Covers of Boolean Expressions

Vijay K. Garg and Yogish Sabharwal

IBM India Research Lab (IRL)
Plot 4, Block C, Institutional Area, Vasant Kunj
New Delhi, India
{vijgarg1,ysabharwal}@in.ibm.com

Abstract. We show that for any boolean expression, B, there exists a 2SAT expression, $cover(B)$, such that (1) all satisfying assignments of B are included in $cover(B)$, and (2) of all 2SAT expressions that satisfy (1), $cover(B)$ is the smallest. Intuitively, $cover(B)$ is a succinct representation of the set of all satisfying assignments of B extended with a minimum number of assignments to allow 2SAT representation. We give an efficient algorithm to compute $cover(B)$ for B in any class of boolean expressions that can be checked for satisfiability in polynomial time such as Horn SAT. For general boolean expressions, computing $cover(B)$ is NP-hard, but we show that an approximate cover can be computed efficiently. Using the concept of 2SAT-cover we give a sufficient condition for unsatisfiability of a boolean expression. We also consider *Horn SAT-cover* (consisting of Horn SAT clauses) of any boolean expression and give an efficient algorithm to compute approximate *Horn SAT-cover*. This algorithm enables us to derive another sufficient condition for unsatisfiability of a general CNF boolean expression.

1 Introduction

Satisfiability is a fundamental problem in computing with applications in bounded model checking, artificial intelligence, and many other disciplines. SAT solvers such as zChaff [1], and miniSAT [2] have already been used in these applications. The problem of checking satisfiability of a general boolean expression is a well-known NP-complete problem; however, when the expression belongs to specialized classes such as 2SAT or Renamable Horn SAT, it can be checked for satisfiability efficiently [3–6].

In this paper, we investigate the problem of "approximating" a general boolean expression B by a boolean expression C that belongs to a tractable class such as 2SAT or Horn SAT. The approximation provides two guarantees. First, any satisfying assignment of B is also a satisfying assignment of C. Since this constraint is easily satisfied by approximating all boolean expressions by the predicate "true," the second constraint restricts the set of satisfying assignments of C. The second constraint guarantees that if there exists any other approximation, D, which also includes all satisfying assignments of B, then C is smaller than or equal to D. We say a predicate P is smaller than or equal to predicate Q if the

set of satisfying assignments of P is contained in the set of satisfying assignments of Q. Intuitively, C is the "least" 2SAT approximation of B. We call C, 2SAT-cover, or simply cover of B, and denote it by $cover(B)$.

For example, consider the boolean expression $B = (x_1 \vee x_2 \vee x_3) \wedge (x_2 \vee x_3 \vee \bar{x_4}) \wedge (x_1 \vee \bar{x_4}) \wedge (\bar{x_1} \vee x_4)$. This expression is not in 2SAT form, but it does have a 2SAT representation. In this case, $cover(B) = (x_2 \vee x_3) \wedge (x_1 \vee \bar{x_4}) \wedge (\bar{x_1} \vee x_4)$. Now consider $B' = B \wedge (\bar{x_2} \vee x_3 \vee \bar{x_4})$. It is clear that B' is smaller than B (has fewer satisfying assignments), because $(x_1, x_2, x_3, x_4 := 0, 1, 0, 1)$ satisfies B but does not satisfy B'. However, $cover(B') = cover(B) = B$.

Intuitively, $cover(B)$ is a succinct representation of the set of all satisfying assignments of B extended with the minimum number of assignments to allow 2SAT representation. We give an efficient algorithm to compute $cover(B)$ for B in any class of boolean expressions that can be checked for satisfiability in polynomial time such as Renamable Horn SAT. Using this algorithm, we also give an efficient one-way test for unsatisfiability of a boolean expression that is a conjunction of binary clauses and Horn clauses.

For a general boolean expression, computing $cover(B)$ is NP-hard, but we show that an approximate cover can be computed efficiently. Using the concept of 2SAT-cover we give a sufficient condition for unsatisfiability of a boolean expression.

Summarizing, this paper makes the following contributions:

- We introduce the idea and theory of 2SAT-cover of any boolean expression.
- We show that the problem of computing 2SAT-cover is NP-hard.
- We present an algorithm to find the 2SAT-cover for any boolean expression that is in a tractable class. This algorithm can be used to find 2SAT-covers for predicates such as Renamable Horn SAT, or generalizations such as [7–9].
- We present an algorithm that generates an approximate 2SAT-cover for any SAT expression. This algorithm is based on converting any SAT formula into a conjunction of a 2SAT and a Horn SAT formula. This conversion may be of independent interest.
- We give an algorithm that allows one-way tests for unsatisfiability of any boolean expression in CNF form.
- We extend the theory to Horn SAT-covers. In contrast to 2SAT-covers, we show that the Horn SAT-cover for an arbitrary boolean expression may consist of an exponential number of clauses (exponential in the number of variables). This also rules out an efficient algorithm for computing Horn SAT covers. However, we show that the Horn SAT-cover for a 2SAT expression can be determined efficiently. This leads to an algorithm to generate an ap-

proximate Horn SAT-cover for any SAT expression and another algorithm for one-way unsatisfiability test of any boolean expression in CNF form.

