Counterexample Guided Abstraction Refinement Algorithm for Propositional Circumscription*

Mikoláš Janota[1], Radu Grigore[2], and Joao Marques-Silva[3]

[1] INESC-ID, Lisbon, Portugal
[2] Queen Mary, University of London
[3] University College Dublin, Ireland

Abstract. Circumscription is a representative example of a nonmonotonic reasoning inference technique. Circumscription has often been studied for first order theories, but its propositional version has also been the subject of extensive research, having been shown equivalent to extended closed world assumption (ECWA). Moreover, entailment in propositional circumscription is a well-known example of a decision problem in the second level of the polynomial hierarchy. This paper proposes a new Boolean Satisfiability (SAT)-based algorithm for entailment in propositional circumscription that explores the relationship of propositional circumscription to minimal models. The new algorithm is inspired by ideas commonly used in SAT-based model checking, namely counterexample guided abstraction refinement. In addition, the new algorithm is refined to compute the theory closure for generalized close world assumption (GCWA). Experimental results show that the new algorithm can solve problem instances that other solutions are unable to solve.

1 Introduction

Closed world reasoning (CWR) and circumscription (CIRC) are well-known nonmonotonic reasoning techniques, that find a wide range of practical applications. Part of the interest in these techniques is that they bring us closer to how humans reason [16,18,17]. While these techniques have been studied in the context of both first-order and propositional logic, this paper addresses the propositional case. Research directions that have characterized the study of nonmonotonic reasoning techniques include expressiveness, computational complexity, applications and algorithms. The different CWR rules proposed in the late 70s and 80s illustrate the evolution in terms of expressive power in first-order and propositional logics. The computational complexity of propositional CWR rules was studied in the early 90s [1,6] and showed that, with few exceptions, the complexity of CWR deduction problems are in the second level of the polynomial hierarchy, being Π_2^{P}-complete [6]. Nonmonotonic reasoning finds a wide range of applications in Artificial Intelligence (AI), but also in description logics [7]

* This work is partially supported by SFI project BEACON (09/IN.1/I2618) and European projects COCONUT (FP7-ICT-217069) and MANCOOSI (FP7-ICT-214898).

and in interactive configuration [13], among many others. Finally, different algorithms have been proposed over the years, examples of which include minimal model resolution [21], tableau calculus [19], Quantified Boolean Formula (QBF) solvers [5] and Disjunctive Logic Programming (DLP) [15,12,20].

The main contribution of this paper is to propose a new algorithm for solving the deduction problem for the propositional version of some CWR rules and for propositional circumscription. The new algorithm is based on iterative calls to a SAT solver, and is motivated by the practical success of modern SAT solvers. However, given the complexity class of entailment for CWR rules, a SAT solver can be expected to be called an exponential number of times in the worst case, or be required to process an exponentially large input. To cope with this issue, we utilize a technique inspired in counterexample guided abstraction refinement (CEGAR), widely used in model checking [3]. One of the key ideas of the new algorithm is that we try to prove a stronger formula, which is weakened if it turns out to be too strong. Based on this idea we develop an algorithm that decides entailment in circumscription. Further, we refine the algorithm to compute the closure of a formula defined by one of the variants of CWR, namely GCWA. As a result, the main contributions of the paper can be summarized as follows: (i) A novel algorithm for propositional circumscription that does not require an enumeration of all minimal models or prime implicates; (ii) Specialization of this algorithm to compute variables that are 0 in all minimal models; and (iii) Computing the closure of GCWA.

2 Preliminaries

All variables are propositional, and represented by a finite set V. A *Conjunctive Normal Form* (CNF) formula ϕ is a conjunction of *clauses*, which are disjunctions of *literals*, which are possibly negated variables. A formula ϕ can also be viewed as a set of sets of literals. The two representations are used interchangeably in this paper. A clause is called *positive*, if it contains only positive literals. Arbitrary Boolean formulas will also be considered, for which the standard definitions apply. A *variable assignment* ν is a total function from V to $\{0, 1\}$. In the text, a variable assignment is represented as $\{x_1^{v_1}, \dots, x_n^{v_n}\}$ where $V = \{x_1, \dots, x_n\}$ and $v_i \in \{0, 1\}$, $i \in 1..n$. For a variable assignment ν and a formula ϕ we write $\nu \models \phi$ to denote that ν satisfies ϕ. In this case, ν is called a *model* of ϕ. We write $\phi \models \psi$ if the models of ϕ are also models of ψ. Given a set of variables $S \subseteq V$ and $v \in \{0, 1\}$, the expression $\phi[S \mapsto v]$ denotes the formula ϕ with all variables in S replaced with v.

