Weak independence allows satisfiability tests for the simple case to be lifted to the general case, while strong independence allows quantifier elimination techniques to be lifted. We note that if arbitrary constant symbols are allowed, then strong and weak independence are equivalent.

Unfortunately, neither strong nor weak independence holds for all Boolean algebras. In the next section we show that strong independence holds for exactly the atomless Boolean algebras and that weak independence holds for exactly the Boolean algebras of infinite height.

4 Independence

In this section we characterize when Boolean algebras are strongly or weakly independent. We first consider weak independence.

The next lemma is a key technical result of the paper. Given the disjunctive normal form of a formula, it allows us to construct a Boolean algebra M such that there is an assignment for M which satisfies exactly the terms in the disjunctive normal form. What is technically difficult is to ensure that the height of M is bounded by the number of terms in the disjunctive normal form.

Lemma 4.1 Let T be the set of terms constructed from the variables $x_1, \ldots, x_n$, $n \geq 1$, and T^+ a non-empty subset of T. Let M be the powerset of height $|T^+|$. Then there is an assignment σ from $x_1, \ldots, x_n$ to M such that $\sigma\ t \neq \emptyset \Leftrightarrow t \in T^+$.

Proof: The proof is by induction on the number of variables n. A simple case analysis shows that the hypothesis holds when $n = 1$.

We now prove it for $n > 1$. Let S be the set of terms constructed from $x_1, \ldots, x_{n-1}$. Then $T = \{x_n \cdot t, \overline{x_n} \cdot t \mid t \in S\}$. Let

$$\begin{array}{rcl} S^+ & = & \{t \in S \mid x_n \cdot t \in T^+ \vee \overline{x_n} \cdot t \in T^+\}, \\ S^* & = & \{t \in S \mid x_n \cdot t \in T^+ \wedge \overline{x_n} \cdot t \in T^+\}. \end{array}$$

Then $|T^+| = |S^+| + |S^*|$. Let M' be the powerset of height $|S^+|$. By the induction hypothesis, there is an assignment σ' from $x_1, \ldots, x_{n-1}$ to M' such that $\forall t \in S. \sigma'\ t \neq \emptyset \Leftrightarrow t \in S^+$. We now embed M' in M, the powerset of height $|T^+|$, by adding $|S^*|$ extra atoms. For each $t \in S^*$ pick an atom $a_t \subseteq \sigma'\ t$. Add extra atoms a'_t, one for each a_t, to M' to give M. Define

$$\sigma\ x \quad = \quad \sigma'\ x + \sum \{a'_t \mid a_t \subseteq \sigma'\ x\}.$$

Then, for all $t \in S$, $\sigma\ t \neq \emptyset \Leftrightarrow \sigma'\ t \neq \emptyset$, and, for all $t \in S^*$, $\sigma\ t$ is not an atom. Let X be the following element of M.

$$X = \sum \left\{ \begin{array}{ll} a_t & \text{if } x \cdot t \in T^+ \text{ and } \overline{x} \cdot t \in T^+ \\ \sigma\ t & \text{if } x \cdot t \in T^+ \text{ and } \overline{x} \cdot t \notin T^+ \\ \overline{\sigma\ t} & \text{if } x \cdot t \notin T^+ \text{ and } \overline{x} \cdot t \in T^+ \end{array} \middle| \ t \in S^+ \right\}.$$

Extend σ to $x_1, ..., x_n$ by defining $\sigma\ x_n = X$. It is straightforward to verify that for all $t \in T$, $\sigma\ t \neq \emptyset \Leftrightarrow t \in T^+$, which proves the inductive step. ■

The next lemma allows us to lift this result to Boolean algebras of greater height.

Lemma 4.2 Let σ' be an assignment from variables $x_1, ..., x_n$ to a Boolean algebra M' of finite height d. Then, for any (possibly infinite) Boolean algebra M with $h(M) \geq h(M')$ there is an assignment σ from $x_1, ..., x_n$ to M such that $\sigma\ t = \emptyset \Leftrightarrow \sigma'\ t = \emptyset$.