2 Model and Notation

Let $x_1, .., x_n$ be a fixed set of n boolean variables. An *assignment* to the variables is a bit vector of size n where i^{th} bit denotes whether x_i is assigned 0 or 1. A boolean predicate is a mapping from any assignment of the variables to the set $\{0, 1\}$. The set of assignments that map to 1 are called satisfying assignments. Two extremal predicates are $false$ and *true*. The predicate $false$ always evaluates to 0 and the predicate *true* always evaluates to 1. We denote the set of satisfying assignments of a boolean predicate B by $S(B)$. The set $S(B)$ is empty for the boolean predicate $false$ and contains all 2^n assignments for the predicate *true*.

Let $\mathcal{B}$ be the set of all boolean predicates. This set has 2^{2^n} elements. We define a natural order $\leq$ on $\mathcal{B}$ as follows:

$$B_1 \leq B_2 \text{ iff } S(B_1) \subseteq S(B_2)$$

For example, the boolean expression $(x_1 \wedge x_2) \leq (x_1 \vee x_2)$ because $S(x_1 \wedge x_2) = \{11\}$, whereas $S(x_1 \vee x_2) = \{01, 10, 11\}$. It is well-known that the set $\mathcal{B}$ forms a boolean lattice with meet corresponding to conjunction and join corresponding to disjunction of boolean predicates.

Note that, we have used semantic equivalence of boolean predicates. Two boolean predicates B_1 and B_2 are considered equal iff $S(B_1) = S(B_2)$.

In this paper, we restrict ourselves to boolean expressions that are presented in conjunctive normal form (CNF). A boolean expression in CNF is simply a set of clauses, where each clause is a set of literals, and each literal is either a variable or its complement. A clause is true if at least one of its literals is true, and the boolean expression is true if all clauses are true.

The SAT problem corresponds to checking if the given boolean expression in CNF form is satisfiable. The class 2SAT consists of all SAT expressions in which each clause has at most two literals. This class of expressions can be checked (for satisfiability) in linear time (linear in the number of clauses and variables) [3]. The class Horn SAT consists of SAT expressions in which a clause has at most one positive literal and this class of expressions can also be checked in linear time. The class Renamable Horn SAT consists of SAT expressions that can be converted into Horn SAT by uniform renaming of a subset of variables [5, 4]. Generalization of these tractable classes exist such as the one defined in [8].

3 2SAT-Cover of a Boolean Function

In this section, we introduce the notion of 2SAT-cover of any general boolean expression. Note that given a fixed set of n boolean variables, there are $O(n^2)$ distinct binary clauses, and therefore at most $2^{O(n^2)}$ boolean predicates that can be expressed in 2SAT form. Since the total number of boolean predicates is $O(2^{2^n})$, it is clear that not all booean predicates are expressible in 2SAT form. Since 2SAT is a tractable class, it is natural to ask if there exists a notion of approximating a boolean predicate by 2SAT expression. This motivates the following definition.

Definition 1 (2SAT-cover). *Given any boolean expression B, 2SAT-cover(B) is a boolean expression C that is in 2SAT form and*

1. $B \leq C$
2. *For any 2SAT boolean expression D that satisfies* $B \leq D$, *we have* $C \leq D$.

We now have the following result.

Theorem 1. *Every boolean predicate, B, has a unique 2SAT-cover.*

Proof. Let $\mathcal{D}$ be the set of boolean predicates that can be expressed in 2SAT form and are greater than or equal to B. The set $\mathcal{D}$ is nonempty because it includes the predicate *true*. The predicate *true* is trivially in 2SAT form and is also greater than or equal to B. Consider the conjunction C of all boolean predicates in $\mathcal{D}$. Since each element in $\mathcal{D}$ is in 2SAT form, C is also in 2SAT form. Furthermore, $C \geq B$ because each predicate in $\mathcal{D}$ is greater than or equal to B. Moreover, if $D \in \mathcal{D}$, then $C \leq D$. Therefore, C is the 2SAT-cover of B. ■

The above proof only uses the fact that *true* is in 2SAT form and that the class of 2SAT expressions is closed under conjunction. Therefore, the above result can be generalized to any set of expressions that is closed under conjunction and includes *true*. Thus, the concept of cover is well-defined for k-SAT and Horn SAT. The proof does not hold for Renamable Horn SAT because the class of Renamable Horn SAT is not closed under conjunction.

