Computing minimal models, stable models and answer sets

Zbigniew Lonc[1] and Mirosław Truszczyński[2]

[1] Faculty of Mathematics and Information Science, Warsaw University of Technology, 00-661 Warsaw, Poland

[2] Department of Computer Science, University of Kentucky, Lexington, KY 40506-0046, USA

Abstract. We propose and study algorithms for computing minimal models, stable models and answer sets of 2- and 3-CNF theories, and normal and disjunctive 2- and 3-programs. We are especially interested in algorithms with *non-trivial worst-case performance bounds*. We show that one can find all minimal models of 2-CNF theories and all answer sets of disjunctive 2-programs in time $O(m1.4422..^n)$, generalizing a similar result we obtained earlier for computing stable models of normal 2-programs (n is the number of atoms in an input theory or program and m is its size). Our main results concern computing stable models of normal 3-programs, minimal models of 3-CNF theories and answer sets of disjunctive 3-programs. We design algorithms that run in time $O(m1.6701..^n)$, in the case of the first problem, and in time $O(mn^2 2.2720..^n)$, in the case of the latter two. All these bounds improve by exponential factors the best algorithms known previously. We also obtain closely related upper bounds on the number of minimal models, stable models and answer sets a 2- or 3-theory or program may have.

1 Introduction

Our goal in this paper is to propose and study algorithms for computing *minimal* models of CNF theories, *stable* models of normal logic programs and *answer sets* of disjunctive logic programs. We are especially interested in algorithms for which we can derive *non-trivial worst-case performance bounds*. Our work builds on studies of algorithms to compute models of propositional CNF theories [Kul99] and improves on our earlier study of algorithms to compute stable models [LT03].

Propositional logic with the minimal-model semantics (propositional circumscription) [McC80,Lif88], logic programming with stable-model semantics [GL88] and disjunctive logic programming with the answer-set semantics [GL91] are among most commonly studied and broadly used knowledge representation formalisms (we refer the reader to [MT93,BDK97] for a detailed treatment of these logics and additional pointers to literature). Recently, they have been receiving much attention due to their applications in *answer-set programming* (ASP) — an emerging declarative programming paradigm. To solve a problem in ASP, programmers write a logic theory or a logic program so that minimal models, stable models and answer sets of the theory or program, respectively, are in one-to-one correspondence to problem solutions [MT99,Nie99,Bar03]. Fast algorithms to compute minimal models, stable models and answer sets are essential for the computational effectiveness of propositional circumscription, logic programming and disjunctive logic programming as answer-set programming systems.

These computational tasks can be solved by a "brute-force" straightforward search. We will now briefly describe this approach. Here and throughout the paper, by $At(T)$ or $At(P)$ we denote the set of atoms occurring in a theory T or a program P. We represent models of propositional theories, stable models of normal logic programs and answer sets of disjunctive logic programs as sets of atoms that are true in these models and answer sets. Finally, we use n to denote the number of atoms of an input theory or program, and m to denote its *size*, that is, the total number of atom occurrences in the theory or program.

We can compute all minimal models of a CNF theory T in time $O(m3^n)$. Indeed, to check whether a set $M \subseteq At(T)$ is a minimal model of T, it is enough to verify that M is a model of T and that none of its subsets is. Each such test can be accomplished in $O(m)$ steps. Thus, if $|M| = i$, we can verify whether M is a minimal model of T in time $O(m2^i)$. Checking all sets of cardinality i requires $O(\binom{n}{i}m2^n)$ steps and checking all sets — $O(\sum_{i=0}^{n}\binom{n}{i}m2^n) = O(m3^n)$.

Next, we note that to determine whether a set of atoms M is an answer set of a disjunctive logic program P we need to verify whether M is a minimal model of the *reduct of P with respect to M* [GL91] or, equivalently, whether M is a minimal model of the propositional theory that corresponds to the reduct. Thus, a similar argument as before demonstrates that also answer sets of a finite propositional disjunctive logic program P can be computed in time $O(m3^n)$.

In the case of stable models we can do better. The task of verifying whether a set of atoms M is a stable model of a finite propositional logic program P can be accomplished in time $O(m)$. Consequently, one can compute all stable models of P, using an exhaustive search through all subsets of $At(P)$, in $O(m2^n)$ steps.

We refer to these straightforward algorithms and to the corresponding worst-case performance bounds of $O(m3^n)$ and $O(m2^n)$ as *trivial*. A fundamental question, and the main topic of interest in this paper, is whether there are algorithms for the three computational problems discussed here with better worst-case performance bounds.

We note that researchers proposed several algorithms to compute minimal models of propositional CNF theories [BEP94,Nie96], stable models of logic programs [SNS02] and answer sets of disjunctive logic programs [EFLP00]. Some implementations based on these algorithms, for instance, *smodels* [SNS02] and *dlv* [EFLP00] perform very well in practice. However, so far very little is known about the worst-case performance of these implementations.

In this paper, we study the three computational problems discussed earlier. We focus our considerations on *t-CNF theories* and *t-programs*, that is, theories and programs, respectively, consisting of clauses containing no more than t literals. Despite restricted syntax, t-CNF theories and t-programs are of significant interest and appear frequently as encodings of problems in planning, model checking, computer-aided verification, circuit design and combinatorics. To encode a problem by means of a propositional (disjunctive) logic program we often proceed in a two-step process. We first develop an encoding of this problem as a program in the syntax of (disjunctive) DATALOG with negation [MT99,Nie99,EFLP00]. In the next step, we *ground* this program and obtain its propositional counterpart also representing the problem in question. It is clear that programs obtained by grounding finite (disjunctive) DATALOG-with-negation programs are t-programs, for some fixed and usually small t that depends only on the problem specification and not on a particular problem instance. In fact, in most cases programs obtained by grounding and some subsequent straightforward simplifications become (disjunctive) 2- or 3-programs.

