Translating between Horn Representations and their Characteristic Models

The Harvard community has made this article openly available. Please share how this access benefits you. Your story matters

<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Citable link</td>
<td><a href="http://nrs.harvard.edu/urn-3:HUL.InstRepos:23574651">http://nrs.harvard.edu/urn-3:HUL.InstRepos:23574651</a></td>
</tr>
<tr>
<td>Terms of Use</td>
<td>This article was downloaded from Harvard University’s DASH repository, and is made available under the terms and conditions applicable to Other Posted Material, as set forth at <a href="http://nrs.harvard.edu/urn-3:HUL.InstRepos:dash.current.terms-of-use#LAA">http://nrs.harvard.edu/urn-3:HUL.InstRepos:dash.current.terms-of-use#LAA</a></td>
</tr>
</tbody>
</table>
Translating between Horn
Representations and their Characteristic
Models

Roni Khardon

TR-03-95
February 1995
(Revised May 5, 1995)

Center for Research in Computing Technology
Harvard University
Cambridge, Massachusetts
Translating between Horn Representations and their Characteristic Models

Roni Khardon
Harvard University
Cambridge, MA 02138
roni@das.harvard.edu

February, 1995
(Revised May 5, 1995)

Abstract

Characteristic models are an alternative, model based, representation for Horn expressions. It has been shown that these two representations are incomparable and each has its advantages over the other. It is therefore natural to ask what is the cost of translating, back and forth, between these representations. Interestingly, the same translation questions arise in database theory, where it has applications to the design of relational databases.

We study the complexity of these problems and prove some positive and negative results. Our main result is that the two translation problems are equivalent under polynomial reductions, and that they are equivalent to the corresponding decision problem. Namely, translating is equivalent to deciding whether a given set a models is the set of characteristic models for a given Horn expression.

We also relate these problems to translating between the CNF and DNF representations of monotone functions, a well known problem for which no polynomial time algorithm is known. It is shown that in general our translation problems are at least as hard as the latter, and in a special case they are equivalent to it.

---

*Research supported by Center for Intelligent Control Systems under ARO contract DAAL03-92-G-0115.
1 Introduction

The traditional form of representing knowledge in AI is through logical formulas [McC58, MH69], where all the logical conclusions of a given formula are assumed to be accessible to an agent. Recently, an alternative way of capturing such information has been developed [KKS93, KR94]. Instead of using a logical formula, the knowledge representation is composed of a particular subset of its models, the set of characteristic models. This set retains all the information about the formula, and is useful for various reasoning tasks. In particular, using model evaluation with the set of characteristic models, one can deduce whether another formula, a query presented to an agent, is implied by the knowledge or not. (This holds as long as the query belongs to a certain class.) While characteristic models exist for arbitrary propositional formulas, in this paper we limit our attention to logical formulas which are in Horn form and to their representation as characteristic models.

The characteristic models of Horn formulas have been shown to be useful. There is a linear time deduction algorithm using this set, and abduction can be performed in polynomial time, while using formulas it is NP-Hard [KKS93]. Furthermore, an algorithm for default reasoning using characteristic models has been developed, for cases where formula based algorithms are not known [KR95]. Hence, the question arises, whether one can efficiently translate a Horn formula into its set of characteristic models and then use this set for the reasoning task. We denote this translation problem by CCM (for Computing Characteristic Models).

On the other hand, given a set of assignments, sampled from the real world, it might be desirable to find the underlying structure behind this set of models. This is the problem of structure identification, studied in [DP92, KKS95, KPS93], which seeks an efficient translation from a set of (characteristic) models into a Horn expression that explains it. We denote this translation problem by SID (for Structure Identification).

Interestingly, the same constructs appear in the theory of relational databases. As shown in a companion paper [KMR95], there is a correspondence between Horn expressions and Functional Dependencies, and a correspondence between characteristic models and an Armstrong relation. The equivalent question of translating between functional dependencies and Armstrong relations has been studied before [BDFS84, MR86, EG91, GL90] and is relevant for the design of relational databases [MR86]. While this paper does not discuss the problems in the database domain, some of the results presented here can be alternatively derived from previous results in database theory using the above mentioned equivalence. (We identify those precisely, later on.) However, this paper makes these results more accessible without resorting to any results in database theory, and with simpler proofs. On the other hand some new results are presented, which resolve a question which was open both in AI and in the database domain.

An Example

Let us introduce the problems in question through an example. Suppose the world has 4 attributes denoted \(a, b, c, d\), each taking a value in \(\{0, 1\}\) to denote whether it is “on” or “off”, and our knowledge is given by the following constraints:

\[ W = (abc \rightarrow d)(cd \rightarrow b)(bc \rightarrow a). \]

Then \(W\) is a Horn expression and it is normally used to decide whether certain constraints are implied by it or not. For example \(W \models (cd \rightarrow a)\), and \(W \not\models (bd \rightarrow a)\), where the symbol \(\models\) stands for implication. This is normally performed by deriving a proof for the constraint in question. If
no such proof exists then implication does not hold. In our example we would notice that \((cd \rightarrow b)\), and therefore \((cd \rightarrow be \rightarrow a)\). As for \((bd \rightarrow a)\), we would fail to find a proof and therefore conclude that it is not implied by \(W\). This general approach is called theorem proving, and is efficient for Horn expressions [DG84].

An alternative approach is to check the implication relation by model checking. Implication is defined as follows: \(W \models \alpha\) if every model of \(W\) is also a model of \(\alpha\) (where \(x \in \{0,1\}^n\) is a model of an expression \(f\) if \(f\) is evaluated to “truth” on \(x\)). So to decide whether \(W \models \alpha\) we can simply use all the models of \(W\), and check, one by one, whether any of them does not satisfy \(\alpha\). In our example \(W\) has 11 models:

\[
\text{models}(W) = \{0000, 0001, 0010, 0100, 0101, 1000, 1001, 1010, 1100, 1101, 1111\}
\]

(where the assignments denote the values assigned to \(abcd\) correspondingly), and we would have to test \(\alpha\) on every one of them. Unfortunately, in general the number of models may be very large, exponential in the number of variables, and therefore this procedure will not be efficient.