Further, note that the proof does not give any efficient method to generate the cover. Later, we give an efficient algorithm for 2SAT-cover for any class of boolean predicates that can be checked in polynomial time.

We can view 2SAT-cover, or simply *cover*, as a function on the boolean lattice of all boolean predicate. It takes a boolean predicate f and returns the least boolean predicate expressible in 2SAT that is greater than f. The following lemma shows that the function *cover*() is a closure operator on the lattice.

Lemma 1. *Let B, B_1 and B_2 be any boolean predicates. The mapping cover() satisfies the following properties.*

1. *It is increasing, i.e.,* $\forall B : B \leq cover(B)$
2. *It is monotone, i.e.,* $\forall B_1, B_2 : B_1 \leq B_2 \Rightarrow cover(B_1) \leq cover(B_2)$
3. *It is idempotent, i.e.,* $\forall B : cover(B) = cover(cover(B))$

Proof. These properties follow easily from the standard lattice theory [10] because the set of all 2SAT expressions are closed under meet and include the top element of the boolean lattice $\mathcal{B}$. ■

It is easy to see that $cover(B) = B$ iff B is equivalent to a 2SAT expression. The next result follows from Lemma 1 and standard lattice theory [10].

Theorem 2. *The set of all boolean predicates that are equivalent to 2SAT expressions forms a lattice.*

Proof. Given any two 2SAT predicates B_1 and B_2, their meet is given by their simple conjunction. Their join is given by meet of all 2SAT expressions that are greater than or equal to both B_1 and B_2. ■

4 Algorithms for 2SAT-cover

We have earlier seen that the 2SAT-cover exists for any boolean expression B. How easy is it to compute the 2SAT-cover? Not surprisingly, the problem of computing 2SAT-cover is NP-hard as shown below.

Theorem 3. *A boolean expression B is unsatisfiable iff $cover(B) = false$.*

Proof. If B is unsatisfiable, its set of satisfying assignments is empty. The predicate $false$ is the smallest boolean predicate that contains empty set and is expressible in 2SAT form.

Conversely, if $cover(B)$ is false, B cannot have any satisfying assignment by the definition of $cover(B)$. ■

Note that given any 2SAT expression it is easy to determine whether it is satisfiable. Since $cover(B)$ is a 2SAT expression, it is easy to determine if it is unsatisfiable. Hence, it follows that computing $cover(B)$ is NP-hard because checking for satisfiability for a general boolean expression is NP-hard.

Although, computing the 2SAT-cover for general boolean expressions is hard, we show that it is efficient to compute a 2SAT-cover for any expression that belongs to a class of boolean expressions whose satisfiability can be determined in polynomial time.

The algorithm for computing 2SAT-cover is shown in Figure 1. The algorithm takes a boolean expression B as input and outputs a 2SAT expression C that is the cover of B.

The program maintains the invariant that all satisfying assignments of B are included in C. The invariant initially holds because C is initialized to true and therefore includes all satisfying assignments. We add a clause to C only if the complement of that clause does not include any satisfying assignment of B.

```
input: Boolean expression B
output: 2SAT-cover(B)
boolExpr function generate-cover(B:boolExpr)
    C := true;
    for all literals x,y do
        B' := B with x set as true and y as true;
        if B' unsatisfiable then C := C ∧ (x̄ ∨ ȳ)
    Output C;
```

Fig. 1. Algorithm to generate 2SAT-cover for boolean expression B

We now show the correctness of the algorithm.

Theorem 4. *The algorithm in Figure 1 outputs the 2SAT-cover of B.*

Proof. C is clearly a 2SAT formula. We only need to show that it includes all the satisfying assignments and there is no smaller 2SAT formula. Due to the invariant of the program, C includes all satisfying assignments of B. To show that C is the smallest 2SAT formula, let D be another 2SAT formula that is smaller than C. Assume that D includes a binary clause $(w \vee z)$ that is not in C. From program, the literals $\bar{w}$ and $\bar{z}$ must have been tried in one of the iterations of the *for* loop. It was not added to C because $\bar{w} \wedge \bar{z} \wedge B$ is satisfiable. However, this implies that D cannot have the clause $(w \vee z)$ without eliminating a satisfying assignment. Thus, every clause that is in D is also in C. Hence, $C \leq D$. ■

The algorithm for 2SAT-cover of a class $\mathcal{D}$ has time complexity $O(n^2T)$ where n is the number of variables and T is the time complexity of checking satisfiability of boolean expressions in class $\mathcal{D}$. Thus, for a Horn SAT with m clauses and n variables, the complexity is $O(n^3 + n^2m)$.

Note that the algorithm in Figure 1 generates a canonical representation of all boolean expressions on a fixed set of variables that are representable in 2SAT form. Therefore, it also allows us to compare any two 2SAT-expressions. Given two 2SAT-equivalent expressions B_1 and B_2, $B_1 \leq B_2$ iff all conjuncts in $cover(B_2)$ are included in $cover(B_1)$.