Proof: By Stone's representation theorem, we can choose M' to be the powerset $\wp\{a_1, ..., a_d\}$. Since $h(M) \geq h(M')$, we can partition M into d pairwise disjoint elements s_i, one for each atom $\{a_i\}$ in M'. Define

$$\sigma\ x = \sum \{s_i \mid a_i \in \sigma'\ x\}$$

It is straightforward to verify $\sigma\ t = \emptyset$ in M iff $\sigma'\ t = \emptyset$ in M'. ■

Proposition 4.3 Given Boolean formula f, $g_1, ..., g_n$, in any Boolean algebra M with $h(M) \geq n$,

$$\exists\, .\, f = \emptyset \wedge g_1 \neq \emptyset \wedge ... \wedge g_n \neq \emptyset \quad \Leftrightarrow \quad \exists\, .\, (f = \emptyset \wedge g_1 \neq \emptyset) \wedge ... \wedge \exists\, .\, (f = \emptyset \wedge g_n \neq \emptyset).$$

Proof: Direction "$\Rightarrow$" is trivial. We now show "$\Leftarrow$". Let $\sum T'$ be the disjunctive normal form of f and $\sum T_i$ the disjunctive normal form of g_i. Since each system $f = \emptyset \wedge g_i \neq \emptyset$ is satisfiable, we have that for each i, there is a term $t_i \in T_i \setminus T'$. From Lemma 4.1 there exists an assignment σ to the powerset of height n such that $\sigma\ t \neq \emptyset \Leftrightarrow \exists t_i\, .\, t = t_i$. Thus, σ is a solution of the original system in this powerset. The result follows then from Lemma 4.2. ■

We can now exactly characterize those Boolean algebras which are weakly independent.

Theorem 4.4 (Weak Independence) A Boolean algebra is weakly independent iff it is infinite.

Proof: The "$\Leftarrow$" direction follows immediately from Proposition 4.3. Now consider the other direction. The proof is by contradiction. Assume that M is a finite Boolean algebra of height n. Let S be the system corresponding to the constraints $\emptyset \subset x_1 \subset x_2 \subset ... \subset x_n \subset 1$. It is straightforward to verify that weak independence does not hold for S. ■

Next we develop a characterization of those Boolean algebras which are strongly independent. Using a construction similar to that used in the proof of Lemma 4.1 we can show that:

Proposition 4.5 In any atomless Boolean algebra:

$$\exists x\, .\, f = \emptyset \wedge g_1 \neq \emptyset \wedge ... \wedge g_n \neq \emptyset \quad \Leftrightarrow \quad \exists x\, .\, (f = \emptyset \wedge g_1 \neq \emptyset) \wedge ... \wedge \exists x\, .\, (f = \emptyset \wedge g_n \neq \emptyset).$$

Proof: Direction "$\Rightarrow$" is trivial. The proof of "$\Leftarrow$" is similar to that of Lemma 4.1. Let the free variables in the system be $x_1, ..., x_m$. Let S be the set of terms constructed from $x_1, ..., x_m$ and $T = \{x \cdot t, \overline{x} \cdot t \mid t \in S\}$.

Consider some assignment σ' to $x_1, ..., x_m$ such that for each g_i, there is an X_i such that the assignment $\sigma_i = \sigma'[x \mapsto X_i]$ is a solution of $f = \emptyset \wedge g_i \neq \emptyset$. Let $T^+ = \{t \in T \mid \exists \sigma_i\, .\, \sigma_i\ t \neq \emptyset\}$ and let $S^+ = \{t \in S \mid x \cdot t \in T^+ \vee \overline{x} \cdot t \in T^+\}$. Note that for each $t \in S^+$, $\sigma'\ t$ is non-empty and not an atom. Hence, there is for each $t \in S^+$ an element $s_t \in M$ such that $\emptyset \subset s_t \subset \sigma'\ t$. Define