2.1 Minimal Models

Minimal models are widely used in nonmonotonic reasoning and AI in general. To introduce minimal models, we consider the bitwise ordering on variable assignments. For variable assignments ν and μ we write $\nu \leq \mu$ and say that ν is *smaller* than μ iff $(\forall x \in V)(\nu(x) \leq \mu(x))$. We write $\nu < \mu$ and say that ν is *strictly smaller* than μ iff $\nu \leq \mu$ and $\nu \neq \mu$. A model ν of ϕ is a *minimal model* iff there is no model of ϕ strictly smaller than ν. Finally, we write $\phi \models_{\min} \psi$ if ψ holds in all minimal models of ϕ.

Proposition 1. *The models of formula ϕ that are strictly smaller than some variable assignment ν are the models of the formula*

$$\phi \wedge \bigwedge\nolimits_{\nu(x)=0} \neg x \wedge \bigvee\nolimits_{\nu(x)=1} \neg x \tag{1}$$

2.2 Closed World Reasoning

The intuition behind closed world assumption (CWA) reasoning is that facts are not considered to be true unless they were specifically stated. This is motivated by the type of reasoning humans use on an everyday basis. For instance, if Alice asks Bob to buy eggs, Bob will clearly buy eggs. However, he will not buy bread even though Alice has not specified that the bread should not be bought. Traditional mathematical logic behaves differently in this respect: the fact *buy-eggs* trivially entails *buy-eggs* but does not entail the fact $\neg$*buy-bread.*

This intuition has been realized by several different formalisms. Here we present only a small portion of these formalisms and the interested reader is referred to appropriate publications for further reference [1,6,4].

The standard formulation of CWA rules partitions set V into three sets: P, Q and Z, where P denotes the variables to be minimized, Z are the variables that can change when minimizing the variables in P, and Q represents all other (fixed) variables. For any set R, R^+ and R^- denote, respectively, the sets of positive and negative literals from variables in R. Following [1,6], a closure operation is defined for CWR rules as follows:

Definition 1. *Let ϕ be a propositional formula, $\langle P; Q; Z\rangle$ a partition of V, and α a CWR-rule. Then, the* closure *of ϕ with respect to α is defined by,*

$$\alpha(\phi; P; Q; Z) = \phi \cup \{\neg K \mid K \textit{ is free for negation in } \phi \textit{ w.r.t. } \alpha\} \tag{2}$$

Each CWR rule considers a different set of formulas that are free for negation. For each CWR rule below, a formula K is free for negation if and only if the corresponding condition holds:

GCWA (Generalized CWA [18]): K is a positive literal and for every positive clause B such that $\phi \not\models B$ it holds that $\phi \not\models B \vee K$.

EGCWA (Extended GCWA [25]): K is a conjunction of positive literals and for every positive clause B such that $\phi \not\models B$ it holds that $\phi \not\models B \vee K$.

ECWA (Extended CWA [25]): K is an arbitrary formula not involving literals from Z, and for every positive clause B whose literals belong to $P^+ \cup Q^+ \cup Q^-$, such that $\phi \not\models B$, it holds that $\phi \not\models B \vee K$.

We consider only a subset of existing CWR rules. A detailed characterization for existing CWR rules can be found elsewhere [1,6,4].

Observe that a single positive literal is free for negation in both GCWA and EGCWA under the same conditions. Since a positive literal corresponds to some variable, we extend the terminology for variables accordingly.

Definition 2. *A variable x is* free for negation in ϕ *iff for every positive clause B such that $\phi \not\models B$ it holds that $\phi \not\models B \vee v$.*

Another concept closely related to closed world assumption is circumscription. Originally, McCarthy defined circumscription in the context of first order logic as a closure of the given theory that considers only predicates with minimal extension [16]. In propositional logic, circumscription of a formula yields a formula whose models are the minimal models of the original one.