5 Horn SAT Covers

In this section, we extend the theory to Horn SAT-covers. Note that Lemma 1 and Theorems 1-3 are applicable for Horn-SAT covers as well. NP-hardness of Horn SAT-covers follows similar to 2SAT-covers. In contrast to 2SAT-covers, we show that the Horn SAT-cover for an arbitrary boolean expression may consist of number of clauses exponential in the number of variables.

Theorem 5. *There exists a Horn SAT predicate B that has exponential number of Horn SAT clauses and cannot be equivalently represented by a Horn SAT predicate having fewer number of clauses.*

Proof. Let $V = \{x_1, \cdots, x_{2m}\}$ be the set of variables. Further, let L be the set of literals corresponding to the variables being assigned *true*, i.e., $L = \{x | x \in \mathcal{V}\}$ and $\bar{L}$ be the set of literals corresponding to the variables being assigned *false*, i.e., $\bar{L} = \{\bar{x} | x \in V\}$.

Consider the predicate B over V such that an assignment $A \in S(B)$ iff fewer than m of the variables are assigned *true* in A.

This predicate is represented by the conjunction of $(2m!)/(m!m!)$ Horn SAT clauses where each clause is obtained by uniquely selecting m distinct literals from the $2m$ literals in $\bar{L}$, i.e., each clause has the form $\bar{x}_{i_1} \vee \bar{x}_{i_2} \vee \cdots \vee \bar{x}_{i_m}$, where all literals in the clause are distinct and belong to V. Each clause bars a paritcular selection of m variables from assuming the value *true* simultaneously. Clearly, these are exponential number of clauses. We prove that this predicate cannot be represented by fewer number of Horn SAT clauses. Suppose that B' is an equivalent Horn SAT predicate with fewer clauses. Then, we claim the following three properties. The detailed proof is deferred to the Appendix.

1. *Any clause C' in B' containing at least m literals from $\bar{L}$ can be reduced to a clause in B.*
   We consider any clause, C, in B that contains m of these literals; we add this clause to B' (we can do this since the two predicates are equivalent); finally we show that C' is now redundant and can therefore be removed.
2. *Any clause C' in B' containing fewer than m literals from $\bar{L}$ violates some predicate in B and therefore cannot exist.*
   Consider the assignment obtained by assigning all literals in C' to *false* and the remaining unassigned variables *false*. It can be verified that this assignment evaluated to *true* in B and *false* in B' – contradiction.
3. *No clause in B is redundant.*
   Consider any clause, C, in B. Then it can be verified that the assignment obtained by setting the variables appearing in C to *false* is only implied *false* by C and is satisfied by every other clause of B.

These three conditions together imply that there cannot exist an equivalent Horn SAT predicate containing fewer clauses. ■

However, we show that the Horn SAT-cover for a 2SAT expression can be determined efficiently.

Theorem 6. *The Horn SAT-cover for a 2SAT expression only contains clauses with at most two literals.*

Before we prove this, we give some definitions and an auxiliary lemma that will be used in the proof.

Definition 2. Impl(): *Consider a boolean 2SAT expression. For a literal y, let Impl(y) denote the set of literals that are implied* true *when only y is assigned true. These are obtained by unit propagation in the 2SAT expression.*

As an example, $Impl(y) = \{\bar{x}, w\}$ for the 2SAT expression $(\bar{x} \vee \bar{y}) \wedge (y \vee \bar{z}) \wedge (x \vee w)$.

Definition 3. Partial Assignment: *For a boolean expression, B, a set of literals, A, is said to be a partial assignment of B if there exists a satisfying assignment of B such that for every literal $y \in A$, y is true in the satisfying assignment. Intuitively, a partial assignment is an assignment on a subset of the variables that can always be extended to form a satisfying assignment by some assignment of the remaining variables.*

As an example, $A = \{\bar{x}\}$ is a partial assignment of the 2SAT expression containing the (single) clause $(x \vee y)$ since it can be extended to form a satisfying assignment $\{\bar{x}, y\}$.

Definition 4. Simplification: *Consider a Horn SAT expression B. We say that a clause C appearing in B can be simplified, if we can remove some literals from this clause to obtain a smaller clause.*

As an example, consider the Horn SAT expression formed by the conjunction of the two clauses $(\bar{x} \vee y)$ and $(\bar{x} \vee \bar{y} \vee \bar{z} \vee w)$. Then, the second clause can be simplified to $(\bar{x} \vee \bar{z} \vee w)$. The equivalence of the two expressions can be verified by performing a case analysis on the value of the variable x.