The question arises therefore, whether there is a small subset of models which still guarantees correct results when used with the model checking procedure. Such a subset is called the set of characteristic models of \(W\) and its existence has been proved in [KKS93, KR94]. In our example this set is:

\[
\text{char}(W) = \{0010, 0101, 1001, 1010, 1100, 1101, 1111\}
\]

so it includes 7 out of the 11 models of \(W\). Model checking with this set is guaranteed to produce correct results for any \(\alpha\) which is a Horn expression, and using a slightly more complicated algorithm one can answer correctly for every \(\alpha\) [KKS93]. In our example, it is easy to check that \((cd \rightarrow a)\) is evaluated to “truth” on all the assignments in \(\text{char}(W)\) and that \((bd \rightarrow a)\) is falsified by 0101.

The utility of these representations, Horn expressions and characteristic models, is not comparable. Each of these representations has its advantages over the other. First, the size of these representations is incomparable. There are short Horn expressions for which the set of characteristic models is of exponential size, and vice versa, there are also exponential size Horn expressions for which the set of characteristic models is small [KKS93]. The representations also differ in the services which they support. On one hand, Horn expressions are more comprehensible. On the other hand characteristic models are advantageous in that they allow for efficient algorithms for abduction and deduction. In this paper we are asking how hard it is to translate between these representations, so as to enjoy the benefits of both.

**Overview of the Paper**

In this paper we study the complexity of the translation problems CCM and SID. For these problems, the output may be exponentially larger than the input. Therefore, it is appropriate to ask whether there are algorithms which can perform the above tasks in time which is polynomial in both the input size and the output size. These are called output polynomial algorithms.

A related problem we discuss is that of translating between the CNF and DNF representations of monotone functions, a well known problem [FK94, EG94, KPS93, Rei80] for which no polynomial algorithm is known. We call this problem the duality problem and denote it by DME (for Dualization of Monotone Expressions).

---

\(^1\)The duality problem, DME, has a lot of equivalent manifestations which appear in various branches of computer science, and is known as the Hypergraph Transversal problem [EG94] and the hitting set problem [Rei80]. A comprehensive study of these problems is given in [EG94].
We first show that the problems CCM, SID are at least as hard as DME. By that we mean that if there is an output polynomial algorithm for CCM (or for SID) then there is an output polynomial algorithm for DME. It has already been shown in [KPS93] that SID is as hard as DME, but our approach makes this exposure somewhat simpler. Further, the question whether CCM is as hard as DME, has been stated as an open problem in [KPS93] and we resolve it here. We note that these hardness results are, in some sense, not too bad since the duality problem, DME, has been shown to possess sub-exponential $n^{O(\log n)}$ time algorithm [FK94]. These hardness results can be alternatively derived by combining results from [EG94, BI93, KMR95, KR94].

Next we consider the corresponding decision problem. The problem of Characteristic Models Identification (CMI), is the problem of deciding, given a Horn expression $H$ and a set of models $G$, whether $G = \text{char}(H)$. We show that CCM, SID, and CMI are equivalent under polynomial reductions. Namely, the translation problems are solvable in polynomial time if and only if the decision problem is solvable in polynomial time. These are new results which have immediate corollaries in the database domain.

We then consider two relaxations of these translation problems. The first is considering redundant Horn expressions which contain all the Horn prime implicates for a given expression. The output of SID is therefore altered to be the set of all prime implicates, and similarly the input of CCM includes all the prime implicates instead of a minimal subset. It is shown that in this special case, SID, CCM, and DME are equivalent under polynomial reductions. Therefore, the algorithm presented by Fredman and Khachiyan [FK94] can be used to solve CCM, and SID in time $n^{O(\log n)}$. This result can be alternatively derived from the discussion in [EG91] on functional dependencies in MAK form. We show however that our argument generalizes to the larger family of $k$-quasi Horn expressions.

The second relaxation is the problem of computing all the prime implicates for a given Horn expression. This is a relaxation of CCM since using the prime implicates one can compute the characteristic models. We show that the algorithm in [FK94] can be adapted to this problem, resulting an algorithm with time complexity $n^{O(\log^2 n)}$.

It is shown, however, that both relaxations do not help in solving the general cases of CCM and SID due to exponential gaps in the size of the corresponding representations.

We also show that a related problem, which is a minor modification of CCM and SID, is Co-NP-Complete. A variant of this result, has already appeared in the database literature [GL90].

To summarize, our main result is that CCM, SID, and the corresponding decision problem CMI are equivalent under polynomial reductions. We further show that CCM, and SID are at least as hard as DME in the general case, and equivalent to DME in a special case where a Horn expression is represented by all of its prime implicates.

The rest of the paper is organized as follows. Section 2 defines characteristic models, describes some of their properties, and formally defines the problems in question. Section 3 discusses the relation to the duality problem. Section 4 discusses the relation between CCM, SID and the corresponding decision problem. Section 5 discusses representing the Horn expression using all the prime implicates, and its effect on CCM and SID. Section 6 discusses enumeration of prime implicates and its relation to CCM. Section 7 shows that a related problem is Co-NP-Hard, and Section 8 concludes with a summary.
2 Preliminaries

We consider Boolean functions $f : \{0, 1\}^n \rightarrow \{0, 1\}$. The elements in the set $\{x_1, \ldots, x_n\}$ are called variables. Assignments in $\{0, 1\}^n$ are denoted by $x, y, z$, and $\text{weight}(x)$ denotes the number of 1 bits in the assignment $x$. A literal is either a variable $x_i$ (called a positive literal) or its negation $\overline{x_i}$ (a negative literal). A clause is a disjunction of literals, and a CNF formula is a conjunction of clauses. For example $(x_1 \lor \overline{x_2}) \land (x_3 \lor \overline{x_1} \land x_4)$ is a CNF formula with two clauses. A term is a conjunction of literals, and a DNF formula is a disjunction of terms. For example $(x_1 \land x_2) \lor (x_3 \land \overline{x_1} \land x_4)$ is a DNF formula with two terms. A CNF formula is Horn if every clause in it has at most one positive literal. A formula is monotone if all the literals that appear in it are positive. The size of CNF and DNF representations is, respectively, the number of clauses and the number of terms in the representation.

An assignment $x \in \{0, 1\}^n$ satisfies $f$ if $f(x) = 1$. Such an assignment $x$ is also called a model of $f$. By “$f$ implies $g$”, denoted $f \models g$, we mean that every model of $f$ is also a model of $g$. Throughout the paper, when no confusion can arise, we identify a Boolean function $f$ with the set of its models, namely $f^{-1}(1)$. Observe that the connective “implies” ($\models$) used between Boolean functions is equivalent to the connective “subset or equal” ($\subseteq$) used for subsets of $\{0, 1\}^n$. That is, $f \models g$ if and only if $f \subseteq g$.