$$X = \sum \left\{ \begin{array}{ll} s_t & \text{if } x \cdot t \in T^+ \text{ and } \overline{x} \cdot t \in T^+ \\ \sigma'\ t & \text{if } x \cdot t \in T^+ \text{ and } \overline{x} \cdot t \notin T^+ \\ \overline{\sigma'\ t} & \text{if } x \cdot t \notin T^+ \text{ and } \overline{x} \cdot t \in T^+ \end{array} \middle| \ t \in S^+ \right\}$$

and let $\sigma = \sigma'[x \mapsto X]$. It is straightforward to show that

$$\forall t \in T\, .\, \sigma\ t \neq \emptyset \Leftrightarrow \exists \sigma_i\, .\, \sigma_i\ t \neq \emptyset.$$

It follows that σ is a solution of $f = \emptyset \wedge g_1 \neq \emptyset \wedge ... \wedge g_n \neq \emptyset$. ■

Theorem 4.6 (Strong Independence) A Boolean algebra is strongly independent iff it is atomless.

Proof: The "$\Leftarrow$" direction follows immediately from Proposition 4.5. Now consider the other direction. The proof is by contradiction. Assume that M is a Boolean algebra with atom a. Let S be $\exists y \,.\, x \cdot y \neq \emptyset \wedge x \cdot \overline{y} \neq \emptyset$. It is straightforward to verify that strong independence does not hold for S when x is assigned a. ∎

In the sequel we shall see that these results can be used as the basis for algorithms for satisfiability testing and variable elimination.

5 Satisfiability

In this section we are concerned with determining satisfiability of Boolean constraint systems and propositions over these systems. There are really a number of different questions depending on whether we are interested in satisfiability in all Boolean algebras, in some Boolean algebra or in a particular Boolean algebra. It follows from Proposition 2.6 that for positive systems these three questions are equivalent. However, this is not true in the general case.

We will lift our discussion to discuss satisfiability of propositional formula constructed from Boolean constraints. For instance, if S and S' are general systems of Boolean constraints, then example propositional formula are S, $S \Leftrightarrow S'$ and $S \Rightarrow S'$. We are interested in these formula because deciding their satisfiability not only gives us a means for deciding satisfiability of a Boolean constraint system but also for determining equivalence and implication between Boolean constraint systems.

Definition. A *Boolean formula proposition (Bf-proposition)* is a positive Boolean constraint, the complement of a Bf-proposition, or a disjunction or conjunction of Bf-propositions.

The set of Bf-propositions is clearly a Boolean algebra. Terms in this algebra are just systems of Boolean constraints. Thus, every Bf-proposition is equivalent to a disjunction of systems of Boolean constraints. Satisfiability of a Bf-proposition can therefore be tested by first computing the disjunctive normal form of the Bf-proposition, and then testing if any system of Boolean constraints in the disjunctive normal form is satisfiable. We will be concerned with the following problems:

S1: Satisfiability in all Boolean algebras: Is a given Bf-proposition satisfiable in all Boolean algebras.

S2: Satisfiability in some Boolean algebra: Is a given Bf-proposition satisfiable in some Boolean algebra.

S3: Satisfiability in a particular Boolean algebra Given a Bf-proposition P and a height d, is S satisfiable in some/all Boolean algebras of height d.

We first consider problem S3: Satisfiability in a Boolean algebra of height d. In the case of finite d, an (inefficient) way to determine satisfiability is to just consider all assignments in the power set with d atoms. In the case of infinite d, the following theorem provides the basis for a satisfiability test.

Theorem 5.1 Let S be a system of the form $f = \emptyset \wedge g_1 \neq \emptyset \wedge \ldots \wedge g_n \neq \emptyset$, $n \geq 1$. Then, for any Boolean algebra M with $h(M) \geq n$, S is satisfiable in M iff for all g_i, $g_i \not\subseteq f$.

Proof: A simple consequence of Proposition 3.1 and Proposition 4.3.