Lemma 2. *Let B be a 2SAT predicate over the variables $x_1, x_2, .., x_n$. Then*

1. *The predicate B is unsatisfiable iff there exists a variable x_i such that $x_i \in Impl(\bar{x}_i)$ and $\bar{x}_i \in Impl(x_i)$.*

2. *If B is satisfiable and there exists a literal y such that $y \in Impl(\bar{y})$, then y must be assigned true in any satisfying assignment of B.*
3. *Let A be a partial assignment of B. If there exists a literal y such that $\bar{y} \notin Impl(y)$ and $\bar{y} \notin Impl(z)$ for all $z \in A$, then $A \cup \{y\}$ is also a partial assignment of B.*

Proof. The proof follows from [3]. ∎

We now give the proof of Theorem 6.

Proof. Consider a 2SAT boolean predicate B. Let C be a Horn SAT-cover for B. If the 2SAT boolean expression is unsatisfiable, then the Horn SAT-cover is simply *false* and we are done. Suppose otherwise.

We assume that the clauses in C are neither redundant nor can they be simplified – this can be achieved by preprocessing. Now, assume that C contains a clause containing more than 2 literals. We show that this clause is either redundant or it can be simplified leading to a contradiction. Let clause C be of the form $\bar{x_1} \vee \bar{x_2} \vee \cdots \vee \bar{x_k} \vee y$ for variables $x_1, x_2, \cdots, x_k, y$ in the boolean expression. We will show how to remove some negated literal from the clause. We start with a partial assignment $A = \phi$.

The following observation relates the variables appearing in this clause. The proof of the Claim is deferred to the Appendix.

Claim 1. For any variables x_i appearing in this clause as a negated literal,

- $\{x_j, \bar{x_j}\} \cap Impl(x_i) = \phi$ for all variables $x_j (\neq x_i)$ also appearing as negated literals.
- $\{y, \bar{y}\} \cap Impl(x_i) = \phi$.

Now we can use this claim to obtain a satisfying assignment to the 2SAT expression that violates this clause implying that such a clause cannot exist resulting in a contradiction. We update the partial assignment A by adding x_i to A (i.e., setting x_i to true) for all the variables appearing as negated literals. This is made possible by the claim above combined with result 3 of Lemma 2. Now, applying result 3 of Lemma 2 again, we can update the partial assignment by adding $\bar{y}$ to $\mathcal{A}$, i.e., assigning y to false. This is the required partial assignment from which we can derive a satisfying assignment. ∎

We can now use Theorem 6 to develop an efficient algorithm that determines the Horn SAT-cover of a 2SAT expression. The algorithm is similar to the algorithm for 2SAT cover given in Figure 1. It tries out assignments for all pairs of literals and checks if the 2SAT expression is satisfiable under the assignment. If so, it adds the complement binary clause (same as in the case of 2SAT-cover) to the cover, provided it is a Horn SAT-clause.

6 Algorithms for Approximate Cover

The algorithm in Figure 1 is useful for any class of predicates for which satisfiability can be determined efficiently. What if we wanted to compute the 2SAT-cover for predicates that are not known to be checkable efficiently? In this case, although computing cover is intractable, we show methods to compute approximate covers. An approximate cover of B is a boolean expression in 2SAT form that is greater than or equal to B, i.e., we drop the condition that it be the smallest of all such expressions.

We first consider the class of CNF expressions in which each clause is either a binary clause or a Horn clause. This class clearly contains the class of 2SAT expressions as well as Horn SAT. It is natural to study this class because both 2SAT and Horn SAT are tractable. We first show that this class is NP-complete.

Theorem 7. *Let $\mathcal{F}$ be the class of CNF expressions in which each clause is either a binary clause or a Horn clause. Checking satisfiability of expressions in $\mathcal{F}$ is NP-complete.*

Proof. The problem is trivially in NP. We use reduction from 3SAT to show NP-hardness. Let B be any boolean expression in 3SAT form. We will transform it into a boolean expression D such that D has only Horn and binary clauses, and B is satisfiable iff D is.

If each clause in B is a Horn clause, we are done. Otherwise, consider any clause which has more than one positive literal. Let x be one of the positive literals. We define a new variable y such that x is equivalent to $\bar{y}$. We replace that instance of x by $\bar{y}$. We also add the following binary clauses $(x \vee y)$ and $(\bar{x} \vee \bar{y})$. By repeating this procedure we can transform B into a boolean expression D such that all clauses are either Horn or binary. It can be easily verified that B is satisfiable iff D is satisfiable. ■

We now study the problem of computing the cover (or an approximate cover) for expressions in class $\mathcal{F}$. This problem motivates computing cover of boolean expressions $B_1 \wedge B_2$ where both B_1 and B_2 belong to tractable classes. For example, B_1 and B_2 could both be Renamable Horn Clauses, or B_1 could be 2SAT and B_2 could be Renamable Horn SAT. We have the following result.