A term $t$ is an implicant of a function $f$, if $t \models f$. A term $t$ is a prime implicant of a function $f$, if $t$ is an implicant of $f$ and the conjunction of any proper subset of the literals in $t$ is not an implicant of $f$.

A clause $d$ is an implicate of a function $f$, if $f \models d$. A clause $d$ is a prime implicate of a function $f$, if $d$ is an implicate of $f$ and the disjunction of any proper subset of the literals in $d$ is not an implicate of $f$.

It is well known that, a minimal DNF representation of $f$ is a conjunction of some of its prime implicants. A minimal CNF representation of $f$ is a disjunction of some of its prime implicants.

If $f$ is a monotone, then it has a unique minimal DNF representation (using all the prime implicants), and a unique minimal CNF representation (using all its prime implicants).

2.1 Characteristic Models

Characteristic models were defined in [KKS93] and were also studied in [DP92, KPS93, KR94] and under a different manifestation, in database theory, [BDFS84, MR86, GL90, EG91, EG94].

For $u, v \in \{0, 1\}^n$, we define the intersection of $u$ and $v$ to be the assignment $z \in \{0, 1\}^n$ such that $z_i = 1$ if and only if $u_i = 1$ and $v_i = 1$ (i.e., the bitwise logical-and of $u$ and $v$).

For a set of assignments $S$, $x = \text{intersect}(S)$ is the assignment we get by intersecting all the assignments in $S$. We say that $S$ is redundant if there exists $x \in S$ and $S' \subseteq S$ such that $x \not\in S'$ and $x = \text{intersect}(S')$. Otherwise $S$ is non-redundant.

The closure of $S \subseteq \{0, 1\}^n$, denoted $\text{closure}(S)$, is defined as the smallest set containing $S$ that is closed under intersection.

Let $H$ be a Horn expression. The set of the Horn characteristic models of $H$, denoted here $\text{char}(H)$ is defined as the set of models of $H$ that are not the intersection of other models of $H$. Note that $\text{char}(H)$ is non-redundant. Formally,

$$\text{char}(H) = \{ u \in H \mid u \not\in \text{closure}(H \setminus \{u\}) \}. \quad (1)$$

As the following theorem shows characteristic models capture all the information about Horn expressions.
**Theorem 2.1 ([KKS93])** Let $H$ be a Horn expression then $H = \text{closure}(\text{char}(H))$.

### 2.2 Monotone Theory and Characteristic Models

The monotone theory was introduced by Bshouty [Bsh93], and was used for a theory for model-based reasoning in [KR94].

**Definition 2.1 (Order)** We denote by $\leq$ the usual partial order on the lattice $\{0,1\}^n$, the one induced by the order $0 < 1$. That is, for $x, y \in \{0,1\}^n$, $x \leq y$ if and only if $\forall i, x_i \leq y_i$. For an assignment $b \in \{0,1\}^n$ we define $x \leq_b y$ if and only if $x \oplus b \leq y \oplus b$ (Here $\oplus$ is the bitwise addition modulo 2). We say that $x > y$ if and only if $x \geq y$ and $x \neq y$.

Intuitively, if $b_i = 0$ then the order relation on the $i$th bit is the normal order; if $b_i = 1$, the order relation is reversed and we have that $1 <_b 0$. We now define:

The **monotone extension of** $z \in \{0,1\}^n$ with respect to $b$:

$$
\mathcal{M}_b(z) = \{x \mid x \geq_b z\}.
$$

The **monotone extension of** $f$ with respect to $b$:

$$
\mathcal{M}_b(f) = \{x \mid x \geq_b z, \text{ for some } z \in f\}.
$$

The set of **minimal assignments** of $f$ with respect to $b$:

$$
\text{min}_b(f) = \{z \mid z \in f, \text{ such that } \forall y \in f, z \not\geq_y y\}.
$$

Clearly, for every assignment $b \in \{0,1\}^n$, $f \subseteq \mathcal{M}_b(f)$. Moreover, if $b \not\in f$, then $b \not\in \mathcal{M}_b(f)$ (since $b$ is the smallest assignment with respect to the order $\leq_b$). Therefore:

$$
f = \bigwedge_{b \in \{0,1\}^n} \mathcal{M}_b(f) = \bigwedge_{b \not\in f} \mathcal{M}_b(f).
$$

The question is if we can find a small set of negative examples $b$, and use it to represent $f$ as above.

**Definition 2.2 (Basis)** A set $B$ is a basis for $f$ if $f = \bigwedge_{b \in B} \mathcal{M}_b(f)$. $B$ is a basis for a class of functions $\mathcal{F}$ if it is a basis for all the functions in $\mathcal{F}$.

Using this definition, we get an alternative representation for functions

$$
f = \bigwedge_{b \in B} \mathcal{M}_b(f) = \bigwedge_{b \in B} \bigvee_{z \in \text{min}_b(f)} \mathcal{M}_b(z)
$$

(2)

It is known that the size of the basis for a function $f$ is bounded by the size of its CNF representation, and that for every $b$ the size of $\text{min}_b(f)$ is bounded by the size of its DNF representation. Furthermore, if $f$ is given in its DNF representation then it is easy to compute the set $\text{min}_b(f)$, for any $b$. Each term in the DNF representation can contribute at most one assignment, where the variables that appear in the term are fixed and the others are set to their minimal value. For example $t = x_1x_3$ will contribute the assignment $1001$ to $\text{min}_{00011}(f)$. Further, it is easy make sure that the set is non-redundant by checking which of the assignments generated is in the intersection of the others.

It is known that the set $B_H = \{u \in \{0,1\}^n \mid \text{weight}(u) \geq n - 1\}$, is a basis for any Horn CNF function. For any function $f$ and set of assignments $B$ let:

$$
\Gamma_f^B = \text{min}_B(f) = \bigcup_{b \in B} \{z \in \text{min}_b(f)\}.
$$

The following theorem gives an alternative way to define $\text{char}(H)$. 

---

6
Theorem 2.2 ([KR94]) Let \( H \) be a Horn expression, then \( \text{char}(H) = \Gamma^B_H \).

As the following theorem shows the set of characteristic models can be used to answer deduction queries.

Theorem 2.3 ([KKS93, KR94]) Let \( H_1, H_2 \) be Horn expressions then \( H_1 \models H_2 \) if and only if for all \( x \in \text{char}(H_1), H_2(x) = 1 \).

For the next claim we need the following notation: A function is \( b \)-monotone iff it is monotone according to the order relation \( \leq_b \).