Definition 3. *Consider the sets of variables P, Q and Z introduced above. The circumscription of a formula ϕ is defined as follows:*

$$\mathit{CIRC}(\phi; P; Q; Z) = \phi \wedge (\forall_{P',Z'})((\phi(P'; Q; Z') \wedge (P' \Rightarrow P)) \Rightarrow (P \Rightarrow P')) \quad (3)$$

Where P', Z' are sets of variables s.t. $X' = \{x' \mid x \in X\}$; $\phi(P', Q, Z')$ is obtained from $\phi(P, Q, Z)$ by replacing the variables in P and Z by the corresponding variables in P' and Z'; finally, $P' \Rightarrow P$ stands for $\bigwedge_{x \in P}(x' \Rightarrow x)$.

In the remainder of the paper the sets Z and Q are assumed to be *empty*. The extension to the general case where these sets are not empty is simple and is outlined in an extended version of the article [14].

It is well-known that for the propositional case, circumscription is equivalent to ECWA [9]. Another well-known relationship is the one of both CWR rules and circumscription to minimal models (e.g. [18,1,6]). In particular variables free for negation take value 0 in all minimal models. And, both EGCWA and circumscription entail the same set of facts as the set of minimal models. These relations are captured by the following propositions (adapted from [18,1,6]):

Proposition 2. *A variable x is free for negation in a formula ϕ iff x is assigned value 0 in all minimal models of ϕ.*

Proposition 3. *Let ϕ and ψ be formulas. It holds that $\mathit{EGCWA}(\phi) \models \psi$ iff $\phi \models_{\min} \psi$. And, it holds that $\mathit{CIRC}(\phi) \models \psi$ iff $\phi \models_{\min} \psi$.*

3 Problems

The CWR rules yield the two following problems. The first problem consists of computing the closure of the theory, as defined by the CWR rule. The second problem is that of computing whether a certain fact is entailed by that closure.

If the closure has been computed, standard satisfiability algorithms can be used to solve the entailment problem. However, whereas the closure of GCWA increases the size of the formula by at most a linear number of literals, the closure of both ECWA and EGCWA may increase the size of the formula by an exponential number of conjuncts of literals. The circumscription of a formula can be constructed easily but gives rise to a QBF formula and our objective is to stay within propositional logic with the ultimate goal of developing purely SAT-based solutions. Hence, this paper focuses on the following problems.

ENTAILS-MIN
instance: formulas ϕ and ψ
question: Does the formula ψ hold in all minimal models of ϕ?

FREE-FOR-NEGATION
instance: formula ϕ and variable $x \in V$
question: Does x take value 0 in all minimal models of ψ?

FREE-FOR-NEGATION-ALL
instance: formula ϕ and a variable $v \in V$
question: What is the set of variables with value 0 in all minimal models of ϕ?

Note that solving ENTAILS-MIN enables answering whether a fact is entailed by ECWA or by circumscription due to Proposition 3. Clearly, the problem FREE-FOR-NEGATION is a special case of ENTAILS-MIN with ψ set to $\neg x$. Solving FREE-FOR-NEGATION-ALL gives us the closure of GCWA.

Interestingly, in terms of complexity, the problem FREE-FOR-NEGATION is not easier than the problem ENTAILS-MIN. Both ENTAILS-MIN and FREE-FOR-NEGATION are Π_2^{P}-complete [6, Lemma 3.1].

4 Computing ENTAILS-MIN

The algorithm we wish to develop will be using a SAT solver. This gives us two objectives. One objective is to construct a propositional formula that corresponds to the validity of $\phi \models_{\min} \psi$. The second objective is to avoid constructing an exponentially large formula. We begin by observing that if $\phi \models_{\min} \psi$ is to hold, then any model of ϕ that violates ψ must *not* be a minimal model.

Proposition 4. *ψ holds in all minimal models of ϕ iff any model ν of ϕ where $\neg\psi$ holds is not a minimal model of ϕ.*

$$[\phi \models_{\min} \psi] \Leftrightarrow [(\forall \nu)\,((\nu \models \phi \wedge \neg\psi) \Rightarrow (\exists \nu')(\nu' < \nu \wedge \nu' \models \phi))]$$

Proposition 4 tells us that whether $\phi \models_{\min} \psi$ holds or not can be decided by deciding whether the following formula is valid:

$$(\forall \nu)\,((\nu \models \phi \wedge \neg\psi) \Rightarrow (\exists \nu')(\nu' < \nu \wedge \nu' \models \phi)) \tag{4}$$

Since our first objective is to find a propositional formula, we need to eliminate $\cdot \models \cdot$ and quantifiers from (4). First, let us focus on the subformula $(\exists \nu')(\nu' < \nu \wedge \nu' \models \phi)$, which expresses that ν is not a minimal model.

