Hindawi Publishing Corporation
Journal of Applied Mathematics and Decision Sciences
Volume 2009, Article ID 845804, 18 pages
doi:10.1155/2009/845804

Research Article

On the Complexities of Selected Satisfiability and Equivalence Queries over Boolean Formulas and Inclusion Queries over Hulls

K. Subramani

LDCSEE, West Virginia University, Morgantown, WV 26506, USA

Correspondence should be addressed to K. Subramani, ksmani@csee.wvu.edu

Received 5 November 2008; Accepted 14 April 2009

Recommended by Ron McGarvey

This paper is concerned with the computational complexities of three types of queries, namely, satisfiability, equivalence, and hull inclusion. The first two queries are analyzed over the domain of CNF formulas, while hull inclusion queries are analyzed over continuous and discrete sets defined by rational polyhedra. Although CNF formulas can be represented by polyhedra over discrete sets, we analyze them separately on account of their distinct structure. In particular, we consider the NAESAT and XSAT versions of satisfiability over HornCNF, 2CNF, and Horn$\oplus$2CNF formulas. These restricted families find applications in a number of practical domains. From the hull inclusion perspective, we are primarily concerned with the question of checking whether two succinct descriptions of a set of points are equivalent. In particular, we analyze the complexities of integer hull inclusion over 2SAT and Horn polyhedra. Hull inclusion problems are important from the perspective of deriving minimal descriptions of point sets. One of the surprising consequences of our work is the stark difference in complexities between equivalence problems in the clausal and polyhedral domains for the same polyhedral structure.

1. Introduction

The problem of testing the satisfiability of CNF formulas (or SAT) is ubiquitous in computer science and operations research. Applications of this problem abound from areas as diverse as econometrics and planning to graph theory and combinatorial optimization [1]. From the perspective of computational complexity, SAT was immortalized in [2] as the first natural `NP-complete` problem. Advances in SAT research have been along both theoretical and practical lines. On the theoretical side, there exist a number of algorithms running in time $o(2^n)$, for instance, see [3–5]. On the practical front, greedy approaches based on random walks have been enormously successful [6, 7].

One of the approaches taken by SAT theoreticians is to identify structures of SAT families that are amenable to solution through polynomial time procedures. This approach has the benefit of defining yardsticks that categorize classes of SAT problems as polynomial time solvable or `NP-complete` [8]. Tractable families are distinguished by a specific clausal structure; these structures either limit the number of literals per clause [9, 10] or the number of times that a variable appears across all the clauses [11]. A completely orthogonal approach to SAT research is to study the complexities of SAT variants, that is, satisfiability problems with additional requirements on the type of solutions. For instance, in the NAESAT problem, we are required to find an assignment that satisfies all the clauses but falsifies at least one literal per clause. This paper is interested in such syntactic variants of the SAT problem, inasmuch as these variants arise naturally in the domains of graph theory and scheduling.

This paper also focuses on hull inclusion problems over continuous and discrete sets of points. Hull inclusion is checkable in polynomial time when the domain is continuous and `NP-complete` when the domain is discrete. However, there exist nontrivial cases of hull inclusion in discrete domains which are polynomial time solvable on account of the structure of the constraint matrix. These hull inclusion problems find applications in program verification [12, 13]. Although SAT problems can be cast as discrete domain polyhedral problems, we choose to treat them separately in order to exploit their structure. In this context, we will show that a particular problem is polynomial time solvable over Boolean CNF formulas, but provably hard over arbitrary polyhedra.

The principal contributions of this paper are as follows.

(a) Establishing the complexities of NAESAT and XSAT queries over restricted CNF families (see Section 2 for definitions of CNF restrictions).

(b) Introducing the problems of NAE-equivalence and X-equivalence and establishing their complexities over restricted CNF families.

(c) Developing a polynomial time algorithm for Linear hull inclusion.

(d) Establishing the complexities of Integer hull inclusion over various polyhedral families.

The rest of this paper is organized as follows. Section 2 formally describes each of the problems considered in this paper. In Section 3, we discuss the motivation for our work as well as related approaches in the literature. The complexities of satisfiability queries over various clausal families are detailed in Section 4. Boolean equivalence queries and their variants are discussed in Section 5. An algorithm for Linear hull inclusion is discussed in Section 6. Section 7 is concerned with Integer hull inclusion over various polyhedral families. We conclude in Section 8 by summarizing our work in this paper and identifying avenues for future research.

2. Statement of Problems

We begin with definitions of satisfiability-related problems on clausal Boolean formulas.

Let $\phi = C_1 \wedge C_2 \wedge \ldots C_m$ denote a Boolean formula in conjunctive normal form (CNF), where the C_is are disjunctions on the literals $\{x_1, \overline{x}_1, x_2, \overline{x}_2, \ldots, x_n, \overline{x}_n\}$. For the rest of the paper, we assume that our formulas are in CNF over the n variables $\{x_1, x_2, \ldots x_n\}$, unless otherwise stated.

Definition 2.1. The Boolean satisfiability (SAT) problem is: Given ϕ, is there a $\{$**true, false**$\}$ assignment to the variables of ϕ, such that at least one literal in each clause is set to **true**.

If such an assignment $\overrightarrow{\mathbf{x}}$ exists for a given CNF formula ϕ, then ϕ is said to be satisfiable and $\overrightarrow{\mathbf{x}}$ is said to be a satisfying assignment. There are other variants of satisfiability of interest in the CNF case, where the number of true literals or the mix of true and false literals per clause is specified.

Definition 2.2. The Not-All-Equal satisfiability (NAESAT) problem is defined as follows: Given a Boolean formula ϕ, does ϕ have a satisfying assignment, such that at least one literal in each clause is set to **false**. If such an assignment exists, then it is called a NAE-satisfying assignment and ϕ is said to be NAE-satisfiable.

Definition 2.3. The Exact satisfiability (XSAT) problem is defined as follows. Given a Boolean formula ϕ, does ϕ have a satisfying assignment, such that exactly one literal in each clause is set to **true**. If such an assignment exists, then it is called an X-satisfying assignment and ϕ is said to be X-satisfiable.

If a Boolean formula ϕ is unsatisfiable, then it is neither NAE-satisfiable nor X-satisfiable; however, satisfiability does not imply NAE-satisfiability or X-satisfiability.

This paper will focus on Boolean formulas with restrictions on the structure of the formula. These restrictions are exploited in the design of polynomial time algorithms.

Definition 2.4. A Boolean formula ϕ is in k-CNF form, if every clause contains exactly k literals.

Definition 2.5. A Boolean formula ϕ is Horn, if every clause contains at most one positive literal.

Definition 2.6. A Boolean formula ϕ is said to be Horn$\oplus$2CNF, if each clause contains at most two literals or at most one positive literal.

Given an assignment $a_1 \in \{\mathbf{true}, \mathbf{false}\}^n$ to the variables of a boolean formula ϕ, let $T(a_1)$ denote the set of variables that have been assigned **true** under a_1.

Definition 2.7. A Boolean formula ϕ is said to be positively monotone, if for every pair of assignments $a_1, a_2 \in \{\mathbf{true}, \mathbf{false}\}^n$ to the variables of ϕ, $(T(a_1) \subseteq T(a_2)) \Rightarrow (\phi(a_1) \Rightarrow \phi(a_2))$.

Definition 2.8. A Boolean formula ϕ is said to be negatively monotone, if for every pair of assignments $a_1, a_2 \in \{\mathbf{true}, \mathbf{false}\}^n$ to the variables of ϕ, $(T(a_1) \subseteq T(a_2)) \Rightarrow (\phi(a_2) \Rightarrow \phi(a_1))$.

A Boolean formula ϕ is monotone if it is either positively monotone or negatively monotone. It is not hard to see that if ϕ is a CNF formula, then ϕ is monotone if all variables occur positively or all variables occur negatively.

It is well known that 3SAT, NAE3SAT, and X3SAT are `NP-complete` [11], whereas HornSAT and 2SAT are solvable in polynomial time [10]. Horn$\oplus$2CNF formulas have also been called Mixed Horn Formulas in the literature [14] and satisfiability checking in these formulas is also `NP-complete` [10]. Monotone SAT is trivial, since all variables can be set to **true** (or **false**). Other monotone problems, however, prove more interesting.