Theorem 8. *For any boolean expressions B_1 and B_2,*

$$B_1 \wedge B_2 \leq \mathit{cover}(B_1 \wedge B_2) \leq \mathit{cover}(B_1) \wedge \mathit{cover}(B_2)$$

Proof. The first inequality holds because *cover* is an increasing predicate. The second inequality holds because $\mathit{cover}(B_1) \wedge \mathit{cover}(B_2)$ is a 2SAT expression that includes all satisfying assignments of $B_1 \wedge B_2$ and $\mathit{cover}(B_1 \wedge B_2)$ is the smallest such predicate.

■

The above Theorem allows us to compute an upper bound on $cover(B_1 \wedge B_2)$. Note that whenever B_1 and B_2 are in tractable classes, we can compute $cover(B_1)$ and $cover(B_2)$ efficiently.

Combining ideas from Theorems 7 and 8, we get an algorithm for generating approximate cover for any SAT expression B. If the approximate cover is unsatisfiable, we know that B is also unsatisfiable. The algorithm is shown in Figure 2. In the first step, we convert the boolean expression into a conjunction of 2SAT and Horn SAT by introducing additional variables if necessary (one such method is illustrated in the proof of Theorem 7). In step 2, it computes the exact 2SAT-cover of the Horn SAT. By composing it with the 2SAT expression, we get an approximate cover of the original boolean expression B from Theorem 8.

We implemented a tool that computes approximate 2SAT-cover and approximate Horn SAT-cover for any SAT expression in CNF form. The covers obtained using the tool on some examples are shown below.

Example 1: Let $B = (x_1 \vee x_2 \vee x_3) \wedge (\bar{x_1} \vee \bar{x_2} \vee x_3) \wedge (x_1 \vee \bar{x_2} \vee \bar{x_3}) \wedge (\bar{x_1} \vee \bar{x_2} \vee \bar{x_3})$. B is not in Horn SAT form because the first clause is not a Horn clause. Two additional variables y_1 and y_2 are introduced such that $y_1 \equiv \bar{x_1}$ and $y_2 \equiv \bar{x_2}$. On substitution, we get $B = B_1 \wedge B_2$ where $B_1 = (y_1 \vee x_1) \wedge (y_2 \vee x_2)$ and $B_2 = (\bar{y_1} \vee \bar{x_1}) \wedge (\bar{y_2} \vee \bar{x_2}) \wedge (\bar{y_1} \vee \bar{y_2} \vee x_3) \wedge (\bar{x_1} \vee \bar{x_2} \vee x_3) \wedge (x_1 \vee \bar{x_2} \vee \bar{x_3}) \wedge (\bar{x_1} \vee \bar{x_2} \vee \bar{x_3})$. B_1 is in 2SAT form and B_2 is in Horn SAT form. The tool computes $cover(B_2)$ using the algorithm in Figure 1, which gives $cover(B_2) = (\bar{y_1} \vee \bar{x_1}) \wedge (\bar{y_2} \vee \bar{x_2}) \wedge (\bar{x_1} \vee \bar{x_2}) \wedge (\bar{x_2} \vee \bar{x_3})$. The tool determines an approximate cover of B to be $B_1 \wedge cover(B_2)$.

Example 2: Let $B = (x_1 \vee x_2) \wedge (x_1 \vee \bar{x_2}) \wedge (\bar{x_1} \vee x_2) \wedge (\bar{x_1} \vee \bar{x_2} \vee x_3) \wedge (\bar{x_1} \vee \bar{x_2} \vee \bar{x_3})$. Now B can be split into B_1 and B_2, $B_1 = (x_1 \vee x_2) \wedge (x_1 \vee \bar{x_2}) \wedge (\bar{x_1} \vee x_2)$ and $B_2 = (\bar{x_1} \vee \bar{x_2} \vee x_3) \wedge (\bar{x_1} \vee \bar{x_2} \vee \bar{x_3})$ such that B_1 is in 2SAT form, B_2 is in Horn SAT form and $B = B_1 \wedge B_2$. Now $cover(B_2) = (\bar{x_1} \vee \bar{x_2})$ and the approximate cover of B obtained is $B_1 \wedge cover(B_2) = (x_1 \vee x_2) \wedge (x_1 \vee \bar{x_2}) \wedge (\bar{x_1} \vee x_2) \wedge (\bar{x_1} \vee \bar{x_2})$ which is unsatisfiable implying that B is also unsatisfiable.

Note that Theorem 8 holds for Horn SAT-covers as well. Therefore, combining ideas from Theorems 6 and 8, we can alternatively compute the Horn SAT-cover of the 2SAT expression B_1 in step 2 of the algorithm presented in Figure 2. This results in another approximate cover (Horn SAT expression) for any SAT expression B and therefore another test for unsatisfiability.