Proposition 5. *A model ν of ϕ is* not *minimal iff there exists a set S of variables such that ν is a model of $\phi[S \mapsto 0]$, and $\nu(x) = 1$ for some $x \in S$.*

$$(\exists \nu')(\nu' < \nu \wedge \nu' \models \phi) \Leftrightarrow (\exists S \subseteq V)\,(\nu \models \phi[S \mapsto 0] \wedge (\exists x \in S)(\nu(x) = 1)) \quad (5)$$

Example 1. Let $\phi = \neg x \vee y$. The model $\mu = \{x^0, y^0\}$ is minimal and the right-hand side of (5) is invalid since there is no set S satisfying the condition $(\exists x \in S)(\nu(x) = 1)$. Let $\nu = \{x^0, y^1\}$ and let us choose $S = \{x, y\}$, which yields $\phi[S \mapsto 0] = 1$. ν is *not* minimal and the right-hand side of (5) is valid since $\nu \models 1$ and $\nu(y) = 1$.

Replacing the left-hand side of (5) with the right-hand side of (5) in (4) yields the following formula:

$$(\forall \nu)\,((\nu \models \phi \wedge \neg\psi) \Rightarrow (\exists S \subseteq V)\,(\nu \models \phi[S \mapsto 0] \wedge (\exists x \in S)(\nu(x) = 1))) \quad (6)$$

Removing the universal quantifier and replacing existential quantifiers with the Boolean operator $\vee$ in (6), gives us that (6) holds iff the following formula is a tautology:

$$(\phi \wedge \neg\psi) \Rightarrow \bigvee_{S \in \mathcal{P}(V)} \left(\phi[S \mapsto 0] \wedge \bigvee_{x \in S} x \right) \quad (7)$$

Intuitively, (7) expresses that if ψ is violated in a model of ϕ, then a different model of ϕ is obtained by flipping a set of variables to 0. That this model is indeed different is guaranteed by the condition $\bigvee_{x \in S} x$. The model obtained by the flipping serves as a *witness* of that the model violating ψ is *not* minimal.

If (7) is constructed, its validity can be decided by calling a SAT solver on its negation. However, the formula is too large to construct since it requires considering all subsets of V. Therefore, we construct a stronger version of it that considers only *some* subsets of V. This stronger version is referred to as the *abstraction* of (7) and always has the following form:

$$(\phi \wedge \neg\psi) \Rightarrow \bigvee_{S \in W} \left(\phi[S \mapsto 0] \wedge \bigvee_{x \in S} x \right) \qquad \text{where } W \subseteq \mathcal{P}(V) \quad (8)$$

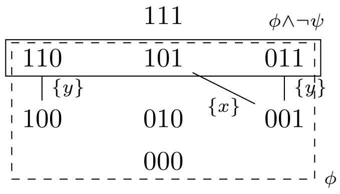
Each abstraction is determined by a set of sets of variables W. For any W, if the abstraction (8) is shown to be a tautology, then (7) is also a tautology and we are done because we have shown that $\phi \models_{\min} \psi$. If the abstraction is not a tautology, it is either because $\phi \models_{\min} \psi$ does not hold or the abstraction is overly strong—it is too coarse. If the abstraction is shown to be too coarse, a different abstraction must be considered.