From the perspective of computational complexity, the following questions are open:

(i) What are the complexities of NAESAT over Horn and Mixed Horn formulas? These problems are called NAEHornSAT and NAEMixedHornSAT, respectively.

(ii) What are complexities of XSAT over Horn and Mixed Horn formulas? These problems are called XHornSAT and XMixedHornSAT, respectively.

A problem closely related to satisfiability is the boolean equivalence problem.

Definition 2.9. Two Boolean formulas ϕ_1 and ϕ_2 are said to be equivalent (denoted $\phi_1 \Leftrightarrow \phi_2$) if every assignment that satisfies ϕ_1 also satisfies ϕ_2 and vice versa.

The problem of checking whether two Boolean formulas are equivalent is denoted by BEQ.

Observe that the equivalence problem can be broken into two subproblems, namely, $\phi_1 \Rightarrow \phi_2$ and $\phi_2 \Rightarrow \phi_1$.

Consider the subproblem $\phi_1 \Rightarrow \phi_2$. As before, we assume that $\phi_1 = C_1 \wedge C_2 \cdots \wedge C_m$ and that $\phi_2 = C'_1 \wedge C'_2 \cdots C'_{m'}$. Now,

$$\phi_1 \Longrightarrow \phi_2 \Longleftrightarrow [\phi_1 \Longrightarrow C'_1 \wedge C'_2 \cdots C'_{m'}] \Longleftrightarrow \wedge_{i=1}^{m'} \left[\phi_1 \Rightarrow C'_j\right]. \tag{2.1}$$

System (2.1) exploits the well-known propositional tautology, for arbitrary propositional formulas A, B, and C:

$$((A \Longrightarrow B) \wedge (A \Longrightarrow C)) \Longleftrightarrow (A \Longrightarrow (B \wedge C)). \tag{2.2}$$

Pick a particular clause $C'_i \in \phi_2$. Applying the tautology

$$(A \Longrightarrow B) \Longleftrightarrow \left(A \wedge \overline{B} \Longleftrightarrow \mathbf{false}\right) \tag{2.3}$$

for arbitrary propositional formulas A and B, it follows that $[\phi_1 \Rightarrow C'_i]$ if and only if $[\phi_1 \wedge \overline{C'_i}]$ is unsatisfiable.

Observe that C'_i is a disjunction of literals, so that $\overline{C'_i}$ is a conjunction of unit literal clauses; hence $\phi_1 \wedge \overline{C'_i}$ is in CNF form. Thus, in order to check whether $\phi_1 \Rightarrow \phi_2$, we merely need to confirm that all the CNF formulas in the set $\{\phi_1 \wedge \overline{C'_i}, \phi_1 \wedge \overline{C'_2}, \ldots, \phi_1 \wedge \overline{C'_{m'}}\}$ are unsatisfiable. In other words, the implication problem for CNF problems has been Turing reduced to the CNF unsatisfiability problem. Then, we have the following lemma.

Lemma 2.10. *BEQ is Turing reducible to the problem of checking whether a CNF formula is unsatisfiable.*

Proof. Given ϕ_1 and ϕ_2, first check whether $\phi_1 \Rightarrow \phi_2$ and then whether $\phi_2 \Rightarrow \phi_1$. □

Observe that there exist polynomial time algorithms to decide (un)satisfiability in 2CNF and HornCNF formulas [10], and hence it follows that BEQ can be decided in polynomial time for 2CNF and HornCNF formulas.

Two problems that are closely related to the boolean equivalence problem are the NAE-equivalence problem and the X-equivalence problem.

Definition 2.11. Two Boolean formulas ϕ_1 and ϕ_2 are said to be NAE-equivalent (denoted $\phi_1 \Leftrightarrow_n \phi_2$) if every assignment that NAE-satisfies ϕ_1 also NAE-satisfies ϕ_2 and vice versa.

The problem of checking whether two boolean formulas are NAE-equivalent is denoted by NAEEQ.

Definition 2.12. Two Boolean formulas ϕ_1 and ϕ_2 are said to be X-equivalent (denoted by $\Leftrightarrow_x$) if every assignment that X-satisfies ϕ_1 also X-satisfies ϕ_2 and vice versa.

The problem of checking whether two boolean formulas are X-equivalent is denoted by XEQ.

Problem 1. Are there classes of Boolean formulas for which NAE-equivalence and X-equivalence can be determined in polynomial time?

We now proceed to describe the linear and integer hull inclusion problems for particular types of polytopes. A polytope is a bounded polyhedron. A polyhedron is defined by a linear system $\mathbf{A}\cdot\overrightarrow{\mathbf{x}} \leq \overrightarrow{\mathbf{b}}$, where $\mathbf{A}$ is an $m \times n$ integer matrix, $\overrightarrow{\mathbf{b}}$ is an integral m-vector, and $\overrightarrow{\mathbf{x}} = [x_1, x_2, \ldots, x_n]^T$ is the variable vector. Polytopes and polyhedra are interesting because they capture a number of problems that arise in combinatorial optimization.

For the following definitions, assume that we are given two polytopes $P_1 : \{\mathbf{A_1}\cdot\overrightarrow{\mathbf{x}} \leq \overrightarrow{\mathbf{b_1}}\}$ and $P_2 : \{\mathbf{A_2}\cdot\overrightarrow{\mathbf{x}} \leq \overrightarrow{\mathbf{b_2}}\}$.

Definition 2.13. The linear hull of a polytope $P_1 : \{\mathbf{A_1}\cdot\overrightarrow{\mathbf{x}} \leq \overrightarrow{\mathbf{b_1}}\}$, denoted by L_{P_1}, is defined as the convex hull of all the points contained in it.

Definition 2.14 (see [15]). The integer hull of a polytope $P_1 : \{\mathbf{A_1}\cdot\overrightarrow{\mathbf{x}} \leq \overrightarrow{\mathbf{b_1}}\}$, denoted by S_{P_1}, is defined as the convex hull of the lattice points contained in P_1.

Note that for a given polytope $\mathbf{A_1}\cdot\overrightarrow{\mathbf{x}} \leq \overrightarrow{\mathbf{b}}$, both the linear hull and the integer hull have exponentially many extreme points. However, just as we are interested in equivalence between formulas with respect to various satisfiability measures, we are interested in comparing polytopes via their convex hulls.

Definition 2.15. The linear hull inclusion (LHI) problem is defined as follows. Given polyhedra $P_1$and P_2, is it the case that $L_{P_1} \subseteq L_{P_2}$?

Definition 2.16. The integer hull inclusion (IHI) problem is defined as follows. Given polyhedra P_1andP_2, is it the case that $S_{P_1} \subseteq S_{P_2}$?

We first observe that the IHI problem is hard in the general case.

Lemma 2.17. *IHI is* coNP-complete.

Proof. Let ϕ denote an arbitrary 3CNF formula. It is well known that this formula can be represented as an integer program over $\{0,1\}$, for instance, see [16]. Set P_1 to be the polyhedral system representing the 3CNF formula and P_2 to the empty polyhedron $\{x_1 \geq 1, x_1 \leq 0\}$. The integer hull of P_1 is contained in the integer hull of P_2 if and only if the 3CNF formula ϕ is unsatisfiable. Then the lemma follows. □

The natural question then is the following.

Problem 2. Are there classes of polytopes for which IHI can be decided in polynomial time?

We now define some special classes of polytopes. The structure of these polytopes will be exploited to design polynomial time algorithms for one or both types of hull inclusion.

Definition 2.18. A polyhedral system $P_1 : \{\mathbf{A} \cdot \overrightarrow{\mathbf{x}} \leq \overrightarrow{\mathbf{b}}\}$ is said to be a 2SAT polytope if all entries of $\mathbf{A}$ belong to the set $\{0, 1, -1\}$ and further, there are at most 2 nonzero entries per row of $\mathbf{A}$.

Individual constraints of a 2SAT polytope have also been referred to as UTVPI constraints in the literature [13].

Definition 2.19. A polyhedral system of the form $P_1 : \{\mathbf{A} \cdot \overrightarrow{\mathbf{x}} \geq \overrightarrow{\mathbf{b}}\}$ is said to be a Horn polytope if all entries in $\mathbf{A}$ belong to $\{1, 0, -1\}$, and there exists at most one positive entry in each row.

