On Computing Minimal Models*

Rachel Ben-Eliyahu
Cognitive Systems Laboratory
Computer Science Department
University of California
Los Angeles, California 90024
rachel@cs.ucla.edu

Rina Dechter
Information & Computer Science
University of California
Irvine, California 92717
dechter@ics.uci.edu

Abstract

This paper addresses the problem of computing the minimal models of a given CNF propositional theory. We present two groups of algorithms. Algorithms in the first group are efficient when the theory is almost Horn, that is, when there are few non-Horn clauses and/or when the set of all literals that appear positive in any non-Horn clause is small. Algorithms in the other group are efficient when the theory can be represented as an acyclic network of low-arity relations. Our algorithms suggest several characterizations of tractable subsets for the problem of finding minimal models.

1 Introduction

One approach to attacking NP-hard problems is to identify *islands of tractability* in the problem domain and to use their associated algorithms as building blocks for solving hard instances, often approximately. A celebrated example of this approach is the treatment of the propositional satisfiability problem.

In this paper, we would like to initiate a similar effort for the problem of finding one, all, or some of the *minimal models* of a propositional theory. Computing minimal models is an essential task in many reasoning systems in Artificial Intelligence, including propositional circumscription [Lif] and minimal diagnosis [dKMR92], and in answering queries posed on logic programs (under stable model semantics [GL91, BNNS91]) and deductive databases (under the generalized closed-world assumption [Min82]). While the ultimate goal in these systems is not to compute minimal models but rather to produce plausible inferences, efficient algorithms for computing minimal models can substantially speed up inference in these systems.

*This work was partially supported by an IBM graduate fellowship to the first author, by NSF grants IRI-9157636 and IRI-9200918, by Air Force Office of Scientific Research grant AFOSR 900136, and by a grant from Xerox Palo Alto research center.

Special cases of this problem have been studied in the diagnosis literature and, more recently, the logic programming literature. Algorithms used in many diagnosis systems [dKW87, dKMR92] are highly complex in the worst case: To find a minimal diagnosis, they first compute all prime implicates of a theory and then find a minimal cover of the prime implicates. The first task is output exponential, while the second is NP-hard. Therefore, in the diagnosis literature, researchers have often compromised completeness by using a heuristic approach. The work in the logic programming literature (e.g. [BNNS91]) focused on using efficient optimization techniques, such as linear programming, for computing minimal models. A limitation of this approach is that it does not address the issue of worst-case and average-case complexities.

We want to complement these approaches by studying the task of finding all or some of the minimal models in general, independent of any specific domain. We will use the "tractable islands" methodology to provide more refined worst-case guarantees. The two primary "islands" that we use are *Horn theories* and *acyclic theories*. It is known that Horn theories have a unique minimal model that can be found in linear time [DG84]. Our near-Horn algorithms try to associate an input theory with a "close" Horn theory, yielding algorithms whose complexity is a function of this "distance". For acyclic theories, we will show that while finding one or a subset of the minimal models can be done in output-polynomial time, the task of finding all minimal models is more complex. We will set up conditions under which the set of all minimal models can be computed in output-polynomial time and we will present a tree-algorithm that solves this problem in general. Once we have an efficient algorithm for generating minimal models of tree-like theories, we can apply it to any arbitrary theory by first compiling the theory into a tree. The resulting complexity will often be dominated by the complexity of this compilation process and will be less demanding for "near-tree" theories.

2 Preliminary definitions

A clause is *positive* if it contains only positive literals and is *negative* if it contains only negative literals. In this paper, a *theory* is a set of clauses. A set of literals *covers* a theory iff it contains at least one literal from each clause in the theory. A set of covers of a theory is *complete* iff it is a superset of all minimal covers of the theory.

A theory is called *positive* if it is composed of positive clauses only. Given a theory Φ and a set of literals S, the operation $\Phi \oslash S$ performs unit propagation on the theory $\Phi \bigcup S$. For each theory Φ, $nf(\Phi)$ denotes $\Phi \oslash \emptyset$. For each model M, $pos(M)$ denotes the set of symbols to which M assigns **true**. We will sometimes refer to a model as a set of literals, where a negative literal $\neg P$ in the model means that the model assigns **false** to P and a positive literal P in the model means that the model assigns **true** to P.