Corollary 5.2 Let S a system of the form $f = \emptyset \wedge g_1 \neq \emptyset \wedge \ldots \wedge g_n \neq \emptyset$, $n \geq 1$, and let M be an infinite Boolean algebra. S is satisfiable in M iff for all g_i, $g_i \not\subseteq f$.

We now show that problems S1 and S2 reduce to problem S3. We first show that Boolean algebras of the same height are "equivalent" with respect to satisfiability. As all finite Boolean algebras of the same height are isomorphic it is immediate that:

Lemma 5.3 A Bf-proposition is satisfiable in some Boolean algebra of finite height d iff it is satisfiable in all Boolean algebras of height d.

Lemma 5.4 If a Bf-proposition is satisfiable in some Boolean algebra of finite height d, it is satisfiable in a Boolean algebras of height $d + d'$ (where d' need not be finite).

Proof: Analogous to the proof of Lemma 4.2

Proof: Let M be the Boolean algebra $\wp\{a_1, ..., a_d\}$ and let M' be the Boolean algebra $\wp\{a_1, ..., a_d, a_{d+1}, ..., a_{d+d'}\}$. Define the function $f : M \rightarrow M'$ by

$$f(x) = \begin{cases} x \cup \{a_{d+1}, ..., a_{d+d'}\} & \text{if } a_d \in x \\ x & \text{otherwise} \end{cases}$$

It is straightforward to verify that f is a homomorphism. Thus, if σ is a solution to Bf-proposition P for M, $f \circ \sigma$ is a solution to P for M'. ■

Theorem 5.5
(a) A Bf-proposition is satisfiable in some Boolean algebra of infinite height iff it is satisfiable in all Boolean algebras of infinite height.
(b) A Bf-proposition is satisfiable in a particular Boolean algebra iff it is satisfiable in all Boolean algebras of that or greater height.

Proof: Consider (a). This follows because a Bf-proposition is satisfiable in a Boolean algebra iff some term in its disjunctive normal form is satisfiable. A consequence of Theorem 5.1 is that a system of Boolean constraints is satisfiable in some infinite Boolean algebra iff it is satisfiable in all infinite Boolean algebras. Thus (a) holds. (b) follows from Lemma 5.3, Lemma 5.4, and (a). ■

Thus the height of a Boolean algebra exactly characterizes those Bf-propositions which are satisfiable in it. It is interesting to compare this to Tarski's characterization of elementarily equivalent Boolean algebras in terms of elementary invariants [13]. Other consequences of the theorem are:

Corollary 5.6 Let P be a Bf-proposition. Then P is satisfiable in all Boolean algebras iff P is satisfiable in the two element Boolean algebra.

Corollary 5.7 Let P be a Bf-proposition. Then P is satisfiable in some Boolean algebra iff P is satisfiable in some/all infinite Boolean algebras.

We now investigate the complexity of the above satisfiability problems.

Theorem 5.8 Problems S1 and S2 are NP-complete. Problem S3 is strongly NP-complete.

Proof: Since determining satisfiability of a single Boolean formula is NP-hard [7], it follows from Proposition 2.6 that each of these problems is NP-hard. That S3 is strongly NP-hard follows because the problem in which d is simply the constant 2 is still NP-hard. Proving that S1, S2 and S3 are in NP is more difficult. We look at each in turn.

Consider S1. As satisfiability in all Boolean algebras is equivalent to satisfiability in the two element Boolean algebra, S1 can be determined by non-deterministically guessing an assignment of 0 and 1 to the variables and checking if it is a solution.

Now consider S2. Let P be a Bf-proposition over m different positive Boolean constraints. Let M_m be the power set with m atoms. Let P have disjunctive normal form $S_1 \vee ... \vee S_n$. Each S_i is a conjunction of at most m Boolean constraints. It therefore follows from Theorem 5.1 that each S_i is satisfiable in some Boolean algebra iff it is satisfiable in M_m. Thus, P is satisfiable in some Boolean algebra iff it is satisfiable in M_m. Clearly each element in M_m can be represented by a bit-vector of length m. Thus, satisfiability in M_m can be determined by non-deterministically guessing an assignment of bit-vectors to the variables in P and checking if it is a solution.