Observe that 2SAT polytopes and Horn polytopes generalize 2CNF clauses and HornCNF clauses, respectively.

Definition 2.20. A matrix $\mathbf{A}$ is said to be totally unimodular (TUM) if every square submatrix of $\mathbf{A}$ has determinant 0, 1, or −1.

TUM matrices arise in network flow problems [17], scheduling problems [18], and a whole host of situations in which only strict difference constraints are permitted between program variables [19]. One of the more celebrated results about TUM matrices in the operations research literature is the following. Let $\mathbf{P_1} : \mathbf{A} \cdot \overrightarrow{\mathbf{x}} \leq \overrightarrow{\mathbf{b}}$ denote a polyhedral system, with $\mathbf{A}$ TUM and $\overrightarrow{\mathbf{b}}$ integral. Then, the integer hull and linear hull of $\mathbf{P_1}$ are identical (see [15]).

Definition 2.21. A polyhedral system of the form $P_1 : \{\mathbf{A} \cdot \overrightarrow{\mathbf{x}} \geq \overrightarrow{\mathbf{b}}\}$, where all entries in $\mathbf{A}$ belong to $\{1, 0, -1\}$ and each row has exactly one +1 and one −1, is called a Difference polytope.

Note that the class of difference polytopes is a proper subset of the class of 2SAT polytopes and also a proper subset of the class of Horn polytopes. (See Figure 1.)

Difference polytopes are a special subset of the class of TUM polytopes. Conjunctions of difference constraints are used to capture requirements in a number of application domains such as symbolic model checking [20], verification of timed systems [21, 22], and timed automata [23, 24]. Difference constraint feasibility has also been studied as the Single Source Shortest Paths problem within the operations research and algorithms communities [25]. Additionally, separation relationships in a number of scheduling problems are captured through difference constraints [18, 26, 27]. In real-time software, temporal requirements are modeled through variants of difference constraints [28, 29].

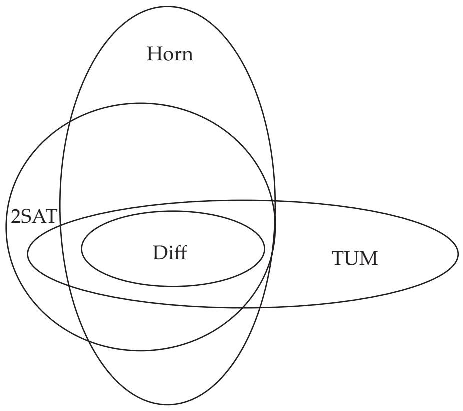

Figure 1: Relation between polyhedral classes.

3. Motivation and Related Work

The two variants of SAT that we are interested in are NAESAT and XSAT. Both these problems have a long and interesting history, inasmuch as they are closely linked to graph coloring problems [30]. Mixed Horn Formulas (MHFs) have been used in the formulation of level-planarity tests in planar graphs and crossing-minimization problems [31]. In [14], it has been argued that graph colorability can also be formulated as an MHF satisfiability problem.

Convex body inclusion has been well studied in the operations research literature [32, 33]. Convex body inclusion is related to the problem of estimating the volumes of convex bodies [34, 35]. In Section 6, we present an algorithm for polytope inclusion that depends on the fact that maximizing a linear function over a polytope can be accomplished in polynomial time. Integer hull inclusion is different from convex body inclusion since in general, the set of lattice points satisfying a system of linear inequalities do not form a convex set. 2SAT polytopes were introduced in [36], where it was shown that the existence of a lattice point could be determined in $O(n^3)$ time, where n is the dimension of the polytope. It is known that the problem of obtaining the integer maximum of a linear function over a 2SAT polytope is, in general, NP-hard. (This problem generalizes the vertex cover problem.) Horn polytopes generalize Horn clausal systems [37] and find wide application in linear complementarity and econometrics research [38].

A secondary motivation for the study of clausal equivalence problems is provided by classical computational complexity. One of the goals in complexity is to find the exact threshold at which problems become hard [10]. This paper shows that the NAE-equivalence and X-equivalence problems are solvable in polynomial time for 2CNF formulas but are provably hard for the other boolean families. Likewise, we show that the integer hull inclusion problem can be solved in polynomial time for 2SAT polytopes. A surprising consequence of our work is the apparent disparity between the clausal and integer programming versions of Horn equivalence. Clausal Horn equivalence can be determined in polynomial time, whereas integer hull inclusion over Horn polytopes is coNP-complete.

4. Boolean Satisfiability Queries

This section focuses on the computational complexity of the satisfiability queries detailed in Section 2. It has been shown that the NAE2SAT problem can be solved in polynomial time [39]. Indeed, the results in [40] establish that NAE2SAT is in the complexity class `L` (Deterministic Logarithmic Space). Likewise, NAE3SAT has been proven `NP-complete` in [8]. The arguments in [8] also establish that Monotone XSAT is `NP-complete`.

We are now ready to prove the following theorem.

Theorem 4.1. *XHornSAT is* `NP-complete`*.*

The proof is a reduction from Monotone XSAT to XHornSAT. Before we construct the reduction, we will prove a minor but useful lemma.

Lemma 4.2. *Any X-satisfying assignment for the formula* $(\overline{z_1} \vee \overline{z_2}) \wedge (\overline{z_1} \vee \overline{z_2} \vee z_3)$ *sets* z_3 *to false.*

Proof. We note that any X-satisfying assignment for $(\overline{z_1} \vee \overline{z_2})$ sets one of z_1 and z_2 to true and the other to false. Therefore, a consistent X-satisfying assignment to $(\overline{z_1} \vee \overline{z_2} \vee z_3)$ must set z_3 to **false**. □

Note that the formula in Lemma 4.2 is Horn.

Proof of Theorem 4.1. Let $\varphi = C_1 \wedge C_2 \wedge \cdots C_k$ be a monotone CNF formula. Let z_1, z_2, and z_3 be variables that do not occur in φ, and assume without loss of generality that all variables in φ appear *negated*.

Let

$$f(\varphi) = (\overline{z_1} \vee \overline{z_2}) \wedge (\overline{z_1} \vee \overline{z_2} \vee z_3) \wedge_{i \le k} [(\neg C_i) \longrightarrow z_3]. \tag{4.1}$$

Observe that if $C_i = (\overline{x}_1, \overline{x}_2, \overline{x}_3)$, then $[(\neg C_i) \to z_3]$ has the form: $[(x_1 \wedge x_2 \wedge x_3) \to z_3]$. As discussed in Lemma 4.2, any X-satisfying assignment to $f(\varphi)$ must set z_3 to **false**. However, any X-satisfying assignment for a Horn clause having the form $[(x_i \wedge x_j \wedge x_k) \to$ **false**$]$ must set exactly one literal of $(x_i \wedge x_j \wedge x_k)$ to **true**. It follows that $f(\varphi)$ is X-satisfiable if and only if φ is. Since $f(\varphi)$ can be computed in time linear in $|\varphi|$, this reduction is polynomial-time computable.

It follows that XHornSAT is `NP-complete`. □

There is a similar reduction from monotone NAESAT to NAEHornSAT. The reduction itself is completely straightforward since a monotone clause C is equivalent to $(\neg C \to$ **false**$)$. (Again, without loss of generality we assume that all variables in C are negated.)

Corollary 4.3. *Monotone NAESAT* $\le_m^P$ *NAEHornSAT, where the* $\le_m^P$ *stands for many-to-one polynomial time.*

In order to show the complexity of these two satisfiability problems, we introduce a coloring problem.

Definition 4.4. The Set Splitting problem is: given a universe $U = \{x_1, \ldots, x_n\}$ and $S \subseteq \mathcal{P}(U)$ (subsets of U), is there a coloring of $\{x_1, \ldots, x_n\}$ by two colors, say red and blue, so that no set in S is monochrome?

Table 1: Complexities of various satisfiability queries over different clausal families.

Problem	2CNF	HornCNF	3CNF	Horn⊕2CNF
SAT	P	P	NP-c	NP-c
NAESAT	P	NP-c	NP-c	NP-c
XSAT	P	NP-c	NP-c	NP-c

Sipser gives the NP completeness of Set Splitting as an exercise [41]. There is a straightforward mapping between the sets in S and clauses in a monotone formula and between the colors and the values **true** and **false**.