Example 2. Let us show that $\neg x \vee y \models_{\min} \neg y$. First, let us try $W_1 = \{\{y\}\}$, which yields the abstraction $((\neg x \vee y) \wedge y) \Rightarrow \neg x$. This abstraction is not a tautology. In particular, it is violated by the assignment $\{x^1, y^1\}$, which means that flipping y to value 0 in this assignment does not yield a model. Now, let us try $W_2 = \{\{x, y\}\}$, which yields the abstraction $((\neg x \vee y) \wedge y) \Rightarrow 1$. This abstraction is a tautology, which means that any model where y is 1 can be turned into another model by flipping both x and y to 0. Therefore, $\neg x \vee y \models_{\min} \neg y$.

```
input  : formulas $\phi$ and $\psi$
output: true iff $\phi \models_{\min} \psi$
1  $\omega \leftarrow \phi \wedge \neg\psi$
2  while true do
3      $(\mathsf{outc}_1, \nu) \leftarrow \mathtt{SAT}(\omega)$
4      if $\mathsf{outc}_1 = \mathsf{false}$ then
5          return true                                  // no counterexample was found
6      $(\mathsf{outc}_2, \nu') \leftarrow \mathtt{SAT}\left(\phi \wedge \bigwedge_{\nu(x)=0} \neg x \wedge \bigvee_{\nu(x)=1} \neg x\right)$      // find $\nu' < \nu$
7      if $\mathsf{outc}_2 = \mathsf{false}$ then                         // $\nu$ is minimal
8          return false                        // abstraction cannot be refined
9      $S \leftarrow \{x \in V \mid \nu(x) = 1 \wedge \nu'(x) = 0\}$
10     $\omega \leftarrow \omega \wedge (\neg\phi[S \mapsto 0] \vee \bigwedge_{x \in S} \neg x)$                       // refine
```

Algorithm 1. Refining

Example 3. Let $\phi = \neg x \vee \neg y \vee \neg z$ and $\psi = (\neg x \vee \neg y) \wedge (\neg x \vee \neg z) \wedge (\neg z \vee \neg y)$ Let us show that $\phi \models_{\min} \psi$. Let us choose the abstraction defined by the set $W = \{\{x\}, \{y\}\}$. The following diagram demonstrates that each model violating ψ has a witness corresponding to one of the sets in W.



Each triple represents a variable assignment where the elements represent the values of x, y, and z, respectively. Models and their pertaining witnesses are connected by an edge, which is labeled by the set of variables S whose values are flipped to 0 *to obtain the witness.*

The approach of searching for the right abstraction follows the Counter-Example Guided Abstract Refinement (CEGAR) loop [3]. If the abstraction is a tautology, the search terminates. If the abstraction is not a tautology, it is weakened by adding some set of variables S to the set W. This weakening is referred to as *refinement* and is done by investigating the counterexample that shows that the current abstraction is not a tautology. If it cannot be refined, (7) is not a tautology and $\phi \models_{\min} \psi$ does not hold.

Algorithm 1 realizes the idea outlined above. The algorithm maintains the negation of the abstraction in variable ω and starts with W being the empty set. Therefore the initial abstraction is $(\phi \wedge \neg\psi) \Rightarrow 0$ with the negation being $\phi\wedge\neg\psi$ (line 1). The test whether the abstraction is a tautology or not is done by calling a SAT solver on its negation (line 3). If the negation is unsatisfiable—the abstraction is a tautology—then the algorithm terminates and returns true (line 4). If a model ν is found showing that the abstraction is not a tautology, it means that for any assignment that is obtained from ν by flipping some set of variables in $S \in W$ to 0 is not a model of ϕ. The algorithm looks for a model ν' that is strictly smaller than ν applying Proposition 1 (line 6). If there is no model strictly smaller than ν then the algorithm terminates and returns false since ν is a minimal model and violates ψ (line 7). If there is a model ν' that is strictly smaller than ν, there is some set of variables that are 1 in ν but are 0 in ν'. This set of variables is added to the sets determining the abstraction (line 10). Observe that a set S will be used at most once to refine the abstraction since once the set is added to W, an assignment for which flipping 1 to 0 for variables in S yields a model cannot satisfy the negation of the abstraction. Consequently, the algorithm is terminating and will perform at most as many iterations as there are subsets of the set V.

5 Computing Free-For-Negation

This section specializes Algorithm 1 to compute variables free for negation—variables that take value 0 in all minimal models. As mentioned earlier, this problem is a special case of the problem Entails-Min, studied in the previous section: x is free for negation in ϕ iff $\phi \models_{\min} \neg x$. However, focusing on this type of formulas enables a more efficient implementation of the algorithm.

The abstractions used in the previous section have to contain the condition that at least one of the variables being flipped to 0 is 1 to guarantee the corresponding witnesses is strictly smaller (see (7)). For variables free for negation these conditions will not be needed thanks to the following proposition.