In our earlier work, we studied the problem of computing stable models of normal t-programs [LT03]. We presented an algorithm to compute all stable models of a normal 2-program in time $O(m3^{n/3}) = O(m1.4422..^n)$ and showed its asymptotic optimality. We proposed a similar algorithm for the class of normal 3-programs and proved that its running time is $O(m1.7071..^n)$.

In this paper, we improve on our results from [LT03] and extend them to the problems of computing minimal models of CNF theories and answer sets of disjunctive logic programs. Since 2- and 3-CNF theories and 2- and 3-programs appear frequently in applications and since our results are strongest in the case of these two classes, we limit our discussion here to 2- and 3-theories and programs only.

Our results are as follows. We show that one can find all minimal models of 2-CNF theories and all answer sets of disjunctive 2-programs in time $O(m1.4422..^n)$, generalizing a similar result we obtained earlier for computing stable models of normal 2-programs. Our main results concern computing stable models of normal 3-programs, minimal models of 3-CNF theories and answer sets of disjunctive 3-programs. We design algorithms that run in time $O(m1.6701..^n)$, in the case of the first problem, and in time $O(mn^2 2.2720..^n)$, in the case of the latter two. These bounds improve by exponential factors the best algorithms known previously. We also obtain closely related upper bounds on the number of minimal models, stable models and answer sets a 2- or 3-theory or program may have.

2 Lower bounds

In this section, we present lower bounds on the running time of algorithms for computing *all* minimal models, stable models or answer sets. We obtain them by constructing theories and programs for which the total size of output is large, and use this quantity to derive a lower bound on the exponential factor in our running-time estimates. Due to space limitations, we present here only the results, leaving out details of the constructions. Let us define $\mu_t = \binom{2t-1}{t}^{1/2t-1}$. We have the following theorem and its straightforward corollary.

Theorem 1. *Let t be an integer, $t \geq 2$. There are positive constants d_t, D_t and D'_t such that for every $n \geq 2t-1$ there is a t-CNF theory T (normal t-program P and disjunctive t-program P, respectively) with n atoms and such that*

1. *The size m of T (P, respectively) satisfies $m \leq d_t n$*
2. *The number of minimal models of T (stable models of P or answer sets of P, respectively) is at least $D_t \mu_t^n$ and the sum of their cardinalities is at least $D'_t n \mu_t^n$.*

Corollary 1. *Let t be an integer, $t \geq 2$.*

1. *Every algorithm computing all minimal models of t-CNF theories (stable models of normal t-programs, answer sets of disjunctive t-programs, respectively) requires in the worst case at least $\Omega(n\mu_t^n)$ steps*
2. *Let $0 < \alpha < \mu_t$. There is no polynomial f and no algorithm for computing all minimal models of t-CNF theories (stable models of normal t-programs, answer sets of disjunctive t-programs, respectively) with worst-case performance of $O(f(m)\alpha^n)$.*

For $t = 2$ and 3, the lower bound given by Corollary 1 specializes to $\Omega(n1.4422..^n)$ and $\Omega(n1.5848..^n)$, respectively.

3 Main results

We will now present and discuss main results of our paper. Even when we do not mention explicitly an input theory or program, we follow our convention and write n for its number of atoms and m for its size.

We start by stating two theorems that deal with minimal models of 2-CNF theories and answer sets of disjunctive 2-programs. They extend the results we obtained in [LT03] for the case of stable models of normal 2-programs.

Theorem 2. *There are algorithms to compute all minimal models of 2-CNF theories, stable models of normal 2-programs and answer sets of disjunctive 2-programs, respectively, that run in time $O(m3^{n/3}) = O(m1.4422..^n)$.*

Theorem 3. *Every 2-CNF theory (every normal 2-program and every disjunctive 2-program, respectively) has at most $3^{n/3} = 1.4422..^n$ minimal models (stable models, answer-sets, respectively).*

It follows from Theorem 1 and Corollary 1 that these results are asymptotically optimal.

Next, we present results concerning the case of 3-CNF theories and normal and disjunctive 3-programs. These results constitute the main contribution of our paper. First, we derive upper estimates on the number of minimal models, stable models and answer sets. In fact, we obtain the same upper bound in all three cases. It improves the bound on the number of stable models of normal 3-programs, which we derived in [LT03]. In the same time, it is the first non-trivial bound on the number of minimal models of 3-CNF theories and answer sets of disjunctive programs.

Theorem 4. *Every 3-CNF theory T (every normal 3-program P and every disjunctive 3-program P, respectively) has at most* $1.6701..^n$ *minimal models (stable models, answer-sets, respectively).*

For 2-CNF theories and 2-programs, the common bound on the number of minimal models, stable models and answer sets, appeared also as an exponential factor in formulas estimating the running time of algorithms to compute the corresponding objects. In contrast, we find that there is a difference in how fast we can compute stable models of normal 3-programs as opposed to minimal models of 3-CNF theories and answer sets of disjunctive 3-programs. The reason is that the problem to check whether a set of atoms is a stable model of a normal program is in P, while the problems of deciding whether a set of atoms is a minimal model of a 3-CNF theory or an answer set of a disjunctive 3-program are co-NP complete [CL94,EG95]. Our next result concerns computing stable models of normal 3-programs. It gives an exponential improvement on the corresponding result from [LT03].

Theorem 5. *There is an algorithm to compute stable models of normal 3-programs that runs in time $O(m1.6701..^n)$.*

Since checking whether a set of atoms is a minimal model of a 3-CNF theory or an answer set of a disjunctive 3-program is NP-hard, our results concerning computing minimal models and answer sets are weaker. Nevertheless, in each case they provide an exponential improvement over the trivial bound of $O(m3^n)$.