Lemma 4.5. *The Splitting Set problem is `NP-complete`, and Monotone NAESAT is equivalent to it.*

Corollary 4.6. *NAEHornSAT is NP-complete.*

Note that any NAESAT assignment to a Horn clause must set at least one body variable to **false**. Otherwise, the NAE constraint would force the body to **true** and the head to **false**, and the assignment would not satisfy the clause.

Observe that NAEHornSAT and XHornSAT are both `NP-complete` as shown in Theorem 4.1 and Corollary 4.6, respectively. Furthermore, HornSAT is a subclass of MixedHornSAT. Finally, note that the reductions that established the `NP-completeness` of NAEHornSAT and XHornSAT have the properties that they map inputs either to NAE-satisfiable (resp, X-satisfiable) Horn formulas, or to NAE-unsatisfiable (resp, X-unsatisfiable) Horn formulas. Therefore, these very reductions establish the `NP-completeness` of NAEMixedHornSAT and XMixedHornSAT, respectively.

Corollary 4.7. *NAEMixedHornSAT and XMixedHornSAT are `NP-complete`.*

Table 1 summarizes our discussion on clausal satisfiability queries.

5. Boolean Equivalence Queries

The reasoning in Section 2 can be used to establish that BEQ is co`NP-complete` over 3CNF and Mixed Horn Formulas.

Theorem 5.1. *BEQ is co`NP-complete` for 3CNF and Mixed Horn Formulas.*

Proof. Let ϕ_1 denote an arbitrary 3CNF formula or Mixed Horn formula. We know that the problem of checking whether ϕ_1 is satisfiable is `NP-complete`; thus, if we set ϕ_2 to **false**, we can determine the satisfiability of ϕ_1 by checking whether $\phi_1 \Leftrightarrow \phi_2$. □

In similar fashion, it can be shown that the NAEEQ and XEQ problems are `coNP-complete` for 3CNF and Mixed Horn Formulas.

We now prove a general theorem relating the NAEEQ and BEQ problems. Let ϕ denote a formula in k-CNF form, and let ϕ' denote the CNF formula obtained by negating every literal in every clause of ϕ.

Lemma 5.2. *ϕ is NAE-satisfiable if and only if $\phi \wedge \phi'$ is satisfiable.*

Table 2: Complexities of equivalence queries over different clausal families.

Equivalence type	2CNF	HornCNF	3CNF	Horn⊕2CNF
Simple	P	P	coNP-complete	coNP-complete
NAE-equivalence	P	coNP-complete	coNP-complete	coNP-complete
X-equivalence	P	coNP-complete	coNP-complete	coNP-complete

Proof. **Only if:** Let ϕ be NAE-satisfiable and let $\overrightarrow{\mathbf{x}}$ denote a NAE-satisfying assignment. Let C_1 denote a clause of ϕ. Since $\overrightarrow{\mathbf{x}}$ is a NAE-satisfying assignment, it sets at least one literal to **true** and at least one literal to **false** in C_1. Let C_1' denote the clause in which each literal in C_1 is complemented. Note that the literal that was set to **false** in C_1 is set to **true** in C_1' and vice versa; since C_1 was chosen arbitrarily, the same argument applies to all clauses of ϕ. In other words, $\overrightarrow{\mathbf{x}}$ NAE-satisfies ϕ', and therefore, $\overrightarrow{\mathbf{x}}$ NAE-satisfies $\phi \wedge \phi'$. Hence $\phi \wedge \phi'$ is satisfiable.

If: Let $\phi \wedge \phi'$ be satisfiable in the ordinary sense, and let $\overrightarrow{\mathbf{x}}$ denote a satisfying assignment. Let us study the clauses C_1 and C_1' under the assignment $\overrightarrow{\mathbf{x}'}$ which is obtained by complementing every assignment in $\overrightarrow{\mathbf{x}}$. As per the definition of ϕ', every literal that is set to **true** in C_i is set to **false** in C_1' and vice versa. It therefore follows that $\overrightarrow{\mathbf{x}'}$ also satisfies both C_1 and C_1'. Inasmuch as C_1 and C_1' were chosen arbitrarily, the same argument holds for all clause pairs, that is, $\overrightarrow{\mathbf{x}'}$ satisfies $\phi \wedge \phi'$. Now, $\phi \wedge \phi'$ has a complementary pair of assignments and therefore is NAE-satisfiable. But this immediately implies the NAE-satisfiability of ϕ. □

Lemma 5.2 leads us directly to the following lemma.

Lemma 5.3. *($\phi \Leftrightarrow_n \psi$) if and only if ($\phi \wedge \phi'$) $\Leftrightarrow$ ($\psi \wedge \psi'$).*

Proof. Suppose that ($\phi \Leftrightarrow_n \psi$) but ($\phi \wedge \phi'$) $\not\Leftrightarrow$ ($\psi \wedge \psi'$). Without loss of generality, assume that there exists an assignment $\overrightarrow{\mathbf{x}}$ which satisfies $\phi \wedge \phi'$ but falsifies $\psi \wedge \psi'$. Since $\overrightarrow{\mathbf{x}}$ does not satisfy $\psi \wedge \psi'$, it cannot NAE-satisfy ψ, as per Lemma 5.2. As per the hypothesis, it cannot NAE-satisfy ϕ either. But since $\overrightarrow{\mathbf{x}}$ satisfies $\phi \wedge \phi'$, it must NAE-satisfy ϕ, as per Lemma 5.2, and thus we have a contradiction. It follows that ($\phi \Leftrightarrow_n \psi$) implies that ($\phi \wedge \phi'$) $\Leftrightarrow$ ($\psi \wedge \psi'$). The converse can be proved similarly. □

Note that in case of 2CNF formulas, the NAEEQ and XEQ problems are identical.

The next theorem follows from the previous proofs.

Theorem 5.4. *The NAEEQ and XEQ problems can be solved in polynomial time for 2CNF formulas.*

Table 2 summarizes our discussion on clausal equivalence queries.

6. Linear Hull Inclusion

In this section, we focus on the problem of checking whether the linear hulls of two polyhedra defined by systems of linear inequalities are equivalent.

Consider two polyhedra represented by

$$\mathbf{A} \cdot \overrightarrow{\mathbf{x}} \leq \overrightarrow{\mathbf{b}}, \; \overrightarrow{\mathbf{x}} \geq \overrightarrow{\mathbf{0}}, \tag{6.1}$$

```
Function Polytope-Include $(\mathbf{A}, \vec{\mathbf{b}}, \mathbf{C}, \vec{\mathbf{d}})$
(1) {Let the $i$th constraint of $\mathbf{C} \cdot \vec{\mathbf{x}} \le \vec{\mathbf{d}}, \vec{\mathbf{x}} \ge \vec{\mathbf{0}}$ be represented as : $\vec{\mathbf{c}}_{\mathbf{i}} \cdot \vec{\mathbf{x}} \le \mathbf{d_i}$}.
(2) for($i = 1$ to $m'$) do
(3)     if ($\max_{\mathbf{A} \cdot \vec{\mathbf{x}} \le \vec{\mathbf{b}}, \vec{\mathbf{x}} \ge \vec{\mathbf{0}}} \vec{\mathbf{c}}_{\mathbf{i}} \cdot \vec{\mathbf{x}} > \mathbf{d_i}$) then
(4)         return (false)
(5)     end if
(6) end for
(7) return (true)
```

Algorithm 1: Algorithm for polytope inclusion.

where

(1) $\mathbf{A}$ is an $m \times n$ rational matrix,

(2) $\vec{\mathbf{b}}$ is a rational m−vector,

(3) $\vec{\mathbf{x}} \in \Re_+^{\mathbf{n}}$,

and

$$\mathbf{C} \cdot \vec{\mathbf{x}} \le \vec{\mathbf{d}}, \ \vec{\mathbf{x}} \ge \vec{\mathbf{0}}, \tag{6.2}$$

where

(1) $\mathbf{C}$, is an $m' \times n$ rational matrix,

(2) $\vec{\mathbf{d}}$ is a rational m'-vector,

(3) $\vec{\mathbf{x}} \in \Re_+^{\mathbf{n}}$

The goal is to decide the following predicate:

$$\left(\forall \vec{\mathbf{x}} \ge \vec{\mathbf{0}}\right)\left(\mathbf{A} \cdot \vec{\mathbf{x}} \le \vec{\mathbf{b}} \Longrightarrow \mathbf{C} \cdot \vec{\mathbf{x}} \le \vec{\mathbf{d}}\right)? \tag{6.3}$$

Algorithm 1 represents our strategy to decide Query (6.3).