Definition 2.1 (X-minimal model) *Let Φ be a theory over a set of symbols $\mathcal{L}$, $X \subseteq \mathcal{L}$, and M a model for Φ. M is an* X-minimal *model for Φ iff there is no other model M' for Φ such that $pos(M') \bigcap X \subset pos(M) \bigcap X$. If M is an X-minimal model for $X = \mathcal{L}$, it will be called simply a* minimal model.

3 General algorithms

Cadoli [Cad92] has shown that the problem of finding an X-minimal model for a theory is $P^{NP[O(\log n)]}$-hard. Roughly, this means that it is at least as hard as problems that can be solved by a deterministic polynomial algorithm that uses $O(\log n)$ calls to an NP oracle. In Figure 1 we show an algorithm for computing X-minimal models that takes $O(n^2)$ steps and uses $O(n)$ calls to an NP oracle (where n is the number of variables in the theory). In Figure 2 we show a variation of this algorithm that uses a procedure for satisfiability that also returns a model in case the theory is satisfiable. The algorithm suggests the following:

Theorem 3.1 *Let C be a class of theories over a language $\mathcal{L}$ having the following properties:*

1. *There is an algorithm α such that for any theory $\Phi \in C$, α both decides whether Φ is satisfiable and produces a model for Φ (if there is one) in time $O(t_C)$.*
2. *C is closed under instantiation, that is, for every $\Phi \in C$ and for every literal L in $\mathcal{L}$, $\Phi \oslash \{L\} \in C$.*

Then for any theory $\Phi \in C$, an X-minimal model for Φ can be found in time $O(|X| t_C)$.

Corollary 3.2 *An X-minimal model for a 2-CNF theory Φ can be found in time $O(|X|n)$ where n is the length of the theory.*

However, using a straightforward reduction from VERTEX COVER [Kar72], we can show that if we are interested in finding a minimum cardinality

Find-X-minimal(Φ, X, M)
Input: A theory Φ and a subset of the variables in Φ, X. **Output:** **true** if Φ is satisfiable, **false** otherwise. In case Φ is satisfiable, the output variable M is an X-minimal model for Φ.

1. If $\neg$sat(Φ) return **false**;
2. For $i = 1$ to n $M[i] =$ **false**;
3. Let $P_1, ..., P_n$ be an ordering on the variables in Φ such that the first $|X|$ variables are all the variables from X.
4. For $i := 1$ to n do
   If sat($\Phi \bigcup \{\neg P_i\}$) then $\Phi := \Phi \oslash \{\neg P_i\}$
   else $\Phi := \Phi \oslash \{P_i\}$, $M[i] =$ **true**;
5. return **true**;

Figure 1: Algorithm Find-X-minimal

Find-X-minimal2(Φ, X, M)

1. If $\neg$model-sat(Φ, M) return **false**;
2. $negX := \{P | P \in X, \neg P \in M\}$; $X := X - negX$; $\Phi := \Phi \bigcup \{\neg P | P \in negX\}$;
3. While $X \neq \emptyset$ do
   a. Let $P \in X$;
   b. If $\neg$model-sat($\Phi \bigcup \{\neg P\}, M'$) then $\Phi := \Phi \oslash \{P\}$ else $\Phi := \Phi \oslash \{\neg P\}$, $M := M'$;
   c. $X := X - \{P\}$; If $X = \emptyset$ return **true**;

Figure 2: Algorithm Find-X-minimal2

model for a 2-CNF theory (namely, a model that assigns **true** to a minimum number of symbols), the situation is not so bright:

Theorem 3.3 *The following decision problem is NP-complete: Given a* positive *2-CNF theory Φ and an integer K, does Φ have a model of cardinality $\leq K$?*

4 Algorithms for almost-Horn theories

In this section, we present algorithms for computing minimal models of a propositional theory which are efficient for almost Horn theories. The basic idea is to instantiate as few variables as possible so that the remaining theory will be a Horn theory and then find a minimal model for the remaining theory in linear time.