Finally consider S3. Using a similar argument to that for S2, a Bf-proposition P is satisfiable in a Boolean algebra of height d iff it is satisfiable in the power set with $min\ \{m, d\}$ atoms, where m is the number of positive Boolean constraints in P. The argument for inclusion in NP proceeds as before. Note that we could not argue that S3 was in NP by just considering the powerset with d atoms. This is because the length of the bit vectors required to represent elements in this power set is d, but any reasonable representation of d in the problem instance has logarithmic length. ■

Consequences of this are that testing for satisfiability of systems of Boolean constraints is NP-complete, and that testing for validity, implication and equivalence of such systems is co-NP-complete.

6 Variable Elimination

We now turn to the problem of variable elimination in systems of Boolean constraints. That is, given a system S of Boolean constraints and a variable x, we wish to find an unquantified system which is equivalent to $\exists x.S$. Boole, Theorem 2.3, showed that positive constraints are closed under existential quantification. Unfortunately, arbitrary systems of Boolean constraints are *not* closed under existential quantification. To see this, consider the following counter-example:

Example 6.1 Consider the system S, $x \cdot y \neq \emptyset \wedge \overline{x} \cdot y \neq \emptyset$. Then $\exists x \,.\, S$ implies that $|y| \geq 2$, but there is no system of Boolean constraints over y which can capture exactly this. ■

However, as we have seen, simple systems are, unlike general systems, closed under existential quantification (Proposition 3.3). Further, strong independence of negative constraints holds in atomless Boolean algebras (Proposition 4.5). Thus, in atomless Boolean algebras, systems of Boolean constraints are closed under existential quantification.

Definition. Let S be the system $f = \emptyset \wedge g_1 \neq \emptyset \wedge \ldots \wedge g_n \neq \emptyset$. Define $proj(S, x)$ to be

$$A \cdot B = \emptyset \;\wedge\; \overline{B} \cdot D_1 + \overline{A} \cdot C_1 \neq \emptyset \;\wedge\; \ldots \;\wedge\; \overline{B} \cdot D_n + \overline{A} \cdot C_n \neq \emptyset$$

where A is $f_x(\emptyset)$, B is $f_x(1)$, C_i is $(g_i)_x(\emptyset)$ and D_i is $(g_i)_x(1)$.

Theorem 6.1 Let S be a system of Boolean constraints. In any atomless Boolean algebra,

$$\exists x \,.\, S \Leftrightarrow proj(S, x).$$

Proof: A simple consequence of Proposition 3.3 and Proposition 4.5.

Corollary 6.2 Atomless Boolean algebras admit quantifier elimination.

Proof: Let P be a Bf-proposition. Then from Theorem 6.1,

$$\exists x \,.\, P \Leftrightarrow \bigvee_{S \in dnf(P)} proj(S, x) \qquad \text{and} \qquad \forall x \,.\, P \Leftrightarrow \bigwedge_{S \in dnf(\neg P)} \neg\, proj(S, x). \quad ■$$

For Boolean algebras with atomic elements, $\exists x \,.\, S \Leftrightarrow proj(S, x)$ need not hold, as shown by Example 6.1. However the system $proj(S, x)$ still gives us information about $\exists x \,.\, S$. We can show that $proj(S, x)$ is the strongest (unquantified) system which is implied by $\exists x \,.\, S$ in all Boolean algebras. Thus it can be used as a "filter" when computing solutions of S, as any solution of S can be obtained by extending a solution of $proj(S, x)$. This is the basis of the spatial database query optimization illustrated in the next section. More formally,

Theorem 6.3 $proj(S, x)$ is the strongest Boolean constraint implied by $\exists x \,.\, S$.