Theorem 6. *There is an algorithm to compute minimal models of 3-CNF theories and answer sets of disjunctive 3-programs, respectively, that runs in time $O(mn^2 2.2720..^n)$.*

4 Outlines of algorithms and proofs

Due to space limitations, we will not discuss here proofs of Theorems 2 and 3. We only mention that one can prove these results by modifying arguments we developed in [LT03] for the case of stable models of normal 2-programs. We will however sketch proofs of Theorems 4, 5 and 6, as they require several new techniques.

Let T be a CNF theory. By $Lit(T)$ we denote the set of all literals built of atoms in $At(T)$. For a set of literals $L \subseteq Lit(T)$, we define:

$$L^+ = \{a \in At(T) \colon a \in L\} \text{ and } L^- = \{a \in At(T) \colon \neg a \in L\}.$$

We also define $L_0 = L^+ \cup L^-$. A set of literals L is *consistent* if $L^+ \cap L^- = \emptyset$. A set of atoms $M \subseteq At(T)$ is *consistent* with a set of literals $L \subseteq Lit(T)$, if $L^+ \subseteq M$ and $L^- \cap M = \emptyset$.

Let T be a CNF theory and let $L \subseteq Lit(T)$ be a set of literals. By T_L we denote the theory obtained from T by removing:

```
$min^+(T, S, L)$
% $T$ and $S$ are CNF theories, $L$ is a set of literals
% $T$, $S$ and $L$ satisfy the preconditions: $L \subseteq Lit(T)$, and $S = T_L$
(0) if $S$ does not contain an empty clause then
(1)    if $S = \emptyset$ then
(2)       $M := L^+$; output$(M)$
(3)    else
(4)       $\mathcal{A} := complete(S)$;
(5)       for every $A \in \mathcal{A}$ do
(6)          $SA := S_A$;
(7)          $min^+(T, SA, L \cup A)$
(8) end of $min^+$.
```

Fig. 1. Algorithm *min*$^+$

1. every clause c that contains a literal $\ell \in L$ (intuitively, eliminate every clause that is subsumed by a literal in L)
2. all occurrences of literals built of atoms in L_0 that remain after performing step (1) (intuitively, resolve the remaining clauses with literals in L).

We note that it may happen that T_L contains empty clauses (is contradictory) or is empty (is a tautology). The theory T_L has the following important property.

Proposition 1. *Let T be a CNF theory and let $L \subseteq Lit(T)$. If $X \subseteq At(T)$ is a minimal model of T consistent with L, then $X \setminus L^+$ is a minimal model of T_L.*

We will now describe an algorithm *min*$^+$, from which we will subsequently derive algorithms for all three computational tasks that we study here. The input parameters of *min*$^+$ are: CNF theories T and S, and a set of literals L. We will require that $L \subseteq Lit(T)$ and $S = T_L$. We will refer to these two conditions as the *preconditions* for *min*$^+$. The output of the algorithm $min^+(T, S, L)$ is a family $\mathcal{M}_L^+(T)$ of sets containing all minimal models of T that are consistent with L. The input parameter S is determined by the two other parameters T and L (through the preconditions on T, S and L). We choose to specify it explicitly as that simplifies the description and the analysis of the algorithm.

A key concept that we need is that of a *complete* collection. Let T be a CNF theory. A non-empty collection $\mathcal{A}$ of non-empty subsets of $Lit(T)$ is *complete* for T if every minimal model of T is consistent with at least one set $A \in \mathcal{A}$. The collection $\mathcal{A} = \{\{a\}, \{\neg a\}\}$, where a is an atom of T, is a simple example of a complete collection for T.

In the description of the algorithm *min*$^+$ given in Figure 1, we assume that *complete* is a procedure computing, for an input CNF theory T, a complete collection of sets of literals.

We note that the algorithm $min^+(T, S, L)$ is well defined as the recursive calls $min^+(T, SA, L \cup A)$ (line (7)) are legal. Indeed, since $L_0 \cap A_0 = \emptyset$ and

$S = T_L$, $SA = S_A = (T_L)_A = T_{L\cup A}$. Moreover, $L \cup A \subseteq Lit(T)$. Thus, T, SA and $L \cup A$ satisfy the preconditions for the algorithm min^+.

Proposition 2. *Let T be a CNF theory and let $L \subseteq Lit(T)$ be a set of literals. If X is a minimal model of T consistent with L, then X is among the sets returned by $min^+(T, T_L, L)$.*

Proof: We prove the assertion, proceeding by induction on $|At(T_L)|$. Let us assume that $|At(T_L)| = 0$ and that X is a minimal model of T. By Proposition 1, $X \setminus L^+$ is a minimal model of T_L. In particular, T_L contains no empty clause. Since $|At(T_L)| = 0$, $T_L = \emptyset$. Consequently, $X \setminus L^+ = \emptyset$ (as $X \setminus L^+$ is a minimal model of T_L). It follows that, $X \subseteq L^+$. Furthermore, since X is consistent with L, $L^+ \subseteq X$. Thus, $X = L^+$. Finally, since T_L is empty, the program enters line (2) and outputs X since, as we already showed, $X = L^+$.

For the inductive step, let us assume that $|At(T_L)| > 0$ and that X is a minimal model of T consistent with L. By Proposition 1, $X \setminus L^+$ is a minimal model of T_L. Since $\mathcal{A}$, computed in line (4), is a complete collection for T_L, there is $A \in \mathcal{A}$ such that $X \setminus L^+$ is consistent with A. Clearly, $L_0 \cap A_0 = \emptyset$. Thus, X is consistent with $L \cup A$. By the induction hypothesis, the call $min^+(T, SA, L \cup A)$ (within loop (5)) returns the set X (as we observed earlier, the parameters T, SA and $L \cup A$ satisfy the preconditions for the algorithm min^+). □

Corollary 2. *Let T be a CNF theory. The family $\mathcal{M}^+_\emptyset(T)$ of sets that are returned by $min^+(T, T, \emptyset)$ contains all minimal models of T.*

Typically, algorithms for computing stable models and answer sets of logic programs, and minimal models of CNF theories search over a binary tree: at each branch point they select a variable and consider for it both truth assignments. The search tree traversed by our algorithm is not necessarily binary. At each branch point, the search splits into as many different paths as there are elements in the complete family returned by the procedure *complete*. While algorithms searching over binary trees can be derived from our template by designing the procedure *complete* to return collections consisting of at most two sets, the class of algorithms specified by our template is broader.