4.1 Algorithm for theories with only a few non-Horn clauses

Algorithm MinSAT is efficient when most of the theory is Horn and there are only few non-Horn clauses. Given a theory, MinSAT works as follows: It first tries to solve satisfiability by unit propagation. If the empty clause was not generated and no positive clause is left, the theory is satisfiable, and the unique minimal model assigns **false** to the vari-

MinSAT(Φ, M)

Input: A theory Φ. **Output:** **true** if Φ is satisfiable, **false** otherwise. In case Φ is satisfiable, the output variable M will contain a set of models for Φ that is a superset of *all* the *minimal* models of Φ.

1. Φ :=UnitInst(Φ, I, Sat); If not Sat return **false**;
2. If Φ contains no positive clauses then *begin* $M := \{I \bigcup \{\neg P | P \in \Phi\}\}$; return **true**; *end.*
3. $M := \emptyset$; Let A be a complete set of covers for the set of all the positive clauses in Φ.
   For each $S \in A$ do:
   If MinSAT($\Phi \bigcup S, M'$) then
   $M := M \bigcup$ combine(I, M');
4. If $M == \emptyset$ then return **false** else return **true**;

Figure 3: Algorithm MinSAT

ables in the remaining theory. If a nonempty set of positive clauses is left, we compute a cover for the remaining set of positive clauses, replace them with the cover, and then call MinSAT recursively on the new theory. If the theory is not satisfiable, or if we are interested in *all* minimal models, we have to call MinSAT again with a different cover.

Algorithm MinSAT is shown in Figure 3. The procedure $UnitInst(\Phi, I, Sat)$ gets a theory Φ and returns $nf(\Phi)$. I contains the set of unit clauses used for the instantiations. Sat is false iff the empty clause belongs to the normal form; otherwise Sat is true. The procedure $combine(I, M)$ gets a set of literals I and a set of sets of literals M and returns the set $\{S | S = W \bigcup I, W \in M\}$.

We can show that MinSAT returns a superset of all the minimal models of the theory. We group all the propositional theories in classes $\Psi_0, \Psi_1, \ldots$ as follows:

- $\Phi \in \Psi_0$ iff $nf(\Phi)$ has no positive clauses or contains the empty clause.
- $\Phi \in \Psi_{k+|C|}$ iff for some A that is a complete set of covers for C and for each S in A, $\Phi \oslash S$ belongs to Ψ_k, where C is the set of positive clauses in $nf(\Phi)$.

Note that if a theory has k non-Horn clauses it belongs to the class Ψ_j for some $j \leq k$ and that all Horn theories belong to Ψ_0. We can show that if $\Phi \in \Psi_k$ then the above algorithm runs in time $O(nm^k)$, where n is the length of the input and m the maximum number of positive literals that appear in any clause. This is also the worst case complexity if we are interested only in deciding satisfiability. Since for every k the class Ψ_k is closed under instantiation, we can use Theorem 3.1 to prove that:

Proposition 4.1 *If a theory Φ belongs to the class Ψ_k for some k, then an X-minimal model for Φ can be found in time $O(|X|nm^k)$.*

Algorithm MinSAT returns a superset of all the minimal models. To identify the set of all minimal models, we need to compare all the models generated. Therefore, the complexity of finding all minimal models for a theory in the class Ψ_k is $O(nm^{2k})$.

4.2 Algorithms that exploit the positive graph of a theory

In this section we will identify tractable subsets for *satisfiability* and for finding all minimal models by using topological analysis of what we call the *positive graph* of a theory. The positive graph reflects on the interactions of the positive literals in the theory.

Definition 4.2 (positive graph of a theory) *Let Φ be a theory. The* positive graph *of Φ is an undirected graph (V, E) defined as follows:*

V= $\{P | P$ *is a positive literal in some clause in* $\Phi\}$,
E= $\{(P, Q) |$ *P and Q appear positive in the same clause*$\}$.

Note that Φ is a Horn theory iff its positive graph has no edges.

Definition 4.3 (vertex cover) *Let $G = (V, E)$ be a graph. A* vertex cover *of G is a set $V' \subseteq V$ such that for each $e \in E$ there is some $v \in V'$ such that $v \in e$.*

We take "vertex cover of the theory" to mean "vertex cover of the positive graph of the theory".