6.1. Analysis

Let $\mathcal{L}(m, n)$ denote the polynomial running time of a linear programming algorithm on m constraints and n variables [42]. Since a total of m' calls are made, the running time of Algorithm 1 is $O(m' \cdot \mathcal{L})$, which is polynomial, since $\mathcal{L}$ is a polynomial function of m and n.

6.2. Correctness

Lemma 6.1. *If Algorithm 1 returns* **true**, *then for all* $\vec{\mathbf{x}} \in \Re_+^{\mathbf{n}}$, $\mathbf{A} \cdot \vec{\mathbf{x}} \le \vec{\mathbf{b}} \Rightarrow \mathbf{C} \cdot \vec{\mathbf{x}} \le \vec{\mathbf{d}}$.

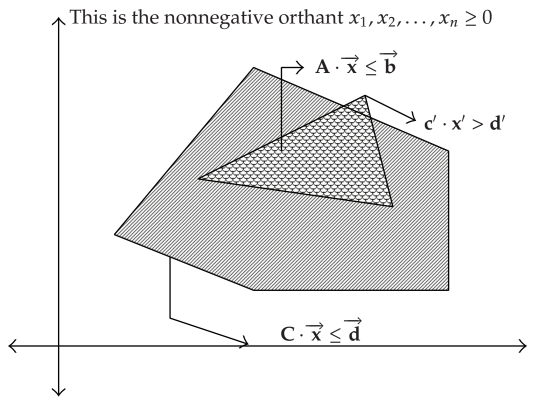
This is the nonnegative orthant $x_1, x_2, \ldots, x_n \geq 0$
$\mathbf{A} \cdot \overrightarrow{\mathbf{x}} \leq \overrightarrow{\mathbf{b}}$
$\mathbf{c}' \cdot \mathbf{x}' > \mathbf{d}'$
$\mathbf{C} \cdot \overrightarrow{\mathbf{x}} \leq \overrightarrow{\mathbf{d}}$

Figure 2: Polytope noninclusion.

Proof. Let us assume the contrary, that is, Algorithm 1 returns **true**, yet there exists a point $\overrightarrow{\mathbf{x}'}$ such that $\mathbf{A} \cdot \overrightarrow{\mathbf{x}'} \leq \overrightarrow{\mathbf{b}}$, but $\mathbf{C} \cdot \overrightarrow{\mathbf{x}'} \nleq \overrightarrow{\mathbf{d}}$. We note that the notation $\mathbf{C} \cdot \overrightarrow{\mathbf{x}'} \nleq \overrightarrow{\mathbf{d}}$ is used to indicate the fact that at least one of the m' constraints defining the polyhedron $\mathbf{C} \cdot \overrightarrow{\mathbf{x}} \leq \overrightarrow{\mathbf{d}}$ is violated. Let $\overrightarrow{\mathbf{c}'} \cdot \overrightarrow{\mathbf{x}} \leq d'$ denote a violated constraint, that is, $\overrightarrow{\mathbf{c}'} \cdot \overrightarrow{\mathbf{x}'} > d'$. (See Figure 2.) Then $\max_{\mathbf{A} \cdot \overrightarrow{\mathbf{x}} \leq \overrightarrow{\mathbf{b}}, \overrightarrow{\mathbf{x}} \geq \overrightarrow{\mathbf{0}}} \overrightarrow{\mathbf{c}'} \cdot \overrightarrow{\mathbf{x}}$ is greater than d', contradicting the hypothesis that **true** was returned by the algorithm. □

Lemma 6.2. *If Algorithm 1 returns **false**, there exists a point* $\overrightarrow{\mathbf{x}'} \in \mathbf{R}^{\mathbf{n}}_{+}$ *such that* $\mathbf{A} \cdot \overrightarrow{\mathbf{x}'} \leq \overrightarrow{\mathbf{b}}$ *and* $\mathbf{C} \cdot \overrightarrow{\mathbf{x}'} \nleq \overrightarrow{\mathbf{d}}$.

Proof. See Figure 2. Let $\max_{\mathbf{A} \cdot \overrightarrow{\mathbf{x}} \leq \overrightarrow{\mathbf{b}}} \overrightarrow{\mathbf{c}'} \cdot \overrightarrow{\mathbf{x}}$ exceed d' at point $\overrightarrow{\mathbf{x}'}$. Then $\overrightarrow{\mathbf{x}'}$ is the required offending point, that is, $\overrightarrow{\mathbf{x}'} \in \mathbf{A} \cdot \overrightarrow{\mathbf{x}} \leq \overrightarrow{\mathbf{b}}, \overrightarrow{\mathbf{x}} \geq \overrightarrow{\mathbf{0}}$, but $\overrightarrow{\mathbf{x}'} \notin \mathbf{C} \cdot \overrightarrow{\mathbf{x}} \leq \overrightarrow{\mathbf{d}},\ \overrightarrow{\mathbf{x}} \geq \overrightarrow{\mathbf{0}}$. □

Corollary 6.3. *Algorithm 1 decides if the polyhedron defined by* $\mathbf{A} \cdot \overrightarrow{\mathbf{x}} \leq \overrightarrow{\mathbf{b}},\ \overrightarrow{\mathbf{x}} \geq \overrightarrow{\mathbf{0}}$ *is contained in the polyhedron represented by* $\mathbf{C} \cdot \overrightarrow{\mathbf{x}} \leq \overrightarrow{\mathbf{d}},\ \overrightarrow{\mathbf{x}} \geq \overrightarrow{\mathbf{0}}$

Note that the linear hull inclusion problem is decidable in polynomial time, irrespective of the polyhedral system involved, that is, the polyhedral system could be completely arbitrary (and not necessarily one of 2SAT, Horn or TUM).

7. Integer Hull Inclusion

Let us restate the Integer hull inclusion problem (IHI).

Given polyhedra $P_1 : \{\mathbf{A_1} \cdot \overrightarrow{\mathbf{x}} \leq \overrightarrow{\mathbf{b_1}}\}$ *and* $P_2 : \{\mathbf{A_2} \cdot \overrightarrow{\mathbf{x}} \leq \overrightarrow{\mathbf{b_2}}\}$*, is it the case that the integer hull of* P_1 *is contained within the integer hull of* P_2*, that is, is* $S_{P_1} \subseteq S_{P_2}$*?*

We point out some relevant information.

(1) Algorithm 1 shows that the hull inclusion problem (Linear or Integer) is Turing reducible to the problem of finding the maximum (linear or integer) of a linear function over a polyhedron.

(2) It therefore follows that the IHI problem can be solved in polynomial time when $\mathbf{A_1}, \mathbf{A_2}$ are TUM. Note that when the vectors $\vec{\mathbf{b}}_1, \vec{\mathbf{b}}_2$ are integral, Algorithm 1 works as is, since the linear hull is identical to the integer hull. If the $\vec{\mathbf{b}}_\mathbf{i}$ vectors are not integral, we make the observation that the integer hull of $\mathbf{A_i} \cdot \vec{\mathbf{x}} \leq \vec{\mathbf{b}}_\mathbf{i}$ is identical to the linear hull of $\mathbf{A_i} \cdot \vec{\mathbf{x}} \leq \lfloor \vec{\mathbf{b}}_\mathbf{i} \rfloor$ when the $\mathbf{A_i}$ are TUM, and use Algorithm 1 on the modified polyhedra. For the rest of this paper, we will assume that the vectors $\vec{\mathbf{b}}_1, \vec{\mathbf{b}}_2$ are integral.

(3) Optimizing a linear function over an arbitrary 2SAT polytope is `NP-hard` since it subsumes the vertex cover problem [15, 43], and hence we cannot directly use the technique discussed in Section 6. Likewise, maximizing an arbitrary linear function over a Horn polytope is `NP-hard`, and thus the technique of Section 6 is inapplicable (see Section 7.2).