We will now study the performance of the algorithm min^+. First, for a CNF theory S we define a tree $\mathcal{T}_S$ inductively as follows. If S contains an empty clause or if S is empty, $\mathcal{T}_S$ consists of a single node labeled with S. Otherwise, let $\mathcal{A}$ be a complete family computed by the procedure *complete*(S). To construct tree $\mathcal{T}_S$, we create a node and label it with S. Next, we connect this node to roots of the trees $\mathcal{T}_{S_A}$, where $A \in \mathcal{A}$. Since the theories S_A have fewer atoms than S, the definition is well founded. We denote the set of leaves of the tree $\mathcal{T}_S$ by $L(\mathcal{T}_S)$.

One can show that for every CNF theory T and for every set of literals $L \subseteq Lit(T)$, the tree $\mathcal{T}_S$, where $S = T_L$, is precisely the tree of recursive calls to the procedure min^+ made by the top-level call $min^+(T, S, L)$. In particular, the tree $\mathcal{T}_T$ describes the exectution of the call $min^+(T, T, \emptyset)$. Since only those recursive calls that correspond to leaves of $\mathcal{T}_T$ produce output, Corollary 2 implies the following result.

Proposition 3. *Let T be a CNF theory. The number of minimal models of T is at most $|L(\mathcal{T}_T)|$.*

We use the tree $\mathcal{T}_T$ to estimate not only the number of minimal models of a CNF theory T but also the running time of the algorithm min^+. We say that an implementation of the procedure *complete* is *splitting* if for every theory T, such that $|At(T)| \geq 2$, it returns a complete family with at least two elements. We can show the following result.

Theorem 7. *Let us assume that there is a splitting implementation of the algorithm complete that runs in time $O(t(m))$, where t is an integer function such that $t(m) = \Omega(m)$. Then, for every CNF theory T, the algorithm min^+ runs in $O(|L(\mathcal{T}_T)|t(m))$ steps in the worst case.*

The specific bounds on $|L(\mathcal{T}_T)|$ and hence, on the number of minimal models of a CNF theory T and on the running time of the algorithm min^+ depend on the procedure *complete*. For 3-CNF theories we have the following results.

Theorem 8. *There is a splitting implementation of the procedure complete such that $t(m) = O(m)$ and for every 3-CNF theory T, $|L(\mathcal{T}_T)| \leq 1.6701..^n$.*

Corollary 3. *There is an implementation of the algorithm min^+ that, for 3-CNF theories, runs in time $O(m1.6701..^n)$.*

Corollary 3 is a direct consequence of Theorems 7 and 8. The proof of Theorem 7 is routine and we omit it. We will outline a proof of Theorem 8 in Section 5. In the remainder of this section, we will show that Theorem 8 and Corollary 3 imply our main results concerning minimal models of 3-CNF theories, stable models of normal 3-programs and answer sets of disjunctive 3-programs (Theorems 4, 5 and 6).

We start with the problem of computing minimal models of a CNF theory T. Let us assume that we have an algorithm *test_min* which, for a given CNF theory T and a set of atoms $M \subseteq At(T)$ returns the boolean value **true** if M is a minimal model of T, and returns **false**, otherwise.

We now modify the algorithm min^+ by replacing each occurrence of the command **output**(M) (line (2)), which outputs a set M, with the command

if *test_min*(T, M) **then output**(M).

We denote the resulting algorithm by *min_mod* (we assume the same preconditions on *min_mod* as in the case of min^+). Since all minimal models of T that are consistent with L belong to $\mathcal{M}_L^+(T)$ (the output of $min^+(T, S, L)$), it is clear that the algorithm *min_mod*(T, S, L) returns all minimal models of T.

Computation of stable models and answer sets of logic programs follows a similar pattern. First, let us recall that we can associate with a disjunctive logic program P (therefore, also with every normal logic program P) its propositional counterpart, a CNF theory $T(P)$ consisting of clauses of P but interpreted in propositional logic and rewritten into CNF. Specifically, to obtain $T(P)$ we replace each disjunctive program clause

$$c_1 \vee \ldots \vee c_p \leftarrow a_1, \ldots, a_r, \mathbf{not}(b_1), \ldots, \mathbf{not}(b_s)$$

in P with a CNF clause

$$\neg a_1 \vee \ldots \vee \neg a_r \vee b_1 \vee \ldots \vee b_s \vee c_1 \vee \ldots \vee c_p.$$

It is well known that stable models (answer sets) of (disjunctive) logic program P are minimal models of $T(P)$ [MT93].

Let us assume that *test_stb*(P, M) and *test_anset*(P, M) are algorithms to check whether a set of atoms M is a stable model and an answer set, respectively, of a program P.

To compute stable models of a logic program P that are consistent with a set of literals L, we first compute the CNF theory $T(P)$. Next, we run on the triple $T(P)$, $T(P)_L$ and L, the algorithm min^+ modified similarly as before (we note that the triple $(T(P), T(P)_L, L)$ satisfies its preconditions). Namely, we replace the command **output**(M) (line (2)), which outputs a set M, with the command

```
    if test_stb(P, M) then output(M).
```

The effect of the change is that we output only those sets in $\mathcal{M}_L^+(T(P))$, which are stable models of P. Since every stable model of P is a minimal model of $T(P)$, it is also an element of $\mathcal{M}_L^+(T(P))$. Thus, this modified algorithm, we will refer to it as *stb_mod*, indeed outputs all stable models of P and nothing more.