An algorithm that computes a superset of all minimal models based on a vertex cover of a theory can consider all possible instantiations of the variables in the cover. Each such instantiation yields a Horn theory for which we can find a minimal model (if there is one) in linear time. When we combine the model for the Horn theory with the cover instantiation, a model of the original theory results. We can show that a superset of all minimal models of a theory can be generated in this way. If we are interested only in deciding satisfiability, we can stop once the first model is found. Hence,

Theorem 4.4 *If the positive graph of a theory Φ has a vertex cover of cardinality c, then the satisfiability of Φ can be decided in time $O(n2^c)$, where n is the size of the theory, and an X-minimal model for Φ can be found in time $O(|X|n2^c)$. The set of all minimal models of Φ can be found in time $O(n2^{2c})$.*

In general, the problem of finding a minimum-cardinality vertex cover of a graph is NP-hard. A greedy heuristic procedure for finding a vertex cover could simply remove the node with maximum degree from the graph and continue with the reduced graph until all nodes are disconnected. The set of all nodes removed is a vertex cover.

Algorithm VC-minSAT (Figure 4) integrates the above heuristic into a backtrack algorithm for finding the minimal models. $MaxDegree$ takes the positive graph as an input and returns a symbol (node) that has a maximum degree. If there is more

VC-minSAT(Φ, M, G)

Input: A theory Φ and a positive graph of Φ, G.

Output: **true** if Φ is satisfiable, otherwise **false**. If Φ is satisfiable, M contains a superset of all minimal models for Φ.

1. $I := \emptyset$; $\Phi := UnitInst(\Phi, I, Sat)$;
2. If $\neg Sat$ return **false**; $G := Update(\Phi, G)$;
3. If G is disconnected then
   begin $M := I \bigcup \{\neg P | P \in \Phi\}$; return **true**; *end.*
4. $P := MaxDegree(G)$; $Sat :=$ **false**; $M = \emptyset$;
5. If VC-minSAT($\Phi \bigcup \{P\}, M^+, G$) then
   $M := combine(I, M^+)$;
6. If VC-minSAT($\Phi \bigcup \{\neg P\}, M^-, G$) then
   $M := M \bigcup combine(I, M^-)$;
7. If $M == \emptyset$ return **false** else return **true**

Figure 4: Algorithm VC-minSAT

than one such symbol, it chooses the one that appears in a maximum number of non-Horn clauses in the theory. $Update(\Phi, G)$ returns the positive graph of Φ, by updating G. We can show that algorithm VC-minSAT produces a superset of all the minimal models.

We should mention here that the idea of initializing variables in a theory until the remaining theory is Horn has been suggested, in the context of solving the satisfiability problem, by Gallo and Scutella [GS88] and was recently extended by Dalal and Etherington [DE92]. The advantages of our approach are that we provide an intuitive criteria for how the variables to be instantiated are selected and we classify the performance of the algorithm using a well-understood and largely explored graphical property, vertex cover.

Also note that we could define the *negative graph* of a theory just as we defined the positive graph. We could then write an algorithm that is analogous to VC-minSAT and is efficient for deciding satisfiability of theories for which the negative graph has a small vertex cover. Clearly, algorithm minSAT also has an analogous algorithm that considers negative instead of positive clauses.

5 Computing minimal models on acyclic networks of relations

In this section we provide efficient algorithms for theories that can be represented as acyclic relations of low arity. We next define the notions of *constraint networks* and *relations* and show how they can represent propositional theories and their satisfying models.

Definition 5.1 (relations, networks, schemes) *Given a set of variables $X = \{X_1, ..., X_n\}$, each associated with a domain of discrete values $D_1, ..., D_n$, respectively, a* relation *(or, alternatively, a* constraint*) $\rho = \rho(X_1, ..., X_n)$ is any subset*

$$\rho \subseteq D_1 \times D_2 \times ... \times D_n.$$

The projection *of ρ onto a subset of variables R, denoted $\Pi_R(\rho)$ or ρ_R, is the set of tuples defined on the variables in R that can be extended to a tuple in ρ. A* constraint network *N over X is a set $\rho_1, \ldots, \rho_t$ of such relations. Each relation ρ_i is defined on a subset of variables $S_i \subseteq X$. We also denote by $\rho(S_i)$ the relation specified over S_i. The set of subsets $S = \{S_1, .., S_t\}$ is called the* scheme *of N. The network N represents a unique relation $rel(N)$ defined over X, which stands for all consistent assignments (or all solutions), namely,*

$$rel(N) = \{x = (x_1, ..., x_n) | \ \forall S_i \in S, \Pi_{S_i}(x) \in \rho_i\}.$$