Claim 2.4 ([KR94]) For any \( b \)-monotone function \( f \), there is a 1-1 correspondence between the prime implicants of \( f \) and the set \( \text{min}_b(f) \). This also implies \( |\text{min}_b(f)| = |\text{DNF}(f)| \).

We would also use the notion of a least upper bound of a Boolean function [SK91], which can sometimes be characterized by the monotone theory.

Definition 2.3 (Least Upper-bound) Let \( F, G \) be classes of Boolean functions. Given \( f \in F \) we say that \( g \in G \) is a \( G \)-least upper bound of \( f \) if and only if \( f \subseteq g \) and there is no \( f' \in G \) such that \( f \subset f' \subseteq g \).

Theorem 2.5 ([KR94]) Let \( f \) be any Boolean function and \( G \) a class of all Boolean functions with basis \( B \). Then
\[
\text{f}^B_{\text{lub}} = \bigwedge_{b \in B} \text{M}_b(f)
\]
is the \( G \)-least upper bound of \( f \).

For the class of Horn expressions we have two ways to express the least upper bound. One using the monotone theory, and one using the closure operator:

Theorem 2.6 ([DP92, KKS95, KR94]) Let \( f : \{0, 1\}^n \rightarrow \{0, 1\} \) be a Boolean function. Then \( \text{f}^B_{\text{lub}} = \text{closure}(f) \), and \( \text{char}(\text{f}^B_{\text{lub}}) \subseteq f \).

2.3 The Computational Problems
This section includes definitions for all the problems discussed in this paper. Let \( H \) be a CNF expression in Horn form, and let \( \text{char}(H) \) be its set of characteristic models. The translation problems considered are:

CCM: Computing Characteristic Models
Input: a Horn CNF \( H \).
Output: the set \( \text{char}(H) \).

SID: Structure Identification (Computing Horn Expressions)
Input: a set of assignments \( \Gamma \).
Output: a Horn CNF \( H \), such that \( \Gamma = \text{char}(H) \).

DME: Dualization of Monotone Expressions
Input: a monotone CNF expression \( C \).
Output: a monotone DNF expression \( D \), such that \( C \equiv D \).
The decision problems discussed:

**DFI: Dual Form Identification**
Input: a monotone CNF expression \( C \), and a monotone DNF expression \( D \).
Output: Yes iff \( C \equiv D \).

**CMI: Characteristic Models Identification**
Input: a Horn CNF \( H \), and a set \( G \) of satisfying assignments of \( H \).
Output: Yes iff \( char(H) \subseteq G \).
Note: The condition is equivalent to \( H \models \text{closure}(G) \), and essentially also to \( G = char(H) \).

**Entailment of Closure (EOC)**
Input: a Horn CNF \( H \), a set \( G \) of assignments.
Output: Yes if and only if \( H \models \text{closure}(G) \).

We also discuss the following variant of CMI:

**Characteristic Models Identification with Counter example (CMIC)**
Input: a Horn CNF \( H \), a set \( G \) of satisfying assignments of \( H \).
Output: If \( Char(H) \subseteq G \) then output Yes. Otherwise, output No and supply a counter example \( x \in Char(H) \setminus G \).

### 2.4 Polynomial Time Algorithms and Reductions

As mentioned above we need to define algorithms that are polynomial with respect to their output. There is more than one way to give such a definition (see e.g., [EG94] for a discussion). We use the weakest of those which is called an *output polynomial algorithm*.

When the output of a problem \( P \) is uniquely defined, we say that an algorithm \( A \) is an output polynomial algorithm for \( P \) if it solves \( P \) correctly in time which is polynomial in the size of its input and output. This is the case with DME, and CCM.

When the output of a problem \( P \) is not uniquely defined, we consider the shortest permissible output \( O(I) \) for input \( I \). We say that an algorithm \( A \) is an output polynomial algorithm for \( P \) if it solves \( P \) correctly in time which is polynomial in the size of its input \( I \) and the size of \( O(I) \). We note that for SID the output is not uniquely defined since there may be redundant clauses in the Horn function.

We define polynomial reductions with respect to an oracle. A problem \( P_1 \) is *polynomially reducible* to a problem \( P_2 \) if there is an output polynomial algorithm that solves \( P_1 \) when given access to an output polynomial subroutine for \( P_2 \).

### 3 The Reductions to the Duality Problem

The duality problem is defined as computing a DNF representation for a monotone function given in its CNF form. It is easy to observe that this is equivalent to computing a CNF representation for a monotone function given in its DNF form. (We can simply exchange the \( \lor \) and \( \land \) operations to get one problem from the other). We can therefore assume that the input for DME is given as either a DNF or a CNF. Another useful observation is that renaming the variables does not change the problem. Therefore if we rename every variable as its negation (namely, replace every \( x_i \) with \( \overline{x_i} \)), we get the equivalent problem of translating between functions which are monotone with respect to the order relation \( \leq_{1} \). We call such functions *anti-monotone*. This is useful since anti-monotone functions have CNF representations in which all variables are negated, which is a
The special case of Horn expressions. Having these observations, the next two theorems follow almost immediately from the definitions, given the correspondence between minimal elements and prime implicants described in Claim 2.4. The following theorem has been proved in \cite{KPS93}; we give an alternative proof here.

**Theorem 3.1 (\cite{KPS93})** The problem DME is polynomially reducible to the problem SID.

**Proof:** Let $A$ be an algorithm for the problem SID. We construct an algorithm $B$ for the problem DME. We may assume that the input to DME is an anti-monotone DNF, $D$, and we want to compute its anti-monotone CNF representation. The algorithm $B$ computes the set $\Gamma = \min_{B_H}(D)$. This is easy since $D$ is given as a DNF expression. It then runs $A$ to produce the Horn formula $H$ such that $\text{char}(H) = \Gamma$. In case $H$ is anti-monotone, $B$ outputs $H$. Otherwise, $B$ constructs an anti-monotone expression $H'$, using $H$. Since $D$ is anti-monotone, every clause in its unique minimal CNF representation should be anti-monotone. Therefore, the set of literals in every clause in $H$, must be a superset of one of the clauses in the unique minimal representation. Therefore $B$ constructs $H'$ from $H$, by taking the negative literals from each clause in $H$.

Clearly the algorithm is polynomial. Its correctness follows from Theorem 2.2 which implies that $\Gamma = \text{char}(D)$, and Theorem 2.1 which implies that $\text{char}(D)$ uniquely identifies $D$. This implies that the input to algorithm $A$ is $\text{char}(D)$, and therefore it correctly computes a Horn expression for it, from which one can extract an anti-monotone expression.