Approximate 2SAT-cover also enables us to check satisfiability of a boolean expression B in $O(M * s)$ time where s is the size of the boolean expression and M is the number of satisfying assignments for any cover, B', of B. In the worst case, when M is large, this algorithm requires exponential time; however,

input: Boolean expression B
output: approx-2SAT-cover(B)
boolExpr **function** generate-approx-cover(B:boolExpr)
    Step 1. Convert B into $B_1 \wedge B_2$ where B_1 is in 2SAT form and B_2 is in Horn SAT form
    Step 2. C := 2SAT-cover(B_2)
    Output $B_1 \wedge C$;

Fig. 2. An efficient algorithm to generate approximate 2SAT-cover for boolean expression B

it is efficient when M is small. The algorithm uses the enumeration technique proposed by Valiant [11]. Basically, Valiant's technique substitutes *true* for any variable X and checks if the resulting boolean expression is satisfiable. If it is, the satisfying assignments of the resulting boolean expression are recursively enumerated, otherwise we are done with the assignment of x as true and the procedure is carried out with the assignment of x as false. By enumerating all satisfying assignments of B' and checking whether this assignment satisfies B, we are guaranteed to find a satisfying assignment for B if there exists one.

7 Conclusions and Future Work

We have defined the notion of 2SAT-cover for any boolean expression. This notion is useful in capturing the set of *all* satisfying assignments in a succinct manner. We have shown that computing 2SAT-cover is NP-hard for general boolean expression but an exact 2SAT-cover can be computed efficiently for any class of expressions in P and approximate 2SAT-cover can be computed for any boolean expression in CNF.

The notion of 2SAT-cover has many applications. We enumerate some below.

1. *Proving unsatisfiability of a boolean expression*: Given any boolean expression B, if its 2SAT-cover (exact or approximate) is unsatisfiable, then B is unsatisfiable.
2. *Representation of Truth Tables*: Suppose that we are required to store (or communicate) a truth table on n variables. The size of the truth table is 2^n but the space available to store or communicate the table is much smaller. By using 2SAT-cover (or more generally kSAT-cover), we can store an approximation of the boolean predicate that represents the truth table. Storing (or sending) 2SAT-cover takes only $O(n^2)$ bits as opposed to 2^n bits. It is interesting to contrast 2SAT-covers with binary decision diagrams (BDD) and their variants. BDD's represent the exact truth table, but their size may be

exponential in n in the worst case. 2SAT-covers represent an approximation of the truth table, but are guaranteed to be polynomial in size.

We have given an algorithm to compute an approximate cover for an arbitrary SAT formula. An interesting open question is to determine if there are algorithms that give better approximate covers?

Acknowledgements

We are thankful to Vikas Aggarwal for his comments on an earlier version of the work.

References

1. Moskewicz, M.W., Madigan, C.F., Zhao, Y., Zhang, L., Malik, S.: Chaff: Engineering an efficient SAT solver. In: DAC, ACM (2001) 530–535
2. Eén, N., Sörensson, N.: An extensible SAT-solver. In Giunchiglia, E., Tacchella, A., eds.: SAT. Volume 2919 of Lecture Notes in Computer Science., Springer (2003) 502–518
3. Aspvall, B., Plass, M.F., Tarjan, R.E.: A linear time algorithm for testing the truth of certain quantified Boolean formulas. Information Processing Letters **8** (1979) 121–123
4. del Val, A.: On 2-SAT and renamable horn. In: Proceedings of the 7th Conference on Artificial Intelligence (AAAI-00) and of the 12th Conference on Innovative Applications of Artificial Intelligence (IAAI-00), Menlo Park, CA, AAAI Press (2000) 279–284
5. Chandru, V., Coullard, C.R., Hammer, P.L., Montañez, M., Sun, X.: On renamable Horn and generalized Horn functions. Annals of Mathematics and Artificial Intelligence **1** (1990) 33–47
6. Lewis: Renaming a set of clauses as a horn set. JACM: Journal of the ACM **25** (1978) 134 – 135
7. Gallo, Scutella: Polynomially solvable satisfiability problems. IPL: Information Processing Letters **29** (1988)
8. Dalal, Etherington: A hierarchy of tractable satisfiability problems. IPL: Information Processing Letters **44** (1992)
9. Pretolani, D.: Hierarchies of polynomially solvable satisfiability problems. Ann. Math. Artif. Intell **17**(3-4) (1996) 339–357
10. Davey, B.A., Priestley, H.A.: Introduction to Lattices and Order. Cambridge University Press, Cambridge, UK (1990)
11. Valiant, L.G.: The complexity of enumeration and reliability problems. SIAM Journal of-Computing **8(3)** (1979) 411–421

APPENDIX

Detailed Proof of Theorem 5

Theorem 5. *There exists a Horn SAT predicate B that has exponential number of Horn SAT clauses and cannot be equivalently represented by a Horn SAT predicate having fewer number of clauses.*

Proof. Let $V = \{x_1, \cdots, x_{2m}\}$ be the set of variables. Further, let L be the set of literals corresponding to the variables being assigned *true*, i.e., $L = \{x | x \in \mathcal{V}\}$ and $\bar{L}$ be the set of literals corresponding to the variables being assigned *false*, i.e., $\bar{L} = \{\bar{x} | x \in V\}$.