A partial assignment $T = t$ is a value assignment to a subset of variables $T \subseteq X$. The operator $\bowtie$ is the join operator in relational databases. If $rel(N) = \rho$, we say that N describes *or* represents *ρ.*

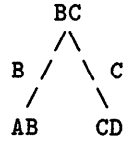
Any propositional theory can be viewed as a special kind of constraint network, where the domain of each variable is $\{0, 1\}$ (corresponding to **{false, true}**) and where each clause specifies a constraint (in other words, a relation) on its propositional symbols. The *scheme of a theory* is accordingly defined as the scheme of its corresponding constraint network, and the set of all models of the theory corresponds exactly to the set of all solutions of its corresponding constraint network.

Example 5.2 *Consider the theory $\Phi = \{\neg A \vee \neg B, \neg B \vee \neg C, C \vee D\}$. This theory can be viewed as a constraint network over the variables $\{A, B, C, D\}$, where the corresponding relations are the truth tables of each clause, that is, $\rho(AB) = \{00, 01, 10\}$, $\rho(BC) = \{00, 01, 10\}$, and $\rho(CD) = \{01, 10, 11\}$. The scheme of the theory Φ is $\{AB, BC, CD\}$. The set of all solutions to this network (and hence the set of models of Φ) is*

$$\rho(ABCD) = \{0001, 0010, 0011, 0101, 1001, 1010, 1011\}.$$

Note that Φ has two minimal models: $\{0001, 0010\}$.

The scheme of a theory can be associated with a *constraint graph* where each relation in the scheme is a node in the graph and two nodes are connected iff the corresponding relations have variables in common. The arcs are labeled by the common variables. For example, the constraint graph of the theory Φ of Example 5.2 is as follows:



Theories that correspond to a constraint graph that is a tree are called *acyclic theories*, and their corresponding tree-like constraint graph is called a *join tree*.

We next present two algorithms for computing minimal models for *acyclic* theories. These algorithms will be extended to arbitrary theories via a procedure known as *tree-clustering* [DP89], which compiles any theory into a tree of relations. Consequently, given a general theory, the algorithms presented next work in two steps: A join-tree is computed by tree-clustering, and then a specialized tree-algorithm for computing the minimal models is applied. The complexity of tree-clustering is exponential in the size of the maximal arity of the generated relations, and hence our algorithms are efficient for theories that can be compiled into networks of low-arity relations. We should note, however, that even in the cases where tree-clustering is expensive, it might still be useful since it offers a systematic way of representing the models of the theory in a hierarchical structure capable of supporting information retrieval without backtracking.

5.1 Finding a subset of all minimal models

For the rest of Section 5, we will assume that we are dealing with constraint networks that correspond to propositional theories, and hence the domain of each variable is $\{0,1\}$ and we have the ordering $1 \succ 0$. We will also assume that we are looking for models that are minimal over all the symbols in the language of the theory, namely, X-minimal models where X is the set of all symbols in the theory.

Definition 5.3 *Given a relation ρ defined on a set of variables X, and given two tuples r and t in ρ, we say that $t \succ r$, iff for some X_0 in X, $t_{X_0} \succ r_{X_0}$ and, for all $X_i \in X$, $t_{X_i} \succ r_{X_i}$ or $t_{X_i} = r_{X_i}$. We say that t and r agree on a subset of variables $S \subseteq X$ iff $r_S = t_S$.*

Definition 5.4 (conditional minimal models) *Given a relation ρ over X and a subset of variables $S \subseteq X$, a tuple $t \in \rho$ is* conditionally minimal *w.r.t. S iff $\nexists\ r \in \rho$ such that r agrees with t on S and $t_{X-S} \succ r_{X-S}$. The set of all conditional minimal models (tuples) of ρ w.r.t. $S = s$ is denoted $min(\rho \setminus S = s)$. The set of all conditional minimal models (tuples) of ρ w.r.t. S is denoted $min(\rho \setminus S)$ and is defined as the union over all possible assignments s to S of $min(\rho \setminus S = s)$. $min(\rho \setminus \emptyset)$ is abbreviated to $min(\rho)$.*

Example 5.5 *Consider the relation*

$$\rho(ABCD) = \{0111, 1011, 1010, 0101, 0001\}.$$

In this case, we have $min(\rho) = \{1010, 0001\}$, $min(\rho \setminus \{C,D\}) = \{0111, 1011, 1010, 0001\}$, and $min(\rho \setminus \{A\}) = \{0001, 1010\}$.