The following result has been stated as an open problem in \cite{KPS93}.

**Theorem 3.2** The problem DME is polynomially reducible to the problem CCM.

**Proof:** Let $A$ be an algorithm for the problem CCM. We construct an algorithm $B$ for the problem DME.

We may assume that the input is an anti-monotone CNF, $C$, and we want to compute its anti-monotone DNF representation.

The algorithm $B$ runs $A$ to compute $\Gamma = \text{char}(C) = \min_{B_H}(C)$, and computes the set $\Gamma_{1n} = \{z \in \Gamma | \forall y \in \Gamma, z \not<_{1n} y\}$. Namely the elements of $\Gamma$ which are minimal with respect to the order relation $b = 1^n$. It then computes the anti-monotone DNF expression $D = \vee_{z \in \Gamma_{1n}} \wedge_{z_i = 0} \overline{x_i}$, which it outputs.

The correctness of the algorithm follows from the correspondence between minimal elements and prime implicants (Claim 2.4). As for the time complexity we observe, using Claim 2.4, that $\Gamma$ is not considerably larger than the size of the DNF. This is true since for all $b$, $|DNF(f)| = |\min_{1}(f)| \geq |\min_b(f)|$, and $|B_H| = n + 1$.

We note that both theorems can be deduced by combining results in \cite{EG94, EG91, BI93} and using the above mentioned equivalence with problems in database theory \cite{KMR95}.

We also note that it has been shown \cite{KKS93} that using the set of characteristic models one can answer abduction queries related to $H$ in polynomial time, while given the formula $H$ is NP-Hard to perform abduction \cite{SL90}. This however does not imply that computing the set of characteristic models is NP-Hard since the construction in the proof yields a Horn formula whose set of characteristic models is of exponential size.
4 Translating is Equivalent to Deciding

In this section we show that the problems CCM, SID, CMI, and CMIC are equivalent under polynomial reductions. Namely, both translation problems are solvable in polynomial time if and only if the corresponding decision problem CMI is solvable in polynomial time.

Theorem 4.1 The problems CCM, SID, CMI, and CMIC are equivalent under polynomial reductions.

Proof: The proof is established in series of lemmas. In particular we show that CMIC ≤ CMI ≤ SID ≤ CMIC, and that CMI ≤ CCM ≤ CMIC, where ≤ denotes “is polynomially reducible to”, in Lemma 4.2, Lemma 4.3, Lemma 4.5, Lemma 4.6, and Lemma 4.7 respectively.

Lemma 4.2 The problem CMIC is polynomially reducible to the problem CMI.

Proof: We get $H, G$ as input to CMIC and an algorithm $A$ to solve CMI. We run $A$ on $H, G$ as an input, and if $A$ replies yes we reply yes. Otherwise we know that there exits an $x \in \text{char}(H) \setminus G$. We need to find such a model and return it as the output of CMIC.

Consider first the easier task of finding $x \in H \setminus \text{closure}(G)$; the assignment $x$ is a witness for the fact $H \not\leq \text{closure}(G)$.

If $\text{closure}(G)$ was presented as a propositional expression we could simply substitute $x_i = \sigma \in \{0, 1\}$, into the propositional expressions, and using self reducibility solve the problem. (An iterative algorithm is used. In each step we substitute $x_i = 1$ and run $A$ again on this sub-problem. If the answer is “no” we continue to $i + 1$, otherwise we substitute $x_i = 0$ and continue to $i + 1$. The crucial point is that after substituting the sub-problem is still in the the same class, and therefore we can still use the decision algorithm for it.) However, $G$ is given as a set of models and we cannot perform this procedure.\footnote{Furthermore, the closed form we can derive using the monotone theory [KR94] does not seem to be useful.}

Observe however, that for $x_i = 1$ a similar substitution works. For $H$ we simply perform the substitution to get an expression $\tilde{H}$, and for $G$ we remove any $z \in G$ in which $z_i = 0$ to get the set $\tilde{G}$. Let $x \in \text{closure}(G)$, such that $x_i = 1$; if $x = \text{intersect}(S)$, and $y \in S$ then $y_i = 1$, and therefore $x \in \text{closure}(\tilde{G})$. Also if $x \in \text{closure}(\tilde{G})$ then $x \in \text{closure}(G)$. Therefore, if there is a witness $x$ with $x_i = 1$ then we can detect it by presenting $A$ with $\tilde{H}, \tilde{G}$ as input (on which it will say “no”).

This however does not work for $x_i = 0$. In this case an element in the closure requires at least one element in $S$ with $y_i = 0$, but we have no information on the other elements.

We circumvent this problem using the following iterative procedure. In each stage we try to turn one more variable to 1. For each $i$, we make the experiment described above of substituting $x_i = 1$. If the answer is “no”, for some $i$, we can proceed to the next stage, just as before (ignoring tests for other values of $i$). If the answer is “yes” for all $i$, then we know that for each $x_i$, that did not receive a value so far, there is no witness with $x_i = 1$, so the only possible witness is the one assigning 0 to all the variables. We return the witness $x \in \{0, 1\}^n$ arrived at, by the above substitutions, as the counter example of CMIC.

From the construction it is clear that $x \in H \setminus \text{closure}(G)$, but the requirement of CMIC is that $x \in \text{char}(H) \setminus G$. We claim that this stronger condition holds. Suppose not, and let $S \subseteq \text{char}(H)$ be such that $x = \text{intersect}(S)$. Then clearly $S$ is not a subset of $G$ or otherwise $x \in \text{closure}(G)$. Let $y \in S \setminus G$, then since $x = \text{intersect}(S)$, we get $x \prec_G y$. Namely, if $x_i = 1$ then $y_i = 1$. But this is a contradiction, since in the last run of the algorithm $A$ for CMI, it was concluded that no more variables could be set to 1, while still maintaining a witness.

$\blacksquare$
Lemma 4.3 The problem CMI is polynomially reducible to the problem SID.

Proof: Given $H, G$ as input to CMI, and an output polynomial time algorithm $A$ for SID, we run $A$ on $G$ until it stops and outputs $H'$ or until it exceeds its time bound (with respect to the size of $H$). In the first case we check whether $H = H'$ (which can be done in polynomial time [DG84]) and answer accordingly, in the second case we answer no.

The proof of the next lemma draws on previous results in computational learning theory. In this framework a function $f : \{0, 1\}^n \rightarrow \{0, 1\}$ is hidden from a learner that has to reproduce it by accessing certain “oracles”. A membership query allows the learner to ask for the value of the function on a certain point.