To design an algorithm to compute answer sets, we will refer to it as *ans_set*, we proceed in the same way. The only difference is that we use the algorithm *test_anset* in place of *test_stb* to decide whether to output a set.

Let t_1 be an integer function such that the worst-case running time of the algorithm min^+ is $O(t_1(n, m))$. Similarly, Let t_2 be an integer function such that the worst-case running time of the algorithm *test_min* (*test_stb* or *test_anset*, depending on the problem) is $O(t_2(n, m))$. The following observation is evident.

Proposition 4. *The running time of the algorithms min_mod, stb_mod and ans_set (in the worst case) is* $O(t_1(n, m) + \gamma t_2(n, m))$, *where* $\gamma = |L(\mathcal{T}_T)|$ *or* $|L(\mathcal{T}_{T(P)})|$, *depending on the problem.*

Proofs of Theorems 4 and 5. As we noted, stable models of a normal program P are minimal models of a CNF theory $T(P)$. The same is true for answer sets of a disjunctive logic program P. Thus, Theorem 4 follows from Proposition 3 and Theorem 8. It is well known that one can check in time $O(m)$ whether a set of atoms $M \subseteq At(P)$ is a stable model of a normal logic program P. Thus, Theorem 5 follows from Proposition 4, Theorem 8 and Corollary 3. □

We will now prove Theorem 6. We focus on the case of minimal models of 3-CNF theories. The argument in the case of answer sets of disjunctive 3-programs is similar. We start with a simple result on testing minimality.

Proposition 5. *Let p be an integer function and t an integer such that $t \geq 2$. If there is an algorithm that decides in time $O(p(n,m))$ whether a t-CNF theory T is satisfiable, then there is an algorithm that decides in time $O(|M|p(|M|, m+1)+m)$ whether a set $M \subseteq At(T)$ of atoms is a minimal model of a t-CNF theory T.*

Proof: Let $M = \{a_1, \ldots, a_k\}$. We define $L = \{\neg x \colon x \in At(T) \setminus M\}$ and observe that M is a minimal model of T if and only if M is a model of T and none of t-CNF theories $T_L \cup \{\neg a_i\}$, $i = 1, \ldots, k$, is satisfiable. $\square$

There is an algorithm to decide satisfiability of 3-CNF theories that runs in time $O(m1.4756..^n)$ [Rod96]. Thus, by Proposition 5, there is an an algorithm to decide whether a set $M \subseteq At(T)$ is a minimal model of a 3-CNF theory T that runs in time $O(|M|m1.4756..^{|M|})$. We denote this algorithm *test_min*.

Proof of Theorem 6. Let β be a real number such that $0.6 < \beta < 1$ (we will specify β later). To estimate the running time of the algorithm min^+, we split $\mathcal{M}_L^+(T)$ into two parts:

$$\mathcal{M}_1 = \{M \in \mathcal{M}_L^+(T) \colon |M| \geq \beta n\} \text{ and } \mathcal{M}_2 = \{M \in \mathcal{M}_L^+(T) \colon |M| < \beta n\}.$$

Clearly, the total time t_{min} needed to execute all calls to *test_min* throughout the execution of min^+ is:

$$t_{min} = \sum_{M \in \mathcal{M}_1} m|M|(1.4756..)^{|M|} + \sum_{M \in \mathcal{M}_2} m|M|(1.4756..)^{|M|}.$$

We have

$$\sum_{M \in \mathcal{M}_1} m|M|(1.4756..)^{|M|} = \sum_{i \geq \beta n} m\binom{n}{i} i(1.4756..)^i$$
$$\leq \beta m n^2 \binom{n}{\lceil \beta n \rceil}(1.4756..)^{\lceil \beta n \rceil}.$$

This last inequality follows from that fact that for every i, $i \geq 0.6n$,

$$m\binom{n}{i} i(1.4756..)^i \geq m\binom{n}{i+1}(i+1)(1.4756..)^{i+1},$$

and from the observation that the number of terms in the sum is less than n.

To estimate the second term, we note that $|\mathcal{M}_2| \leq |\mathcal{M}_L^+(T)|$ and, for every $M \in \mathcal{M}_2$, $|M| < \beta n$. Thus,

$$\sum_{M \in \mathcal{M}_2} m|M|(1.4756..)^{|M|} \leq (\beta m n)(1.6701..)^n(1.4756..)^{\beta n}.$$

Let us choose β so that $1.6701..^n = \binom{n}{\lceil \beta n \rceil}$. For this β, we obtain that $t_{min} = O(mn^2(1.6701..(1.4756..)^\beta)^n)$. One can verify that $\beta = 0.7907..$. It follows that the complexity of our algorithm is $O(mn^2 2.2720..^n)$, which completes the proof of Theorem 6. $\square$

5 Algorithm *complete* and a proof of Theorem 8

In this section we outline main ideas behind the proof of Theorem 8. They are concerned with the design of the procedure *complete*. In our approach, we are guided by a method to estimate the number of leaves in a search tree proposed in [Kul99]. We outline here this method adapted to our needs.

Let $\mathcal{T}$ be a rooted tree and let, as before, $L(\mathcal{T})$ be the set of leaves in $\mathcal{T}$. For a node x in $\mathcal{T}$, we denote by $C(x)$ the set of *directed* edges that link x with its children. For a leaf w of $\mathcal{T}$, we denote by $P(w)$ the set of *directed* edges on the unique path from the root of $\mathcal{T}$ to the leaf w. The following observation was shown in [Kul99].