One can verify that: (1) any minimal tuple of a projection $\Pi_S(\rho)$ can be extended to a minimal tuple of ρ, but not vice versa; (2) a conditionally minimal tuple is not necessarily a minimal tuple; and (3) a minimal tuple is a conditional minimal tuple w.r.t. to all subsets.

Next we show that, given a join-tree, a subset of all minimal models can be computed in output polynomial time. The idea is as follows: Once we have a rooted join-tree (which is pair-wise consistent[1]), we can take all minimal tuples in the root node and extend them (via the join operation) with the matching conditional minimal tuples in their child nodes. This can be continued until we reach the leaves. It can be shown that all the models computed in this way are minimal and that they are generated in a backtrack-free manner; however, not *all* the minimal models will be generated. In order to enlarge the set of minimal models captured, we can reapply the procedure where each node serves as a root. We can show that if every minimal model has a projection that is minimal in at least one relation of the tree, the algorithm will generate all the minimal models. Formally,

Definition 5.6 (parents of S) *Given a scheme $S = \{S_1, ..., S_t\}$ of a rooted join-tree, we associate each subset S_i with its parent subset $S_{p(i)}$ in the rooted tree. We call an ordering $d = S_1, .., S_t$ a* tree-ordering *iff a parent node always precedes its child nodes.*

Definition 5.7 *Let T be a rooted join-tree with S_0 at the root. Let ρ_i be the relation associated with S_i and let $d = S_0, S_1, ..., S_t$ be a tree-ordering. We define*

$$\rho^0(T) = min(\rho_0) \bowtie_{i=1..t} (min(\rho_i \setminus S_{p(i)})).$$

Theorem 5.8 *Let T be a rooted join-tree with a tree-ordering $\{S_1, ..., S_t\}$. Then $\rho^0(T)$ is a subset of all the minimal models of T, and $\rho^0(T)$ can be computed in $O(L \sum_{i=1}^{t} |\rho_i|)$ steps where L is the number of minimal models in the output and ρ_i is the input relation associated with S_i.*

Example 5.9 *Consider the join-tree that corresponds to the theory Φ in Example 5.2. Assuming BC is the root, we can use the tree-ordering $d = BC, AB, CD$. Since tuple $(BC = 00)$ is the only minimal model of $\rho(BC)$, it is selected. This tuple can be extended by $A = 0$ and by $D = 1$, resulting in one minimal model of ρ, namely the tuple $(ABCD = 0001)$. If AB plays the role of a root, we will still be computing the same minimal model. However, when CD plays the role of a root, we will*

[1] Pair-wise consistency, or arc consistency, is a process that when applied to join-trees will delete from the join-tree all the tuples that do not belong to any solution. Pair-wise consistency can be achieved in polynomial time.

min1(Φ)
Input: A theory Φ.
Output: A subset of all the minimal models of Φ.

1. Apply tree-clustering to Φ. If the theory is not satisfiable, stop and exit. Else, generate join-tree T. Apply pair-wise consistency to T.
2. For each node R in T do: For each join tree T' rooted at R compute $\rho^0(T')$.
3. Output the union of all models computed.

Figure 5: Algorithm min1

compute the tuple $(ABCD = 0010)$, *which is also a minimal model of* ρ.

From Theorem 5.8, it follows that, given an acyclic network or any general backtrack-free network relative to an ordering d, one minimal model can be computed in time that is linear in the size of the network and the total subset of minimal models $\rho^0(T)$ can be computed in time proportional to the size of the set. We summarize this in algorithm *min1*, given in Figure 5.