Definition 4.1 A membership query oracle for a function $f : \{0, 1\}^n \rightarrow \{0, 1\}$, denoted $MQ(f)$, is an oracle that when presented with $x \in \{0, 1\}^n$ returns $f(x)$.

An equivalence query allows the learner to find out whether a hypothesis he has is equivalent to $f$ or not. In case it is not equivalent the learner is supplied with a counter example.

Definition 4.2 An equivalence query oracle for a function $f : \{0, 1\}^n \rightarrow \{0, 1\}$, denoted $EQ(f)$, is an oracle that when presented with a hypothesis $h : \{0, 1\}^n \rightarrow \{0, 1\}$ returns Yes if $f \equiv h$. Otherwise it returns No and a counter example $x$ such that $f(x) \neq h(x)$.

We use a result that has been obtained in this framework.

Theorem 4.4 ([AFP92]) There is an algorithm $A$, that when given access to $MQ(f)$ and $EQ(f)$, where $f$ is a hidden Horn expression, runs in time polynomial in the number of variables and in the size of $f$, and outputs a Horn expression $H$ which is equivalent to $f$.

The hypothesis $h$, in the algorithm’s accesses to $EQ(f)$, is always a Horn expression.

The following lemma, and the simulation in its proof, are implicit in [DP92, KKS95, KM94].

Lemma 4.5 The problem SID is polynomially reducible to the problem CMIC.

Proof: We are given $G$ as input to SID, and a polynomial time algorithm $C$ for CMIC. Our algorithm will run the algorithm $A$ from Theorem 4.4 and answer the $MQ$ and $EQ$ queries that $A$ presents.

Given $x \in \{0, 1\}^n$ for $MQ$ the algorithm tests whether $x \in \text{closure}(G)$. This can be done by testing whether $x$ is equal to the intersection of all elements $y$ in $G$ such that $y \geq x$.

Given a Horn expression $h$ for $EQ$ (the theorem guarantees that the hypothesis is a Horn expression), we have to test whether $h \equiv \text{closure}(G)$. We first test whether $\text{closure}(G) \subseteq h$, which is equivalent to $\text{closure}(G) \models h$. Theorem 2.6 together with Theorem 2.3 imply that if the answer is no, then for some $x \in G$, $h(x) = 0$. Such an $x$ is a counter example for the equivalence query, and the test can be performed simply by evaluating $h$ on all the assignments in $G$.

If $\text{closure}(G) \models h$, namely all the assignments in $G$ satisfy $h$, we present $h, G$ as input to the algorithm $C$ for the problem CMIC. The input to CMIC is legal. $C$ may answer yes, meaning $\text{char}(h) \subseteq G$, which implies $h \subseteq \text{closure}(G)$. In this case we answer yes to the equivalence query. Otherwise $C$ says no and supplies a counter example $x \in \text{char}(h) \setminus G$. Since $G \subseteq h$ we get $x \in h \setminus \text{closure}(G)$ and therefore we can pass $x$ on as a counter example to the equivalence query.
We next consider the problem CCM:

**Lemma 4.6** The problem CMI is polynomially reducible to the problem CCM.

**Proof:** Given $H, G$ as input to CMI, and an output polynomial algorithm $C$ for CCM, we run $C$ on $H$ until it stops and outputs $G'$ or until it exceeds its time bound (with respect to the size of $G$). In the first case we compare $G$ and $G'$ and answer accordingly, in the second case we say no. ■

**Lemma 4.7** The problem CCM is polynomially reducible to the problem CMIC.

**Proof:** Given $H$ as input for CCM, an algorithm for CMIC can be used repeatedly to produce the elements of $\text{char}(H)$. (In fact we get an “incremental polynomial algorithm” [EG94] which is even stronger than “output polynomial” as we require here.) ■

5 **Enumerating Prime Implicates**

Having obtained the hardness results in Section 3, a natural question is whether CCM, SID are as easy as DME. This would help settle the exact complexity of the problems discussed, and more importantly would imply a sub-exponential algorithm for the problem. While no such reduction has been found (see also [EG94] for a discussion) we show here that it holds in a special case. We show, however, that the solution obtained in this way may need exponential time in the general case.

This result has already been obtained in the database domain [EG91], where restrictions of functional dependencies to be in MAK form is discussed. Our argument, however, can be generalized to richer languages, and in particular holds for the family of $k$-quasi Horn expressions defined below.

In particular we relax the problems so as to use the largest Horn expression for a function instead of using a small Horn expression. In this case the problem SID amounts to computing all the (Horn) prime implicates of the function identified by $\Gamma$. For CCM we have to compute the set of characteristic models given the set of all prime implicates rather than a small expression.

These are relaxations of the problems since, an algorithm for SID is allowed more time to compute its output, and CCM is given more information and more time for its computation.

Let $f$ be a Horn expression, then using the monotone theory representation (Equation (2)) we know that $f = \bigwedge_{b \in B_H} M_b(f)$. Recall that $B_H = \{ u \in \{0, 1\}^n \mid \text{weight}(u) \geq n - 1 \}$, and denote by $b^{(i)}$, $1 \leq i \leq n$, the assignment with $x_i$ set to zero and all other bits set to 1, and by $b^{(0)}$ the assignment $1^n$.

Let $D_i$ be the set of disjunctions that can be falsified by $b^{(i)}$, and let $G_i$ denote the language of all CNF expressions with disjunctions from $D_i$. Theorem 2.5 implies that $M_{b^{(i)}}(f)$ is equal to the least upper bound of $f$ in $G_i$. Namely, the intersection of all disjunctions in $D_i$ which are implied by $f$. Denoting by $PI(f, i)$ the set of prime implicates of $f$ with respect to $b^{(i)}$,

$$PI(f, i) = \{d \in D_i \mid f \models d \text{ and } \forall d' \subseteq d, f \not\models d' \}$$

we get:

$$M_{b^{(i)}}(f) = \bigwedge_{d \in PI(f, i)} d. \quad (3)$$

Note that the partition of the prime implicates of $f$ is not disjoint. In particular, all the anti-monotone prime implicates (except for $x_1 \lor x_2 \lor \ldots \lor x_n$ if it is a prime implicate) appear in all the $PI(f, i)$ sets. From the fact that the functions in equation (3) are $b^{(i)}$-monotone we get the connection to the duality problem DME.
Theorem 5.1 The problem SID, when the output required is all Horn prime implicates, is polynomially equivalent to DME.