Proposition 6. *Let ν be a model of ϕ s.t. $\nu(x) = 1$ for a variable x. If x is free for negation, then there exists a model ν' of ϕ s.t. $\nu' < \nu$ and $\nu'(x) = 0$.*

Proposition 6 tells us that if $\nu(x) = 1$ and x is free for negation, there must be a witness ν' that flips x to 0 (and possibly some other variables). This ensures that ν and ν' are different. This observation enables us to compute $\phi \models_{\min} \neg x$ by determining the validity of a stronger and more concise formula than before.

Proposition 7. *A variable x is free for negation in ϕ iff the following formula is a tautology.*

$$(\phi \wedge x) \Rightarrow \bigvee_{S \subseteq V \wedge x \in S} \phi[S \mapsto 0] \tag{9}$$

The abstraction of (9) is analogous to the one used in the previous section with the difference that only sets of variables containing x are considered. Hence, the abstraction always has the following form.

$$(\phi \wedge x) \Rightarrow \bigvee_{S \in W} \phi[S \mapsto 0], \text{ where } W \subseteq \mathcal{P}(V) \text{ and } (\forall S \in W)(x \in S) \tag{10}$$

5.1 Constructing and Refining Abstraction

Whenever the abstraction is being refined (weakened) the size of the formula representing the negation of the abstraction increases. Since the abstraction is refined in the worst case exponentially many times, it is warranted to pay attention to the size of the formula representing the negation of the abstraction.

The negation of an abstraction is a conjunct of the left-hand side of the implication and formulas capturing the substitutions.

```
input  : CNF formula $\phi$ and a variable $x$
output: true iff $\phi \models_{\min} \neg x$
1 $\phi_0 \leftarrow \phi[x \mapsto 0]$
2 $\phi'_0 \leftarrow \{\neg r_c \vee c \mid c \in \phi_0\} \cup \{\neg l \vee r_c \mid c \in \phi_0, l \in c\} \cup \left\{\bigvee_{c \in \phi_0} \neg r_c\right\}$
3 $\omega \leftarrow \phi \wedge x \wedge \phi'_0$
4 while true do
5     $(\mathsf{outc}_1, \nu) \leftarrow \mathtt{SAT}(\omega)$
6     if $\mathsf{outc}_1 = \mathsf{false}$ then
7         return true                          // no counterexample was found
8     $(\mathsf{outc}_2, \nu') \leftarrow \mathtt{SAT}\left(\phi \wedge \neg x \wedge \bigwedge_{\nu(z)=0} \neg z\right)$   // find $\nu' < \nu$ and $\nu'(x) = 0$
9     if $\mathsf{outc}_2 = \mathsf{false}$ then
10        return false                         // abstraction cannot be refined
11    $S \leftarrow \{z \in V \mid \nu(z) = 1 \wedge \nu'(z) = 0\}$
12    $C_p \leftarrow \{c \in \phi_0 \mid (c \cap S) \neq \emptyset\}$          // clauses with some $y \in S$
13    $C_n \leftarrow \{c \in \phi_0 \mid (c \cap \neg S) \neq \emptyset\}$         // clauses with some $\neg y \in S$
14    $C \leftarrow \{c' \mid c \in (C_p \setminus C_n) \wedge c' = c[S \mapsto 0]\}$        // new clauses
15    $\omega \leftarrow \omega \cup \{\neg r_c \vee c \mid c \in C\} \cup \{\neg l \vee r_c \mid c \in C, l \in c\}$   // representation
16    $\omega \leftarrow \omega \cup \left\{\bigvee_{c \in \phi \setminus (C_n \cup C_p)} \neg r_c \vee \bigvee_{c \in C} \neg r_c\right\}$       // negation of clauses
```

Algorithm 2. Deciding whether a variable is free for negation

$$(\phi \wedge x) \wedge \bigwedge_{S \in W} \neg\phi[S \mapsto 0], \text{ where } W \subseteq \mathcal{P}(V) \text{ and } (\forall S \in W)(x \in S) \tag{11}$$

When the abstraction is being refined, a new set of variables S is added to the set W, therefore, the negation of the abstraction is strengthened by conjoining it with $\neg\phi[S \mapsto 0]$. We aim to implement this strengthening without duplicating those parts of the formula that are already present.