Proposition 6. [Kul99] *Let p be a function assigning positive real numbers to edges of a rooted tree $\mathcal{T}$ such that for every internal node x in $\mathcal{T}$, $\sum_{e\in C(x)} p(e) = 1$. Then, $|L(\mathcal{T})| \leq \max_{w\in L(\mathcal{T})}(\prod_{e\in P(w)} p(e))^{-1}$.*

To estimate the number of leaves in our search tree $\mathcal{T}_T$, we associate with every 3-CNF theory T a certain measure μ of its complexity, which approximates the number of leaves in the tree $\mathcal{T}_T$. We define $\mu(T) = n(T) - \alpha c(T)$, where $n(T) = |At(T)|$, $c(T)$ is the maximum number of 2-clauses in T with pairwise disjoint sets of atoms, and α is a constant such that $0 \leq \alpha \leq 1$ (we will specify α later). Clearly, for every 3-CNF theory T, $\mu(T) \geq 0$.

This definition reflects the following intuition. The number of leaves in the tree $\mathcal{T}_T$ grows when the number of atoms in T grows. On the other hand, if T has a large number of 2-clauses, the number of leaves in $\mathcal{T}_T$ is usually smaller than in the case when T has few 2-clauses (and the same number of atoms).

For a directed edge $e = (T', T'')$ in the tree $\mathcal{T}_T$ (we recall that vertices of $\mathcal{T}_T$ are CNF theories), we define $\Delta(e) = \mu(T') - \mu(T'')$ and make an assumption (which we will verify later) that $\Delta(e) \geq 0$. Clearly, for every leaf $W \in L(\mathcal{T}_T)$,

$$\sum_{e\in P(W)} \Delta(e) = \mu(T) - \mu(W) \leq \mu(T) \leq n(T). \qquad (1)$$

Next, for every internal node S of the tree $\mathcal{T}_T$, by τ_S we denote the unique positive real number satisfying the equation

$$\sum_{e\in C(S)} \tau^{-\Delta(e)} = 1 \qquad (2)$$

and we define τ_0 to be the largest of these roots. One can verify that for each internal node S, $\tau_S \geq 1$. Thus, $\tau_0 \geq 1$, too.

Finally, for every edge $e = (S, S')$ in the tree $\mathcal{T}_T$, we define $p(e) = \tau_S^{-\Delta(e)}$. Since $\Delta(e) \geq 0$, $p(e)^{-1} = \tau_S^{\Delta(e)} \leq \tau_0^{\Delta(e)}$.

Let $W \in L(\mathcal{T}_T)$. We now have (the last inequality in the chain follows by (1))

$$(\prod_{e\in P(W)} p(e))^{-1} \leq \prod_{e\in P(W)} \tau_0^{\Delta(e)} = \tau_0^{\sum_{e\in P(W)} \Delta(e)} \leq \tau_0^{n(T)}.$$

By the definition, the function p satisfies the assumptions of Proposition 6. Thus (we recall that by our convention, $n(T) = n$),

$$|L(\mathcal{T}_T)| \leq \tau_0^n. \tag{3}$$

When designing the procedure *complete*, our goal is then to make sure that the value τ_0 be as small as possible. To this end, we consider several cases that reflect different structural properties of a 3-CNF theory S. In each of the cases we compute the numbers $\Delta(e)$, where $e = (S, S_A)$ and A is an element of the complete family produced by running the procedure *complete* on input S. Next we find the positive root τ_S of the equation (2).

To get the best upper bound for the number $|L(T)|$ of leaves in $\mathcal{T}_T$ (see inequality (3)), we choose the value of α so that the maximum of the solutions of the equation (2) over all cases considered in the definition of the procedure *complete* is as small as possible. This optimal value turns out to be $\alpha = 0.1950..$. Moreover, for this choice of α we can also show that for each edge $e = (S, S_A)$ in the tree $\mathcal{T}_T$, $\Delta(e) \geq 0$ thus, verifying an earlier assumption.

Since there are many cases in our definition of the procedure *complete*, we do not have space to give a full description of it. Instead we consider here in detail the two cases which actually determine the optimal value of α.

Case 1. We assume that the theory S has no 2-clauses and there are four clauses in S of the form $a \vee \beta_i \vee \gamma_i$, $i = 1, 2, 3, 4$, where a is an atom and $\beta_1, \ldots, \beta_4, \gamma_1, \ldots, \gamma_4$ are literals with pairwise distinct atoms different from a.

In this case the procedure *complete* outputs the family $\mathcal{A} = \{\{a\}, \{\neg a\}\}$. The set of atoms of both $S_{\{a\}}$ and $S_{\{\neg a\}}$ are $At(S) - \{a\}$ so $n(S_{\{a\}}) = n(S_{\{\neg a\}}) = n(S) - 1$. We also observe that, $c(S) = 0$ and that by the definition of $S_{\{\neg a\}}$, the theory $S_{\{\neg a\}}$ contains at least four 2-clauses $\beta_i \vee \gamma_i$, $i = 1, 2, 3, 4$, with pairwise different atoms. Hence $c(S_{\{\neg a\}}) \geq 4$. Consequently,

$$\Delta(S_{\{\neg a\}}) = \mu(S) - \mu(S_{\{\neg a\}}) = n(S) - n(S_{\{\neg a\}}) - \alpha(c(S) - c(S_{\{\neg a\}})) \geq 1 + 4\alpha.$$

The positive root of the equation (2) is in this case not larger than the positive root τ of the equation $\tau^{-1} + \tau^{-1-4\alpha} = 1$, where $\alpha = 0.1950..$ (indeed, the function $\phi(\tau, t_1, \ldots, t_k) = \sum_{i=1}^{k} \frac{1}{\tau^{t_i}}$ is decreasing for $\tau \geq 1$ and $t_1, \ldots, t_k \geq 0$). This root is $\tau = 1.6701..$.

Case 2. We assume that no two 2-clauses of the theory S have a common atom. Moreover there are five 2-clauses $a_i \vee b_i$, $i = 1, 2, 3, 4, 5$, and two 3-clauses $a_1 \vee a_2 \vee a_3$, $a_1 \vee a_4 \vee a_5$ in S, where $a_1, \ldots, a_5, b_1, \ldots, b_5$ are pairwise different atoms.