Proof: It follows from Proposition 3.3 that $\exists x \,.\, S \Rightarrow proj(S, x)$. We now show that it is the strongest implied constraint. Let R be an (unquantified) system, such that $\exists x \,.\, S \Rightarrow R$ holds. With Theorem 6.1, we have $M \models proj(S, x) \Rightarrow R$ for all atomless Boolean algebras M. Now, a consequence of Corollary 5.7 is that a Bf-proposition P is valid in all Boolean algebras iff it is valid in all atomless Boolean algebras. Thus, $M' \models proj(S, x) \Rightarrow R$ for all Boolean algebras M'. ■

Example 6.2 Consider the system S, $x \cdot y \neq \emptyset \wedge \overline{x} \cdot y \neq \emptyset$, from above. In this case $proj(S, x)$ is $y \neq \emptyset$, the strongest implicant of $\exists x \,.\, S$. ■

Note that Theorem 6.1 and Theorem 6.3 first appeared in [9]. However the presentation and proofs given here are quite different and substantially simpler.

We can lift Theorem 6.3 to several existentially quantified variables by iteratively projecting on a single variable. To do this we extend the definition of *proj* to more than one variable by recursively defining $proj(S, x_1, x_2, \ldots, x_n)$ to be $proj(proj(S, x_2, \ldots, x_n), x_1)$.

Theorem 6.4 *proj*$(S, x_1, x_2, ..., x_n)$ is the strongest Boolean constraint implied by $\exists x_1 . \exists x_2 \ldots \exists x_n . S$.

Proof: The proof is by induction on the number of variables n. With Theorem 6.1 we have for in all atomless algebras M:

$$\begin{array}{ll} & proj(S, x_1, x_2, ..., x_n) \\ \Leftrightarrow & \exists x_1 . \exists x_2 \ldots \exists x_n . S \\ \Leftrightarrow & proj(\exists x_2 \ldots \exists x_n . S, x_1) \\ \Leftrightarrow & proj(proj(S, x_2, ..., x_n), x_1) \end{array}$$

With Corollary 5.7 this result carries over to all Boolean algebras. ■

7 Related Work

Our results concerning satisfiability and quantifier elimination fall between analogous results obtained for positive Boolean constraints by Boole [2], and the result of Tarski [19] that the elementary theory of Boolean algebras is decidable. Boole's results form the basis for so-called "Boolean unification" [4, 16] used in constraint logic programming systems that allow positive Boolean constraints.

To prove decidability of the elementary theory of Boolean algebras, Tarski showed that the theory supports quantifier elimination. This should not be confused with our result, as quantifier elimination for general formula with disjunctions does not imply that formula without disjunctions are also closed under existential quantification: in fact Example 6.1 showed that they are not. An alternative proof sketch that propositional formulae over atomless Boolean algebras are closed under quantification Corollary 6.2 may be found in Exercise 6.13 of Koppelberg [13]. Note that this implication works only in one direction: the Corollary (and proof sketch in Koppelberg) does not imply that propositional formulae without disjunctions are closed under existential quantification (Theorem 6.1).

Kozen [14] has shown that the decision problem for the elementary theory of Boolean algebras and many interesting subclasses including the atomless Boolean algebras is $\leq_{log}$-complete for $\cup STA(*, 2^{cn}, n)$ where $STA(*, 2^{cn}, n)$ is the class of sets accepted by an alternating Turing machine running in time 2^{cn} which may make only n alternations of universal and existential states, where n is the input length. Related results were also obtained by Berman [1]. Grädel [8] has shown that the subclasses of formula in which quantification alternation is bounded by m have essentially the same complexity as the entire theory whenever $m > 1$. He did not consider the case when $m = 1$, that is when the variables are either all existentially quantified or all universally quantified. A consequence of Theorem 5.8 is that, in this case, if all variables are existentially quantified the complexity is (only) NP complete and if they are all universally quantified then it is co-NP complete.

The most closely related result appears in Rudenau [17] who gives a characterization of satisfiability for systems of constraints in which there is a single negative constraint. However he states that the general problem with arbitrary negative constraints is still unsolved. In fact we show that weak independence is the key to lifting this result to the general case.