Proof: First observe that the reduction in Theorem 3.1 uses an anti-monotone function, which has a unique Horn representation. Namely the smallest and the largest representations are the same in this case. This implies that the problem remains as hard as DME in this special case.

For the other direction, assume we get as input a set \( \Gamma \), and an algorithm \( A \) for DME. We first partition \( \Gamma \) into sets \( \Gamma_i \) according to minimality with respect to \( b^{(i)} \). (Note that the sets are not disjoint.) Then we use Claim 2.4 to transform each \( \Gamma_i \) into a DNF expression for the function \( \mathcal{M}_{(y^i)}(f) \). For each such DNF expression we run the procedure \( A \) (with the appropriate name changes to the variables so that the function appears monotone) to compute its CNF representation. By Equation (3), the intersection, with respect to \( i \), of these CNF expressions is the Horn expression we need.

Similarly we get for CCM:

Theorem 5.2 The problem CCM, when the input is given as the set of all Horn prime implicates, is polynomially equivalent to DME.

Proof: The proof is similar to the proof of the previous theorem. The hardness follows from Theorem 3.2.

For the other direction, we first partition the input into the sets \( PI(f, i) \), and then use a procedure for DME in order to translate each set to a DNF representation. Then using Claim 2.4 we translate the DNF expression to the set of minimal assignments. The crucial point is that we have DNF representations for the functions \( \mathcal{M}_{(y^i)}(f) \) rather than for \( f \). This implies that each term in these DNF representations is represented as an element in \( \text{char}(f) \) and therefore the reduction is polynomial. (We may get some of the elements in \( \text{char}(f) \) more than once, but at most \( n \) times, which is still polynomial.)

From the above two theorems we get the following corollary.

Corollary 5.3 The problems CCM and SID, when the Horn expression is represented as the set of all Horn prime implicates, are polynomially equivalent, and are polynomially equivalent to DME.

The equivalence of CCM and SID, in this special case, has been observed before in the database domain (Heikki Mannila, private communication). In fact this led us to the results of this section. As mentioned above a similar result for relational databases is reported in [EG91] where the restriction is called the MAK form for functional dependencies.

Lifting the Restriction: The polynomial equivalence to the problem DME, implies the existence of sub-exponential \( n^{O(\log n)} \) algorithm for these problems which may have some practical implications. However, as the following example shows one cannot apply it to solve the general case of the problem SID. Aizenstein and Pitt [AP93] present some functions with interesting properties. These functions can be manipulated to create examples with the following properties: (1) \( f \) has a short Horn expression, (2) \( |\text{char}(f)| \) is small, (3) the number of Horn “prime implicates” is exponential. In particular

\[
f = (\overline{x_1} \lor x_2 \lor \ldots \lor x_m) \land (x_1 \lor \overline{y_1}) \land (x_2 \lor \overline{y_2}) \land \ldots \land (x_m \lor \overline{y_m})
\]

has these properties. This means that arbitrary enumeration of the prime implicates, for a given set of models \( \Gamma \), is not sufficient for solving SID.
A Generalization: While we concentrate in this paper on Horn expressions, we note that the same arguments and proofs hold in the more general case of $k$-quasi Horn expressions. These are expressions in CNF form where in every disjunction there are at most $k$ positive literals (so that Horn expressions are simply 1-quasi Horn expressions). The set $B_{H_k} = \{ u \in \{0,1\}^n \mid \text{weight}(u) \geq n - k \}$ is a basis for $k$-quasi Horn expressions, and $\Gamma_f^{B_{H_k}}$ can serve as the set of characteristic models for $f$ [KR94]. The generalized versions of CCM and SID, when restricted to hold all prime implicates are still equivalent to DME.

6 Enumerating Prime Implicants

As mentioned above, given a DNF representation for $f$ we can easily compute the set of characteristic models. One might therefore try to solve CCM by first translating the Horn expression into a DNF expression and then computing the characteristic models from this set. Another possible relaxation is to first compute all the prime implicates of the function and then to extract a DNF representation from it. We consider this problem here. Namely, we consider the problem of enumerating all the prime implicates of a Horn expression, and its application for the solution of CCM.

While we have not found a general reduction from this problem to DME, the technique of [FK94] can be applied directly to get an incremental $n^{O(\log^2 n)}$ algorithm for this problem (for details see appendix). However, as we discuss below, enumeration of prime implicates of a Horn expression is not sufficient for solving CCM. The problem in such an application is an exponential gap in the sizes of these representations.

Denotes by $\#PIs(f)$ the number of prime implicates of $f$. While the representations (1) Prime Implicants (PIs), (2) DNF representation, and (3) Characteristic models, satisfy the inequalities $\#PIs(f) \geq |DNF(f)| \geq |char(f)|/n$, each of the inequalities may allow for an exponential gap. The function

$$f = (x_1 \lor x_2 \lor \cdots \lor x_{\sqrt{n}-1} \lor x_{\sqrt{n}}) \land \cdots \land (x_{n-\sqrt{n}+1} \lor x_{n-\sqrt{n}+2} \lor \cdots \lor x_{n-1} \lor x_n)$$

[KR94] shows a gap between (2) and (3). The function

$$f = x_1 x_2 \cdots x_m \lor \overline{x_1 y_1} \lor \overline{x_2 y_2} \lor \cdots \lor \overline{x_m y_m}$$

[AP93] shows a gap between (1) and (2). Both functions are Horn (for the latter by multiplying out we see that every clause for $f$ is Horn, although its Horn expression is large) and both have a small set of characteristic models. These examples show that enumeration of prime implicates may be an inefficient way for producing the characteristic models for some functions.

7 A Related Problem

In this section we show that a related problem, which is a minor variant of CCM and SID, is Co-NP-Complete. Recall the definition of EOC:

Entailment of Closure (EOC)
Input: a Horn CNF $H$, a set $G$ of assignments.
Output: Yes if and only if $H \models \text{closure}(G)$.

The important difference between CMI and EOC is that the set $G$ is not required to include only satisfying assignments of $H$. This enables the following reduction for EOC, while the complexity of CMI is still open. A similar result in the database domain has been obtained in [GL90].
Theorem 7.1 The decision problem EOC is Co-NP-Complete.

Proof: The problem is trivially in Co-NP (guess an assignment $x$ and say “no” if $x \in H \setminus \text{closure}(G)$).