The following family

$$\mathcal{A}' = \{\{a_1\}, \{\neg a_1, a_2, a_4\}, \{\neg a_1, a_2, \neg a_4\}, \{\neg a_1, \neg a_2, a_4\}, \{\neg a_1, \neg a_2, \neg a_4\}\}$$

is complete. We observe that every model consistent with $\neg a_i$ is consistent with b_i because the clause $a_i \vee b_i$ has to be satisfied. Moreover, every model consistent with $\neg a_1$ and $\neg a_2$ (respectively $\neg a_1$ and $\neg a_4$) has to be consistent with a_3

(respectively a_5) because the clause $a_1 \vee a_2 \vee a_3$ (respectively $a_1 \vee a_4 \vee a_5$) has to be satisfied. These observations show that the family

$$\begin{aligned}\mathcal{A} = \ & \{\{a_1\}, \{\neg a_1, a_2, a_4, b_1\}, \{\neg a_1, a_2, \neg a_4, b_1, b_4, a_5\}, \{\neg a_1, \neg a_2, a_4, b_1, b_2, a_3\}, \\ & \{\neg a_1, \neg a_2, \neg a_4, b_1, b_2, b_4, a_3, a_5\}\}\end{aligned}$$

is complete.

For every $A \in \mathcal{A}$, the theory S_A contains all 2-clauses of S except for those which have an atom or its negation in A. Since no two 2-clauses in S have a common atom, the only clauses which are in S but not in S_A are of the form $a_i \vee b_i$. For example, for $A = \{\neg a_1, a_2, \neg a_4, b_1, b_4, a_5\}$, S_A has the same 2-clauses as S, except for $a_1 \vee b_1$, $a_2 \vee b_2$, $a_4 \vee b_4$, and $a_5 \vee b_5$, which are not in S_A. Thus, for this particular A, $c(S_A) = c(S) - 4$. Proceeding similarly for all the sets $A \in \mathcal{A}$, we get

$$c(S_A) = \begin{cases} c(S) - 1, \text{if } A = \{a_1\}; \\ c(S) - 3, \text{if } A = \{\neg a_1, a_2, a_4, b_1\}; \\ c(S) - 4, \text{if } A = \{\neg a_1, a_2, \neg a_4, b_1, b_4, a_5\}; \\ c(S) - 4, \text{if } A = \{\neg a_1, \neg a_2, a_4, b_1, b_2, a_3\}; \\ c(S) - 5, \text{if } A = \{\neg a_1, \neg a_2, \neg a_4, b_1, b_2, b_4, a_3, a_5\}. \end{cases}$$

Clearly, $\Delta(S_A) = \mu(S) - \mu(S_A) = |A| - \alpha(c(S) - c(S_A))$. Consequently,

$$\Delta(S_A) = \begin{cases} 1 - \alpha, \ \ \text{if } A = \{a_1\}; \\ 4 - 3\alpha, \text{if } A = \{\neg a_1, a_2, a_4, b_1\}; \\ 6 - 4\alpha, \text{if } A = \{\neg a_1, a_2, \neg a_4, b_1, b_4, a_5\}; \\ 6 - 4\alpha, \text{if } A = \{\neg a_1, \neg a_2, a_4, b_1, b_2, a_3\}; \\ 8 - 5\alpha, \text{if } A = \{\neg a_1, \neg a_2, \neg a_4, b_1, b_2, b_4, a_3, a_5\}. \end{cases}$$

Thus, the equation (2) in this case reduces to $\tau^{-1+\alpha} + \tau^{-4+3\alpha} + 2\tau^{-6+4\alpha} + \tau^{-8+5\alpha} = 1$. Its positive solution (assuming $\alpha = 0.1950..$) is $\tau = 1.6701..$.

It turns out that in all remaining cases the solutions of the equations (2) are smaller than 1.6701.. so, $\tau_0 = 1.6701..$. Thus, by the inequality (3), the bound on $|L(\mathcal{T}_T)|$ follows.

Given a 3-CNF theory T, one can recognize in time $O(m)$ which case in the procedure *complete* applies (clearly each of the cases we discussed here can be recognized in time $O(m)$ and all other cases are described in similar terms). Thus, our procedure *complete* can be implemented to run in time $t(m) = O(m)$. Moreover, in each case, one can produce the appropriate complete collection also in time $O(m)$. Lastly, one can design all the cases so that the procedure *complete* is splitting (again, this is evident for the two cases discussed here). These observations complete the proof of Theorem 8. □

6 Discussion

Algorithms we presented in the case of 2-CNF theories, and normal and disjunctive 2-programs have worst-case performance of $O(m1.4422..^n)$. The algorithm we designed for computing stable models of normal 3-programs runs in time $O(m1.6701..^n)$. Finally, our algorithms for computing minimal models of 3-CNF theories and answer sets of disjunctive logic programs run in time $O(mn^2 2.2720..^n)$. All these bounds improve by exponential factors over the corresponding straightforward ones.

The key question is whether still better algorithms are possible. In this context, we note that our algorithms developed for the case of 2-CNF theories and 2-programs are optimal, as long as we are interested in *all* minimal models, stable models and answer sets, respectively. However, we can compute a *single* minimal model of a CNF theory T or decide that T is unsatisfiable in *polynomial* time, using Proposition 5 and a well known fact that deciding satisfiability of 2-CNF theories is in P. In contrast, deciding whether a normal 2-program has a stable model and whether a disjunctive 2-program has an answer set is NP-complete. Thus, it is unlikely that there are polynomial-time algorithms to compute a single stable model (answer set) of a (disjunctive) 2-program or decide that none exist. Whether our bound of $O(m1.4422..^n)$ can be improved by an exponential factor if we are interested in computing a single stable model or a single answer set, rather than all of them, is an open problem.