7.1. 2SAT Polytopes

Assume that $\mathbf{A_1}$ has dimension $m \times n$ and that $\mathbf{A_2}$ has dimensions $m' \times n$. Note that P_1 is constructed by taking the intersection of the m half-spaces $\vec{\mathbf{a}}_\mathbf{1}{}^\mathbf{1} \cdot \vec{\mathbf{x}} \leq b_1^1$, $\vec{\mathbf{a}}_\mathbf{1}{}^\mathbf{2} \cdot \vec{\mathbf{x}} \leq b_1^2, \ldots \vec{\mathbf{a}}_\mathbf{1}{}^\mathbf{m} \cdot \vec{\mathbf{x}} \leq b_1^m$. Likewise, P_2 is constructed by taking the intersection of the m' half-spaces, $\vec{\mathbf{a}}_\mathbf{2}{}^\mathbf{1} \cdot \vec{\mathbf{x}} \leq b_2^1$, $\vec{\mathbf{a}}_\mathbf{2}{}^\mathbf{2} \cdot \vec{\mathbf{x}} \leq b_2^2, \ldots \vec{\mathbf{a}}_\mathbf{2}{}^{\mathbf{m'}} \cdot \vec{\mathbf{x}} \leq b_2^{m'}$.

Observe that $S_{P_1} \subseteq S_{P_2}$ if and only if for all lattice points $\vec{\mathbf{x}}$,$(\vec{\mathbf{x}} \in P_1 \Rightarrow \vec{\mathbf{x}} \in P_2)$. Now, for a lattice point $\vec{\mathbf{x}}$,

$$\begin{aligned}(\vec{\mathbf{x}} \in P_1 \Longrightarrow \vec{\mathbf{x}} \in P_2) &\Longleftrightarrow \left(\vec{\mathbf{x}} \in P_1 \Longrightarrow \vec{\mathbf{x}} \in \wedge_{j=1}^{m'} \vec{\mathbf{a}}_\mathbf{2}{}^\mathbf{j} \cdot \vec{\mathbf{x}} \leq b_2^j\right) \\ &\Longleftrightarrow \left(\wedge_{j=1}^{m'}\left[\vec{\mathbf{x}} \in P_1 \Longrightarrow \vec{\mathbf{a}}_\mathbf{2}{}^\mathbf{j} \cdot \vec{\mathbf{x}} \leq b_2^j\right]\right).\end{aligned} \tag{7.1}$$

Let us focus on proving $(\vec{\mathbf{x}} \in P_1 \Rightarrow \vec{\mathbf{a}}_\mathbf{2}{}^\mathbf{j} \cdot \vec{\mathbf{x}} \leq b_2^j)$, for a specific constraint of P_2, that is, the jth half-space defining P_2. As in the case of CNF equivalence, we observe that for lattice points $\vec{\mathbf{x}}$, $(\vec{\mathbf{x}} \in P_1 \Rightarrow \vec{\mathbf{a}}_\mathbf{2}{}^\mathbf{j} \cdot \vec{\mathbf{x}} \leq b_2^j)$, if and only if the set $P_1 \wedge [\vec{\mathbf{a}}_\mathbf{2}{}^\mathbf{j} \cdot \vec{\mathbf{x}} \nleq b_2^j]$ is empty with respect to lattice points. Note that the constraint $\vec{\mathbf{a}}_\mathbf{2}{}^\mathbf{j} \cdot \vec{\mathbf{x}} \nleq b_2^j$ can be written as: $l_1 : \vec{\mathbf{a}}_\mathbf{2}{}^\mathbf{j} \cdot \vec{\mathbf{x}} > b_2^j$. But we are only interested in lattice point solutions. Consequently, a lattice point solution will satisfy the constraint l_1 if and only if it satisfies the constraint $\vec{\mathbf{a}}_\mathbf{2}{}^\mathbf{j} \cdot \vec{\mathbf{x}} \geq (b_2^j + 1)$, which is a closed half-space. Thus, we can check that $(\vec{\mathbf{x}} \in P_1 \Rightarrow \vec{\mathbf{a}}_\mathbf{2}{}^\mathbf{j} \cdot \vec{\mathbf{x}} \leq b_2^j)$ for all lattice points $\vec{\mathbf{x}}$, by checking whether the polyhedron $P_1 \wedge \vec{\mathbf{a}}_\mathbf{2}{}^\mathbf{j} \cdot \vec{\mathbf{x}} \geq (b_2^j + 1)$ is empty with respect to lattice points. The operation of negating a constraint is referred to as *constraint complementation*.

The discussion above implies the following theorem.

Theorem 7.1. *Given polyhedra P_1 and P_2, the IHI problem turing reduces to the problem of checking whether a polyhedron does not have any lattice point.*

Corollary 7.2. *The IHI problem can be solved in polynomial time for 2SAT polytopes.*

Proof. 2SAT polytopes are defined by constraints that are closed under constraint complementation, insofar as lattice points are concerned. For instance, the complement of the constraint $x_1 - x_2 \leq 4$ is $-x_1 + x_2 > -4$, which is equivalent to $-x_1 + x_2 \geq -3$ if only integral values of x_1 and x_2 are permitted. In other words, the complement of a constraint in a 2SAT polytope is also a 2SAT constraint. The presence or absence of lattice points in a 2SAT algorithm can be checked using the algorithm in [36], which runs in $O(n^3)$ time. The corollary follows. □

7.2. Horn Polytopes

Observe that Horn constraints are not closed under constraint complementation. For instance, the negation of the Horn constraint $x_1 - x_2 - x_3 \geq 5$ is $x_1 - x_2 - x_3 < 5$, which is equivalent to $x_1 - x_2 - x_3 \leq -4$ (since the x_is are integral) and hence equivalent to $-x_1 + x_2 + x_3 \geq 4$. Note that the last constraint is not a Horn constraint but an anti-Horn constraint!

We will show that the IHI problem for Horn polytopes is coNP-complete. In order to establish this result, we study a related problem, namely, PK(Horn, 1).

Given a Horn system $\mathbf{A} \cdot \overrightarrow{\mathbf{x}} \geq \overrightarrow{\mathbf{b}}$ *and a single non-Horn constraint* $\overrightarrow{\mathbf{c}} \cdot \overrightarrow{\mathbf{x}} \geq d$, *does the polyhedral system* $\mathbf{A} \cdot \overrightarrow{\mathbf{x}} \geq \overrightarrow{\mathbf{b}} \wedge \overrightarrow{\mathbf{c}} \cdot \overrightarrow{\mathbf{x}} \geq d$, *enclose a lattice point?*

We will use a reduction from the following problem.

Definition 7.3. The Hitting *Set* (HS) problem is: Given a set $S = \{s_1, s_2, \ldots, s_n\}$ called the ground set, a collection of m subsets $T_i \subseteq S$, $i = 1, 2, \ldots, m$, and a number $K \leq n$, does there exist a set $S' \subseteq S$, such that $|S'| \leq K$ and $T_i \cap S' \neq \phi,\ i = 1, 2, \ldots m$?

Theorem 7.4. *HS is* NP-complete.

The Hitting Set problem and a proof of its NP-completeness are provided in [11].

Corollary 7.5. *PK(Horn,* 1) *is* NP-complete.

Proof. PK(Horn, 1) $\in$ NP from the NP-completeness of Integer Programming [19]. We reduce the Hitting Set problem to PK(Horn, 1).

Given an instance of HS, with ground set $S = \{s_1, s_2, \ldots, s_n\}$, m sets $T_i \subseteq S$ and the target K, we construct an instance of the following variation of CNF satisfiability called MCNF. Corresponding to the ground set S, we create the literal set $L = \{x_1, x_2, \ldots, x_n\}$. Corresponding to each subset $T_i = \{s_{i_1}, s_{i_2}, \ldots, s_{i_p}\}$, we create the clause $C_i = \{x_{i_1}, x_{i_2}, \ldots, x_{i_p}\}$, where each i_k indexes the set $\{1, 2, \ldots, n\}$. The query in MCNF is as follows. Does there exist an assignment that satisfies the clause set $\phi = C_1 \wedge C_2 \wedge \cdots \wedge C_m$ such that the number of variables set to **true** is at most K? Note that all literals are positive in the MCNF instance.