Recently there has been interest in weak independence, usually called independence in the literature, as a general means of lifting satisfiability and canonicity results from conjunctions of positive constraints to conjunctions with negative constraints. In particular Lassez and McAloon [15] studied canonical forms and Colmerauer [5] has investigated sufficient conditions for weak independence of equations and inequations in a general algebraic setting. However, Colmerauer's results do not apply in the Boolean domain as positive Boolean constraints do not admit "eliminable variables" in his precise sense. To our knowledge the notion of strong independence has not been explicitly identified before.

References

[1] P. Berman. Complexity of the theory of atomless Boolean algebras. In *Fundamentals of Computation Theory (FCT '79)*, pages 64–70, 1979.

[2] G. Boole. *An Investigation of the Laws of Thought.* Walton, London, 1847. (Reprinted by Philisophical Library, New York, 1954).

[3] F. Brown. *Boolean Reasoning, The Logic of Boolean Equations.* Kluwer Academic Publishers, Norwell, Massachusetts, USA, 1990.

[4] W. Buttner and H. Simonis. Embedding Boolean expressions into logic programming. *Journal of Symbolic Computation*, 4:191–205, 1987.

[5] A. Colmerauer. Equations and inequations on finite and infinite trees. In *International Conference on Fifth Generation Computer Systems*, pages 85–99. ICOT, 1984.

[6] A. Colmerauer. An introduction to Prolog III. *Communications of the ACM*, 33(7):69–90, 1990.

[7] M. Garey and D. Johnson. *Computers and Intractability.* W.H. Freeman, 1979.

[8] E. Grädel. Dominoe games with an application to the complexity of Boolean algebras with bounded quantifier alternations. In *Proc. Symp. on Theoretical Aspects of Computer Science*, volume 294, pages 98–107. Springer Verlag, Lecture Notes in Computer Science, 1988.

[9] R. Helm, K. Marriott, and M. Odersky. Constraint-based query optimization for spatial databases. In *Proc. ACM Symp. on Principles of Database Systems*, pages 181–191, 1991.

[10] R. Helm, K. Marriott, and M. Odersky. Spatial query optimization: From Boolean constraints to range queries. Research Report 17231, IBM Thomas J. Watson Research Center, 1991. (Also submitted to JCSS).

[11] J. Jaffar and J.-L. Lassez. Constraint logic programming. In *ACM Symposium on Principles of Programming Languages*, pages 111–119, 1987.

[12] P. C. Kanellakis, G. M. Kuper, and P. Z. Revesz. Constraint query languages. In *Proc. ACM Symp. on Principles of Database Systems*, pages 299–313, 1990.

[13] S. Koppelberg. *The Handbook of Boolean Algebras (Vol. I).* Elsevier Science Publishers B.V.(North Holland), New York, N.Y., 1989.

[14] D. Kozen. Complexity of Boolean algebras. *Theoretical Computer Science*, 10:221–247, 1980.

[15] J.-L. Lassez and K. McAloon. A constraint sequent calculus. In *IEEE Symposium on Logic in Computer Science*, pages 52–61. IEEE Press, 1990.

[16] U. Martin and T. Nipkow. Boolean unification – the story so far. *Journal of Symbolic Computation*, 7:275–293, 1989.

[17] S. Rudeanu. *Boolean Functions and Equations.* Elsevier Science Publishers B.V.(North Holland), New York, N.Y., 1974.

[18] H. Simonis and M. Dincbas. Propositional calculus problems in CHIP. In *2nd International Conference on Algebraic and Logic Programming*, volume 490, pages 189–203. Springer Verlag, Lecture Notes in Computer Science, 1990.

[19] A. Tarski. Arithmetical classes and types of Boolean algebras. *Bull. Amer. Math. Soc*, 55(64):1192, 1949.

[20] D. Willard. Efficient processing of relational calculus using range query theory. In *Proc. ACM Symp. on Principles of Database Systems*, pages 164–175, 1984.