The worst-case behavior of our algorithms designed for the case of 3-CNF theories and 3-programs does not match the lower bound of $O(n1.5848..^n)$ implied by Corollary 1. Thus, there is still room for improvement, even when we want to compute *all* minimal models, stable models and answer sets. In fact, we conjecture that exponentially faster algorithms exist.

In the case of 3-CNF theories, one can show, again using Proposition 5 and the algorithm from [Rod96], that there is a simple algorithm to compute *one* minimal model of a 3-CNF theory or determines that it is unsatisfiable, running in time $O(p(m,n)1.4756..^n)$, where p is a polynomial. This is a significantly lower bound than $O(mn^2 2.2720..^n)$ that we obtained for computing *all* minimal models. We do not know however, whether the bound $O(p(m,n)1.4756..^n)$ is optimal. Furthermore, we do not known whether an exponential improvement over the bound of $O(mn^2 2.2720..^n)$ is possible if we want to compute a single answer set of a disjunctive 3-program or determine that none exist. Similarly, we do not know whether one can compute a single stable model of a 3-program or determine that none exists in time exponentially lower than $O(m1.6701..^n)$.

In some cases, our bound in Theorems 6 can be improved. Let $\mathcal{F}$ be the class of all CNF theories consisting of clauses of the form $a_1 \vee \ldots \vee a_p$ or $a \vee \neg b$, where $a_1, \ldots, a_p$, a and b are atoms. Similarly, let $\mathcal{G}$ be the class of all disjunctive programs with clauses of the form $a_1 \vee \ldots \vee a_p \leftarrow \mathbf{not}(b_1), \ldots, \mathbf{not}(b_r)$ or $a \leftarrow b, \mathbf{not}(b_1), \ldots, \mathbf{not}(b_r)$, where $a_1, \ldots, a_p, b_1, \ldots, b_r$, a and b are atoms. Checking whether a set M is a minimal model of a theory from $\mathcal{F}$ or an answer set of a program from $\mathcal{G}$ is in P (can be solved in linear time). Thus, using Proposition 4, one can show the following result.

Theorem 9. *There is an algorithm to compute minimal models of 3-CNF theories in $\mathcal{F}$ (answer sets of disjunctive 3-programs in $\mathcal{G}$, respectively), that runs in time $O(m1.6701..^n)$.*

References

[Bar03] C. Baral. *Knowledge representation, reasoning and declarative problem solving.* Cambridge University Press, 2003. ISBN 0521818028.

[BDK97] G. Brewka, J. Dix, and K. Konolige. *Nonmonotonic Reasoning, An Overview.* CSLI Publications, 1997.

[BEP94] R. Ben-Eliyahu and L. Palopoli. Reasoning with minimal models: Efficient algorithms and applications. In *Proceedings of KR'94*, San Francisco, CA, 1994. Morgan Kaufmann.

[CL94] M. Cadoli and M. Lenzerini. The complexity of propositional closed world reasoning and circumscription. *Journal of Computer and System Sciences*, 48:255–310, 1994. Shorter version in the Proceedings of AAAI-90.

[EFLP00] Thomas Eiter, Wolfgang Faber, Nicola Leone, and Gerald Pfeifer. Declarative problem-solving in DLV. In Jack Minker, editor, *Logic-Based Artificial Intelligence*, pages 79–103. Kluwer Academic Publishers, Dordrecht, 2000.

[EG95] T. Eiter and G. Gottlob. On the computational cost of disjunctive logic programming: propositional case. *Annals of Mathematics and Artificial Intelligence*, 15(3-4):289–323, 1995.

[GL88] M. Gelfond and V. Lifschitz. The stable semantics for logic programs. In R. Kowalski and K. Bowen, editors, *Proceedings of the 5th International Conference on Logic Programming*, pages 1070–1080. MIT Press, 1988.

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

[Kul99] O. Kullmann. New methods for 3-SAT decision and worst-case analysis. *Theoretical Computer Science*, pages 1–72, 1999.

[Lif88] V. Lifschitz. Circumscriptive theories: a logic-based framework for knowledge representation. *Journal of Philosophical Logic*, 17(4):391–441, 1988.

[LT03] Z. Lonc and M. Truszczyński. Computing stable models: worst-case performance estimates. *Theory and Practice of Logic Programming*, 2003. To appear.

[McC80] J. McCarthy. Circumscription — a form of non-monotonic reasoning. *Artificial Intelligence*, 13(1-2):27–39, 1980.

[MT93] W. Marek and M. Truszczyński. *Nonmonotonic Logic; Context-Dependent Reasoning.* Springer-Verlag, Berlin, 1993.

[MT99] V.W. Marek and M. Truszczyński. Stable models and an alternative logic programming paradigm. In K.R. Apt, W. Marek, M. Truszczyński, and D.S. Warren, editors, *The Logic Programming Paradigm: a 25-Year Perspective*, pages 375–398. Springer Verlag, 1999.

[Nie96] Ilkka Niemelä. A tableau calculus for minimal model reasoning. In *Proceedings of the Fifth Workshop on Theorem Proving with Analytic Tableaux and Related Methods*, Lecture Notes in Computer Science, pages 278–294. Springer-Verlag, 1996.

[Nie99] I. Niemelä. Logic programming with stable model semantics as a constraint programming paradigm. *Annals of Mathematics and Artificial Intelligence*, 25(3-4):241–273, 1999.

[Rod96] R. Rodošek. A new approach on solving 3-satisfiability. In *Proc. 3rd Int. Conf. on AI and Symbolic Math. Comput.*, pages 197–212. Springer-Verlag, 1996. LNCS 1138.

[SNS02] P. Simons, I. Niemelä, and T. Soininen. Extending and implementing the stable model semantics. *Artificial Intelligence*, 138:181–234, 2002.