Assume that the instance of HS is a "yes" instance, that is, there exists a set $S' \subseteq S$ such that $|S'| \cap T_i \neq \phi$, such that $|S'| \leq K$. Set the literals corresponding to the variables in S' to **true** and all other literals to **false**. By construction, each clause C_i is satisfied. Since the number of literals set to **true** is K, the instance of MCNF is also a "yes" instance.

Now assume that the instance of MCNF is a "yes" instance, that is, there exists a $\{$**true, false**$\}$ assignment to the literals of ϕ, such that all clauses are satisfied and the number of literals set to **true** is at most K. Construct the set S' with those elements $s_i \in S$, such that $x_i =$ **true**. By construction, S' intersects each T_i in at least one element (since each clause is satisfied), and the cardinality of S' is at most K. It follows that the instance of HS is also a "yes" instance.

We now transform the MCNF instance into an instance of PK(Horn, 1) as follows. Create n variables $y_1, y_2, \ldots, y_n$, where each $y_i \in \{0,1\}$. Corresponding to the clause $C_i = (x_{i_1}, x_{i_2}, x_{i_3})$ (say), create the Horn constraint: $(1 - y_{i_1}) + (1 - y_{i_2}) + (1 - y_{i_3}) \geq 1$. Finally, add the non-Horn constraint: $\sum_{i=1}^{n} y_i \geq n - K$. Thus, the MCNF instance is transformed into the following PK(Horn, 1) instance: $\mathbf{A} \cdot \overrightarrow{\mathbf{y}} \geq \overrightarrow{\mathbf{b}} \wedge \sum_{i=1}^{n} y_i \geq n - K$, where $\mathbf{A}$ has the Horn structure and $\overrightarrow{\mathbf{b}}$ is an integral vector.

Assume that the MCNF instance is a "yes" instance. For each literal x_i set to **true**, set y_i to 0 and for each literal x_j set to **false**, set y_j to 1. Since each clause is satisfied, it must be the case that each Horn constraint is satisfied, as per construction of the Horn constraint. Further, since the number of variables set to **true** in the MCNF instance is at most K, the number of y_is set to 0 is at most K and hence the number of y_is set to 1 is at least $n - K$, that is, we must have $\sum_{i=1}^{n} y_i \geq n - K$. Thus, the PK(Horn, 1) instance is a "yes"-instance.

Now assume that the PK(Horn, 1) instance is a "yes" instance. Set x_i to **true**, if $y_i = 0$ and x_i to **false**, if $y_i = 1$. As per the construction, each clause must be satisfied, since each Horn constraint is satisfied. Further, the number of y_is set to 1 is at least $n - K$, and hence the number of x_is set to **true** is at most K. It follows that the MCNF instance is a "yes" instance. □

We get a small bonus from this result which is useful for our next theorem.

Definition 7.6. Let HMax denote the following problem. Given a Horn polytope $\mathbf{A} \cdot \overrightarrow{\mathbf{x}} \geq \overrightarrow{\mathbf{b}}$ and a number K, do there exist $x_1, x_2, \ldots, x_n \in \{0,1\}$ such that $\mathbf{A} \cdot \overrightarrow{\mathbf{x}'} \geq \overrightarrow{\mathbf{b}}$ and $\sum_{i=1}^{n} x_i \geq K$, where $\overrightarrow{\mathbf{x}'} = [x_1, x_2, \ldots, x_n]^T$?

Corollary 7.7. *HMax is NP-complete.*

Proof. This follows from Corollary 7.2. □

We now use the `NP-completeness` of HMax to prove the following.

Theorem 7.8. *The IHI problem for Horn polytopes, that is, IH-Horn, is* co`NP-complete`.

Proof. From the proof that integer programming is in `NP`, we know that the IHI problem is in `coNP` [10].

Let $\mathbf{A_1} \cdot \overrightarrow{\mathbf{x}} \geq \overrightarrow{\mathbf{b_1}}, x_i \in \{0,1\}, i = 1, 2, \ldots, n$, and K describe an instance of HMax. We create the following instance of the IHI-Horn problem: $P_1 : \mathbf{A}_1 \cdot \overrightarrow{\mathbf{x}} \geq \overrightarrow{\mathbf{b}}_1, x_i \in \{0,1\}, i = 1, 2, \ldots, n$, and $P_2 : -\sum_{i=1}^{n} x_i \geq (1 - K)$.

We now argue that the instance of HMax is true if and only if the instance of IHI-Horn is false, that is,

$$\exists \overrightarrow{\mathbf{x}} \in \{0,1\}^n \quad \mathbf{A_1} \cdot \overrightarrow{\mathbf{x}} \geq \overrightarrow{\mathbf{b}} \wedge \sum_{i=1}^{n} x_i \geq K \Longleftrightarrow P_1 \not\subseteq P_2. \tag{7.2}$$

Note that P_1 is a Horn system, as per the hypothesis and P_2 is a Horn system by definition.

Assume that the IHI-Horn instance is true, that is, $P_1 \subseteq P_2$.

We then have for all $\overrightarrow{\mathbf{x}} : \mathbf{A_1} \cdot \overrightarrow{\mathbf{x}} \geq \overrightarrow{\mathbf{b}} \Rightarrow -\sum_{i=1}^{n} x_i \geq (1 - K)$. It follows that for all $\overrightarrow{\mathbf{x}} : \mathbf{A_1} \cdot \overrightarrow{\mathbf{x}} \geq \overrightarrow{\mathbf{b}} \Rightarrow \sum_{i=1}^{n} x_i \leq (K - 1)$. Hence, the instance of HMax is false.

Table 3: Complexities of Integer hull inclusion problems over various polyhedral families.

Polyhedral system	Hull inclusion
Diff	P
2SAT	P
TUM	P
Horn	coNP-complete

Now assume that the HMax instance is true, that is, $\exists \overrightarrow{\mathbf{x}'} \in \{0,1\}^n\ \mathbf{A_1} \cdot \overrightarrow{\mathbf{x}} \geq \overrightarrow{\mathbf{b}} \wedge \sum_{i=1}^{n} x'_i \geq K$. This implies that there exists $\overrightarrow{\mathbf{x}'} \in P_1$, with $\sum_{i=1}^{n} x'_i \geq K$, that is, $-\sum_{i=1}^{n} x'_i \leq -K$. Thus, $-\sum_{i=1}^{n} x'_i < (-K+1)$, and hence $-\sum_{i=1}^{n} x'_i < -(K-1)$. It follows that $\overrightarrow{\mathbf{x}}' \in P_1$ is not contained in P_2, and hence the instance of IHI-Horn is false.

The theorem follows. □

Table 3 summarizes the discussion on Integer hull inclusion problems.

8. Conclusion

In this paper, we discussed clausal equivalence and hull inclusion (both linear and integer) from the perspectives of a number of specialized constraint clauses. We also detailed the complexities of some satisfiability variants. Finally, we showed that the Integer hull inclusion problem for Horn polytopes is `coNP-complete`. To the best of our knowledge, our results are the first of their kind.

The work in this paper is important from the orthogonal perspectives of providing efficient strategies for special cases of hard problems and exposing interesting avenues for future research. The following interesting open problems have arisen from this work.

(i) While the satisfiability of a Horn clause system can be checked by resolution in $O(n^2)$ time, to date, the only known strategy for checking the feasibility of a general Horn polytope is linear programming [37]. Finding a simpler strategy for Horn polytope feasibility is of paramount importance since Horn polytopes find wide application in engineering domains.

(ii) Although the IHI-Horn problem is `coNP-complete`, the complexity of the problem, when the number of nonzero variables defining each constraint is a fixed constant, is unknown. A polynomial time algorithm for this problem is of enormous practical significance.

Acknowledgments

The author is extremely grateful to Judy Goldsmith for numerous discussions and to Matthew Williamson for proofreading the paper. This research has been supported in part by the Air Force Office of Scientific Research under Grant FA9550-06-0050 and in part by the National Science Foundation through Award CCF-0827397.

References

[1] Editors, Proceedings of SAT 2001, SAT 2002 and SAT 2003.

[2] S. A. Cook, “The complexity of theorem-proving procedures,” in *Proceedings of the ACM Symposium on Theory of Computing (STOC ’71)*, 1971.