Theorem 5.10 (complexity of min1) *The complexity of* **min1** *is* $O(n2^k + nL|\rho|)$, *where* k *is the maximum arity of any relation in the join-tree,* n *is the number of relations,* ρ *is the largest relation in the generated tree* T, *and* L *is the number of minimal models generated.*

So **min1** is especially efficient when the theory is compiled into a join-tree having relations with low arity. We next present two sufficient conditions for the completeness of algorithm **min1** .

Theorem 5.11 (sufficient condition) *Suppose* T *is a join-tree having the scheme* $S = \{S_1, ..., S_t\}$, *and suppose that for every minimal model* t *of* T *there is a scheme* $S_i \in S$ *such that* $\Pi_{S_i}(t)$ *is in* $min(\rho(S_i))$. *Then* **min1** , *when applied to* T, *will generate all the minimal models of* T.

Theorem 5.12 (local sufficient condition) *Suppose that for every node* S *in a join-tree* T *the set* $min(\rho(S) \setminus S')$ *is totally ordered, where* S' *is the set of all variables that are common to* S *and at least one of its neighbors in the tree. Then* **min1** , *when applied to* T, *will generate all the minimal models.*

5.2 Listing all minimal models

Algorithm *min1* does not necessarily produce all minimal models because, as the following example shows, it is not always the case that all minimal models are minimal within at least one subrelation.

Example 5.13 *Consider the join-tree where the variables are* $\{A, B, C, D, E, F, G\}$, *the scheme is a tree* $\{ABC, BCDEF, EFG\}$, *and the corresponding relations are* $\rho(ABC) = \{011, 110, 000\}$, $\rho(BCDEF) = \{11011, 10100, 00010\}$, *and* $\rho(EFG) = \{110, 000, 101\}$. *The reader can verify that the tuple* $\{0110110\}$ *is a minimal model for this network, but its projection relative to any of the relations is not minimal.*

min2(T)
Input: A pair-wise consistent join tree T which corresponds to a theory Φ.
Output: All minimal models of ϕ.

1. Traverse the tree bottom up and compute $Pmin(R)$ for each node R visited using equations (1) and (2).
2. Output $Pmin(R^0)$, where R^0 is the root node.

Figure 6: Algorithm min2

We now present a second algorithm, **min2** , that computes all the minimal models but is not as efficient as **min1** in the sense that during processing it may generate models of the theory that are not minimal. Some of those models will be pruned only at the final stage. Nevertheless, we conjecture that the algorithm is optimal for trees.

Basically, algorithm **min2** computes partial minimal models recursively while traversing the join-tree bottom up. When we visit a node R, we prune all the partial models that we already know cannot be extended to a minimal model. The resulting subset of partial models is denoted by $Pmin(R)$. More formally, let T_R denote the network rooted at node R and S_R the set of all variables that R shares with its parent. We define

$$Pmin(R) = min(rel(T_R) \setminus S_R).$$

Since $S_R \subseteq R$,

$$Pmin(R) = min(J_R \setminus S_R) \qquad (1)$$

where J_R is defined to be

$$J_R = min(rel(T_R) \setminus R).$$

Note that for the root node R^0, $Pmin(R^0)$ is the set of all minimal models of the whole tree (conditioning is on the empty set). We can show that J_R can be expressed recursively as a function of $Pmin(U_1), ..., Pmin(U_n)$ where $U_1, ..., U_n$ are R's children:

$$J_R = \rho(R) \bowtie (\bowtie_{i=1}^{n} Pmin(U_i)). \qquad (2)$$

This allows a bottom-up computation of $Pmin(R)$ starting at the leaf nodes. The algorithm is summarized in Figure 6.

Example 5.14 *Consider again the tree-network of Example 5.9. Algorithm* **min2** *will perform the following computations:*

$Pmin(AB) = min(\rho(AB) \setminus \{B\}) = \{00, 01\}$,
$Pmin(CD) = min(\rho(CD) \setminus \{C\}) = \{01, 10\}$,

$$Pmin(BC) = min(\rho(BC) \bowtie (Pmin(AB) \bowtie Pmin(CD))) = min(\rho(BC) \bowtie (ABCD = \{0001, 0010, 0101, 0110\})) = min(\{0001, 0010, 0101\}) = \{0010, 0001\}.$$

We see that although the theory has 7 models, only 4 intermediate models were generated during the computation.