Consider the predicate B over V such that an assignment $A \in S(B)$ iff fewer than m of the variables are assigned *true* in A.

This predicate is represented by the conjunction of $(2m!)/(m!m!)$ Horn SAT clauses where each clause is obtained by uniquely selecting m distinct literals from the $2m$ literals in $\bar{L}$, i.e., each clause has the form $\bar{x}_{i_1} \vee \bar{x}_{i_2} \vee \cdots \vee \bar{x}_{i_m}$, where all literals in the clause are distinct and belong to V. Each clause bars a particular selection of m variables from assuming the value *true* simultaneously. Clearly, these are exponential number of clauses. We prove that this predicate cannot be represented by fewer number of Horn SAT clauses. Suppose that B' is an equivalent Horn SAT predicate with fewer clauses. Then, we claim the following three properties.

1. Any clause in B' containing at least m literals from $\bar{L}$ can be reduced to a clause in B.
2. Any clause in B' containing fewer than m literals from $\bar{L}$ violates some predicate in B and therefore cannot exist.
3. No clause in B is redundant.

First, we show that any clause in B' containing at least m literals from $\bar{L}$ can be reduced to a clause in B. Let C' be such a clause. Let C be any clause from B containing exactly m literals from $\bar{L}$ that also appear in C' (note that such a clause always exists). Since the predicates B and B' are equivalent, we can add the clause C to B'. Now, since C' contains all the literals in C, the clause C' is weaker than C (redundant) and hence can be removed.

Next, we show that any clause C' in B' containing fewer than m literals from $\bar{L}$ violates some predicate in B and therefore cannot exist. Consider the assignment obtained by assigning each variable x to *true* if it appears in negated form in C', i.e., literal $\bar{x}$ appears in C' and *false* otherwise. In this assignment, fewer than m variables are assigned *true* and therefore by construction this evaluates

to *true* in the predicate B. However, this assignment does not satisfy the clause C' in B' and therefore evaluates to *false* in B'. This implies the two predicates are not equivalent – contradiction.

Finally, we show that none of these clauses is redundant. Let C be any clause. Then there is at least one assignment of the variables that is implied *false* only by C. Consider the assignment where a variable x_i is assigned *true* if it appears in C and *false* otherwise. Clearly, this is implied *false* by C whereas it is not implied *false* by any other clause in B. These three conditions together imply that there cannot exist an equivalent Horn SAT predicate containing fewer clauses. ■

Proof of Claim 1

Claim 1. For any variables x_i appearing in this clause as a negated literal,

- $\{x_j, \bar{x_j}\} \cap Impl(x_i) = \phi$ for all variables $x_j (\neq x_i)$ also appearing as negated literals.
- $\{y, \bar{y}\} \cap Impl(x_i) = \phi$.

Proof. Consider the following cases.

- First consider the case when $\bar{x_j} \in Impl(x_i)$. Then performing case analysis on the variable x_i, we observe that if x_i is assigned *false*, then the clause is satisfied and if on the other hand it is assigned *true*, then x_j is *false* so that the clause us again satisfied. Hence the clause is redundant and therefore we can remove it.
  Now consider the case when $x_j \in Impl(x_i)$, then we can simplify this clause to $\bar{x_1} \vee \cdots \vee \bar{x_{j-1}} \vee \bar{x_{j+1}} \vee \cdots \bar{x_k} \vee y$ (removing x_j from this clause). We need to prove that under the assumption $x_j \in Impl(x_i)$, the second clause (with x_j removed) is equivalent to the original clause. This can be shown by performing a case analysis on x_i. If x_i is assigned *false*, then both clauses are satisfied. If on the other hand, x_i is assigned *true*, then the assumption implies that x_j must also be *true* so that both the clauses reduce to the same expression.
- If $\bar{y} \in Impl(x_i)$, then either the expression is unsatisfiable or this clause will always be satisfied. Since we assume that the 2SAT expression is satisfiable to begin with, it must be that this clause is always satisfied and therefore we can eliminate this clause.
  On the other hand, if $y \in Impl(x_i)$, then we can eliminate this clause as it is always satisfied. This can be verified by performing case analysis on the value of the variable x_i. If x_i is *false*, then this clause is trivially satisfied and if x_i is *true*, then the assumption $y \in Impl(x_i)$ implies that y is *true* so that this clause is again satisfied. Hence this clause is redundant and can be removed. ■