To show its hardness we reduce Co-Monotone 3-SAT to EOC. Monotone 3-SAT [GJ79] is the problem of satisfiability of CNF formulas in which in every clause (has 3 literals and) either all the literals are positive (we call these clauses monotone) or all the literals are negated (we call such clauses anti-monotone). Let $f = M \land A$ an instance of Monotone 3-SAT where $M$ denotes a conjunction of monotone clauses and $A$ is a conjunction of anti-monotone clauses. We translate it to the instance of EOC: $H = A$ and $\Gamma = \bigcup_{b \in B} \min_b(\overline{M})$. First we claim that the reduction is polynomial. Note that since $M$ is a monotone CNF, $\overline{M}$ is a DNF formula in which all the variables are negated, and can therefore be written as an anti-monotone CNF formula. This implies that $\overline{M}$ is Horn, but we have it in a DNF representation. Further computing $\Gamma$ is easy given the DNF representation of $\overline{M}$, and its size is bounded by $(n+1)$ times the number of clauses in $M$.

We now claim that $f$ is satisfiable if and only if $H \not\subseteq \text{closure}(\Gamma)$. Assume first that $f$ is satisfiable, and let $x \in A \land M$. This implies that $x \in H$ and $x \not\in \overline{M}$. Since $\overline{M}$ is Horn, and the models of Horn functions are closed under intersection (Theorem 2.1) we get that $x \not\in \text{closure}(\overline{M})$, and since $\Gamma \subseteq \overline{M}$, $x \not\in \text{closure}(\Gamma)$. Therefore, $H \not\subseteq \text{closure}(\Gamma)$.

For the other direction assume $H \not\subseteq \text{closure}(\Gamma)$, and let $x$ be an assignment such that $x \in H$ and $x \not\in \text{closure}(\Gamma)$. We get that $x \in A$, and since by Theorem 2.1 and Theorem 2.2 $\overline{M} = \text{closure}(\Gamma)$ we have $x \not\in \overline{M}$. So, $x \in A \land M$ and $f$ is satisfiable. \hfill \qed

8 Conclusions

Horn expressions and characteristic models are two alternative representations for the same information and none of the two dominates the other in the computational services it can support. The same representations occur in database theory where they have a role in the design of relational databases.

We have studied the complexity of these problems and proved some positive and negative results. Our main result is that the two translation problems CCM, and SID, are equivalent to each other (under polynomial reductions), and that they are equivalent to the corresponding decision problem CMI. Namely, translating in either direction is equivalent to deciding whether a given set a models is the set of characteristic models for a given Horn expression.

We have also shown that in general CCM, and SID are at least as hard as the duality problem DME, and that in a special case CCM,SID, and DME are equivalent.

Some of the results presented in this paper can be obtained from previous results in database theory, using the equivalence between Armstrong relations and characteristic models reported in a companion paper [KMR95]. However, our proofs and exposition make these results much more accessible.

The main open problems which remain are therefore to determine the exact complexity of CMI, and that of DME. While DME has a sub-exponential algorithm, the problems CMI might still be co-NP-Hard.

Acknowledgments

I am grateful to Thomas Eiter, Heikki Mannila, and Dan Roth for very useful comments on earlier drafts of this paper.
A Enumerating Prime Implicants

The technique of [FK94] can be adapted to enumerate the prime implicants of Horn expressions (in fact, it works for any self-reducible class of CNF expression for which we can solve satisfiability). The same arguments in the first part of their analysis go through, with the main differences being the recursive step, which is only slightly more complicated, and the sampling argument below. We briefly sketch the arguments here.

Let $H$ be a Horn expression, $D$ a DNF expression, and suppose we want to test whether they are equivalent.

1. If $H$ is equivalent to $D$ then every clause in $H$ must have a literal in common with every term in $D$ (since otherwise we can satisfy $D$ and falsify $H$ at the same time).
2. Let $m$ be the size of $H$ plus the size of $D$. Either there is a short ($\leq \log 2m$) clause in $H$, or a short term in $D$, or deciding whether $H \equiv D$ can be easily solved by sampling (this is done deterministically in [FK94]).

This is true since in such a case the formula $C \land \overline{D}$ is a CNF expression with $m$ clauses and each falsifies at most $2^n - \log 2m$ assignments. Therefore the formula falsifies at most $m \cdot 2^n - \log 2m = 2^n - 1$ assignments, and a single random sample has a probability $1/2$ of hitting a positive assignment.

3. If there is a short clause (term) then there is a literal which appears in $D$ ($H$) with “high frequency. Namely, with frequency $1/\text{length}$.

4. We can recurse on this literal, replacing either 0 or 1 to it and creating two subproblems, such that the recursion takes $n^{O(\log^2 n)}$. The recursion stops if either $C$ or $D$ have only one element.

In such a case we can find a counter example by solving satisfiability for the (other) formula.

We now illustrate the recursive step. Suppose the literal with high frequency is $x_1$, and that it appears in $D$ with frequency $\epsilon$, and appears at least once in $H$. The expressions $H$ and $D$ can be described as $D = x_1 D_0 + \overline{x_1} D_1 + D_2$, and $H = (x_1 + H_0)(\overline{x_1} + H_1)(H_2)$, where $D_0, D_1, D_2, H_0, H_1, H_2$ do not include the variable $x_1$. Substituting $x_1 = 0$ we need to test whether $D_1 + D_2 = H_0 H_2$.

The size of the sub-problem we get is: $|\text{DNF}| \leq (1 - \epsilon)|D|$ and $|\text{Horn}| \leq |H|$. Substituting $x_1 = 1$ we need to test $D_0 + D_2 = H_1 H_2$. The size of the sub-problem we get is: $|\text{DNF}| \leq |D|$ and $|\text{Horn}| \leq |H| - 1$. This is the same recursion as in [FK94] which is shown to be bounded by $n^{O(\log^2 n)}$.

5. The above arguments can be combined to yield a procedure which either answers yes or provides a counter example (similar to CMIC discussed above) which satisfies $H$ but not $D$. Using this counter example $x$ it is easy to get a new prime implicant of $H$. This can be done by a greedy search, starting with the term that satisfies only $x$ and deleting literals if possible. Since $t \not\models H$ iff $t \land \overline{H}$ is not satisfiable, this test requires solving satisfiability for a DNF expression which is trivial.

6. There is one last caveat to consider: once the set of prime implicants (PIs) enumerated constitute a representation for the function, there may still be PIs not enumerated but we cannot find any more counter examples. Enumeration of these PIs can be performed by taking “consensus” operations [AP93] on the existing PIs and testing whether a new PI is created. One can show that this yields a polynomial incremental algorithm.

References


The more complicated recursion that achieves time $n^{O(\log n)}$ uses the fact that the form of the functions is simpler and seems to be non-applicable here.