The reader can also verify that algorithm **min2** produces all the minimal models of Example 5.13. We can show that **min2** computes all and only the minimal models. The complexity of *min2* (without the tree-clustering preprocessing step) can be bounded as follows:

Theorem 5.15 *Let r be the maximum number of tuples in any relation ρ_i in the join-tree, and suppose that for every node R in the join-tree $|J_R| \leq m$. Then the complexity of* **min2** *is $O(nm^2)$, where n is the number of relations.*

Consequently, if the ratio between the number of minimal models, l, and $|J_R|$ is less than some c for every node R in the tree, then the number of models generated will be linear in $c \cdot l$.

6 Conclusion

The task of finding all or some of the minimal models of a theory is at the heart of many knowledge representation systems. This paper focuses on this task and introduces several characterizations of tractable subsets for this problem.

We have presented new algorithms for finding minimal models of a propositional theory. The first group of algorithms is effective for almost-Horn theories. The other group is effective for theories that can be represented as an acyclic network of small-arity relations.

Loveland and colleagues (e.g. [Lov91]) have shown how *SLD* resolution for first-order Horn theories can be modified to be efficient for near-Horn theories. We use different methods and provide worst-case complexities. Cadoli [Cad92] has described a partition of the set of propositional theories into classes for which the problem of finding one minimal model is tractable or NP-hard. His classification is different from ours but, as in Section 5, is also done by considering the set of logical relations that correspond to to the theory. An algorithm that exploits acyclic theories for computing minimum cardinality models is given in [FD92].

The ultimate usefulness of our algorithms must be tested by implementing them in systems that solve real-world problems in diagnosis or logic programming. We believe, however, that in any event the algorithms and the theoretical bounds provided in this paper are of value since the problem of computing minimal models is so fundamental.

Acknowledgments

We thank Yousri El Fattah, Itay Meiri, and Judea Pearl for useful discussions and helpful comments on earlier drafts of this paper. We have benefited from discussions with Adam Grove and Daphne Koller on the topic of computing minimal models. Thanks also to Michelle Bonnice for editing.

References

[BNNS91] C. Bell, A. Nerode, R.T. Ng, and V.S. Subrahmanian. Computation and implementation of non-monotonic deductive databases. Technical Report CS-TR-2801, University of Maryland, 1991.

[Cad92] Marco Cadoli. On the complexity of model finding for nonmonotonic propositional logics. In *Proceedings of the Fourth Italian Conference on Theoretical Computer Science*, October 1992.

[DE92] M. Dalal and D. Etherington. A hierarchy of tractable satisfiability problems. *IPL*, 44:173–180, 1992.

[DG84] W. Dowling and J. Gallier. Linear time algorithms for testing the satisfiability of propositional horn formulae. *journal of Logic Programming*, 3:267–284, 1984.

[dKMR92] J. de Kleer, A.K. Mackworth, and R. Reiter. Characterizing diagnosis and systems. *Artificial Intelligence*, 56:197–222, 1992.

[dKW87] J. de Kleer and B.C. Williams. Diagnosis multiple faults. *Artificial Intelligence*, 32:97–130, 1987.

[DP89] R. Dechter and J. Pearl. Tree clustering for constraint networks. *Artificial Intelligence*, 38:353–366, 1989.

[FD92] Y. El Fattah and R. Dechter. Empirical evaluation of diagnosis as optimization in constraint networks. In *DX-92: Proceedings of the workshop on Principles of Diagnosis*, October 1992.

[GL91] Michael Gelfond and Vladimir Lifschitz. Classical negation in logic programs and disjunctive databases. *New Generation Computing*, 9:365–385, 1991.

[GS88] G. Gallo and M. Scutella. Polynomially solvable satisfiability problems. *IPL*, 29:221–227, 1988.

[Kar72] R. M. Karp. Reducibility among combinatorial problems. In *Complexity of Computer Computations*. Plenum Press, 1972.

[Lif] V. Lifshitz. Computing cricumscription. In *IJCAI 1985.*

[Lov91] D. Loveland. Near-horn prolog and beyond. *Journal of Automated Reasoning*, 7:1–26, 1991.

[Min82] J. Minker. On indefinite databases and the closed world assumption. In *Proceedings of the 6th Conference on Automated Deduction*. Springer-Verlag, 1982.