[3] U. Schöning, “New algorithms for k-SAT based on the local search principle,” in *Proceedings of the Symposium on Mathematical Foundations of Computer Science (MFCS ’01)*, 2001.

[4] U. Schöning, “A probabilistic algorithm for k-SAT based on limited local search and restart,” *Algorithmica*, vol. 32, 2002.

[5] O. Kullmann, “New methods for 3-SAT decision and worst-case analysis,” *Theoretical Computer Science*, vol. 223, no. 1-2, pp. 1–72, 1999.

[6] B. Selman, H. J. Levesque, and D. G. Mitchell, “A new method for solving hard satisfiability problems,” in *Proceedings of the 10th National Conference on Artificial Intelligence (AAAI ’92)*, pp. 440–446, San Jose, Calif, USA, July 1992.

[7] W. Wei and B. Selman, “Accelerating random walks,” in *Proceedings of the 8th International Conference on Constraint Programming (CP ’02)*, Lecture Notes in Computer Science, pp. 216–232, 2002.

[8] T. J. Schaefer, “The complexity of satisfiability problems,” in *Proceedings of the 10th Annual ACM Symposium on Theory of Computing*, A. Aho, Ed., pp. 216–226, ACM Press, San Diego, Calif, USA, 1978.

[9] B. Aspvall, M. F. Plass, and R. E. Tarjan, “A linear-time algorithm for testing the truth of certain quantified Boolean formulas,” *Information Processing Letters*, vol. 8, no. 3, pp. 121–123, 1979.

[10] C. H. Papadimitriou, *Computational Complexity*, Addison-Wesley, Reading, Mass, USA, 1994.

[11] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*, W. H. Freeman, San Francisco, Calif, USA, 1979.

[12] S. A. Seshia, K. Subramani, and R. E. Bryant, “On solving boolean combinations of UTVPI constraints,” *Journal on Satisfiability, Boolean Modeling and Computation*, vol. 3, no. 1-2, pp. 67–90, 2007.

[13] S. K. Lahiri and M. Musuvathi, “An efficient decision procedure for UTVPI constraints,” in *Proceedings of the 5th International Workshop on Frontiers of Combining Systems (FroCoS ’05)*, pp. 168–183, Vienna, Austria, September 2005.

[14] S. Porschen and E. Speckenmeyer, “Satisfiability of mixed Horn formulas,” Tech. Rep., University of Cologne, Cologne, Germany, 2005.

[15] G. L. Nemhauser and L. A. Wolsey, *Integer and Combinatorial Optimization*, Wiley-Interscience Series in Discrete Mathematics and Optimization, John Wiley & Sons, New York, NY, USA, 1988.

[16] T. H. Cormen, C. E. Leiserson, and R. L. Rivest, *Introduction to Algorithms*, MIT Press, Cambridge, Mass, USA, 2nd edition, 1992.

[17] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin, *Network Flows: Theory, Algorithms, and Application*, Prentice-Hall, Englewood Cliffs, NJ, USA, 1993.

[18] M. L. Pinedo, *Scheduling: Theory, Algorithms, and Systems*, Prentice-Hall, Englewood Cliffs, NJ, USA, 1995.

[19] A. Schrijver, *Theory of Linear and Integer Programming*, John Wiley & Sons, New York, NY, USA, 1987.

[20] E. Clarke, A. Biere, R. Raimi, and Y. Zhu, “Bounded model checking using satisfiability solving,” *Formal Methods in System Design*, vol. 19, no. 1, pp. 7–34, 2001.

[21] A. Wall, K. Sandström, J. Mäki-Turja, C. Norström, and W. Yi, “Verifying temporal constraints on data in multi-rate transactions using timed automata,” in *RTCSA*, pp. 263–270, 2000.

[22] J. Møller, J. Lichtenberg, H. R. Andersen, and H. Hulgaard, “Difference decision diagrams,” in *Computer Science Logic (Madrid, 1999)*, vol. 1683, pp. 111–125, Springer, Berlin, Germany, 1999.

[23] R. Alur and D. L. Dill, “A theory of timed automata,” *Theoretical Computer Science*, vol. 126, no. 2, pp. 183–235, 1994.

[24] J. I. Rasmussen, K. G. Larsen, and K. Subramani, “Resource-optimal scheduling using priced timed automata,” in *Proceedings of the 10th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS ’04)*, K. Jensen and A. Podelski, Eds., vol. 2988 of *Lecture Notes in Computer Science*, pp. 220–235, Springer, April 2004.

[25] A. V. Goldberg, “Scaling algorithms for the shortest paths problem,” *SIAM Journal on Computing*, vol. 24, no. 3, pp. 494–504, 1995.

[26] P. Brucker, *Scheduling Algorithms*, Springer, Berlin, Germany, 2nd edition, 1998.

[27] C.-C. Han and K.-J. Lin, “Scheduling real-time computations with separation constraints,” *Information Processing Letters*, vol. 42, no. 2, pp. 61–66, 1992.

[28] N. Muscettola, B. Smith, S. Chien, et al., “In-board planning for autonomous spacecraft,” in *Proceedings of the 4th International Symposium on Artificial Intelligence, Robotics, and Automation for Space (i-SAIRAS ’97)*, July 1997.

[29] N. Muscettola, P. Morris, B. Pell, and B. Smith, “Issues in temporal reasoning for autonomous control systems,” in *Proceedings of the 2nd International Conference on Autonomous Agents*, Minneapolis, Minn, USA, 1998.

[30] B. A. Madsen, “An algorithm for exact satisfiability analysed with the number of clauses as parameter,” *Information Processing Letters*, vol. 97, no. 1, pp. 28–30, 2006.

[31] H. van Maaren and L. van Norden, “Hidden threshold phenomena for fixed-density sat formulae,” in *Proceedings of the 6th International Conference on Theory and Applications of Satisfiability Testing*, 2004.

[32] R. T. Rockafellar, *Convex Analysis*, vol. 28 of *Princeton Mathematics Series*, Princeton University Press, Princeton, NJ, USA, 1970.

[33] J. B. Hiriart-Urruty and C. Lemarechal, *Convex Analysis and Minimization Algorithms*, Springer, Berlin, Germany, 1993.

[34] M. Dyer, A. Frieze, and R. Kannan, “A random polynomial-time algorithm for approximating the volume of convex bodies,” *Journal of the Association for Computing Machinery*, vol. 38, no. 1, pp. 1–17, 1991.

[35] M. E. Dyer and A. M. Frieze, “On the complexity of computing the volume of a polyhedron,” *SIAM Journal on Computing*, vol. 17, no. 5, pp. 967–974, 1988.

[36] K. Subramani, “On identifying simple and quantified lattice points in the 2SAT polytope,” in *Proceedings of the 5th International Conference on Artificial Intelligence and Symbolic Computation (AISC ’02)*, J. Calmet, et al., Ed., vol. 2385 of *Lecture Notes in Artificial Intelligence*, pp. 217–230, Springer, July 2002.

[37] V. Chandru and J. Hooker, *Optimization Methods for Logical Inference*, Wiley-Interscience Series in Discrete Mathematics and Optimization, John Wiley & Sons, New York, NY, USA, 1999.

[38] K. Truemper, personal communication, 2003.

[39] J. H. Reif, “Symmetric complementation,” *Journal of the Association for Computing Machinery*, vol. 31, no. 2, pp. 401–421, 1984.

[40] O. Reingold, “Undirected ST-connectivity in log-space,” in *Proceedings of the 37th Annual ACM Symposium on Theory of Computing (STOC ’05)*, pp. 376–385, Baltimore, Md, USA, May 2005.

[41] M. Sipser, *Introduction to the Theory of Computation. Thompson Course Technology*, 2nd edition, 2006.

[42] P. M. VaidyaA. Aho, “An algorithm for linear programming which requires $O(((m+n)n^2+(m+n)^{1.5}n)L)$ arithmetic operations,” in *Proceedings of the 19th Annual ACM Symposium on Theory of Computing*, pp. 29–38, ACM Press, New York, NY, USA, May 1987.

[43] K. Subramani, “On deciding the non-emptiness of 2SAT polytopes with respect to first order queries,” *Mathematical Logic Quarterly*, vol. 50, no. 3, pp. 281–292, 2004.