Algorithm 2 outlines this procedure. Since all the sets S must contain x, the algorithm starts with the abstraction determined by $W = \{\{x\}\}$. In the initialization phase, the negation of this abstraction is $\phi \wedge x \wedge \neg\phi[x \mapsto 0]$ and is computed using the Tseitin transformation [24]. Each clause c in $\phi[x \mapsto 0]$ is represented by a fresh variable r_c and a clause is added that expresses that at least one of these variables must be 0 (line 2). As in the previous section, variable ω represents the negation of the abstraction (see (11)).

When the abstraction is being refined, the formula in variable ω is conjoined with $\neg\phi[S \mapsto 0]$. Since ω already contains clauses from $\neg\phi[x \mapsto 0]$, we need to consider only those clauses that contain literals on the variables in S. Clauses containing negative literals on variables from S are skipped, positive literals are removed. Each of the affected clauses is represented by a fresh Tseitin variable. Finally, a clause is added to express that one of the clauses in $\phi[S \mapsto 0]$ is 0. Note that this clause is referring to the original Tseitin variables for the clauses that are not affected by the substitution besides the freshly created ones. Note that when looking for a model $\nu' < \nu$, the algorithm requires that x has value 0 in ν' since the set S must contain x (line 8).

5.2 Finding Models

An abstraction is refined according to two responses from the underlying SAT solver (ν and ν'). This enables us to devise heuristics that prefer some responses of the solver to another. The motivation for these heuristics is to find abstractions where the set W determining the abstraction contains few sets S. Dually, this means that each of $S \in W$ yields a witness for many models. The heuristics used in the current implementation are motivated by the two following examples.

Example 4. Let $\phi = (x \Rightarrow y) \wedge (w \vee z)$. The abstraction defined by $W = \{\{x, y\}\}$ shows that $\phi \models_{\min} \neg y$ since flipping both x and y in any model yields a model (a witness). The abstraction determined by $W = \{\{x, y, z\}\}$ is not sufficient. This abstraction provides a witness for models with w having value 1 but not for the others. Intuitively, variable z is irrelevant to the relation $x \Rightarrow y$ and therefore it is better to choose a small S.

Example 5. Let $\phi = x \Rightarrow (y \vee w_1 \vee \dots w_n)$ and let us prove that $\phi \models_{\min} \neg y$. The abstraction determined by $W = \{\{x, y\}\}$ is sufficient. However, if ν is not minimal, it may be that $\nu = x^1, y^1, w_1^1, \dots w_n^1$ which gives us an exponential number of possibilities for ν' while only one of them is desirable. Intuitively, if ν is not minimal and there is some set S that yields a witness for both ν and some $\nu_1 < \nu$, then the set S is more likely to be found when ν_1 is inspected.

Based on this last observation, the model ν is required to be minimal. To make the difference between ν and ν' small, and therefore make this set S small, the solution ν' is required to be a maximal model.

To obtain a minimal, respectively maximal, model from a SAT solver is done by specifying the *phase*—the value that the solver prefers when making decisions when traversing the search space. Namely, preferring 0 yields a minimal model while preferring 1 yields a maximal model [10,22].

6 Computing FREE-FOR-NEGATION-ALL

To calculate the set of variables that are free for negation, we invoke the algorithm described in the previous section for each variable. This procedure is optimized by conjoining the negations of the variables that have already been shown to be free for negation, which is justified by the following proposition.

Proposition 8. *Let ϕ and ψ be formulas such that $\phi \models_{\min} \psi$. The formula $\phi \wedge \psi$ has the same set of minimal models as ϕ. In particular, if $\phi \models_{\min} \neg x$ then $(\phi \wedge \neg x) \models_{\min} \neg y$ iff $\phi \models_{\min} \neg y$.*

The motivation for conjoining negations of variables free for negation is to give more information to subsequent inferences. The effectiveness of this technique, however, depends on the ordering of the variables. Hence, the approach we use is to set timeouts for testing a single variable and if a test times out, the variable is tested again but with information gained from the other tests.

```
input : CNF formula $\phi$ and a set of variables $V$
output: subset of $V$ that are free for negation
$F \leftarrow \emptyset$
$X \leftarrow V$
timeout $\leftarrow$ initial-timeout
while $X \neq \emptyset$ do
    $G \leftarrow \emptyset$
    foreach $x$ in $X$ do
        (success, outc) $\leftarrow$ Free-For-Negation($\phi, x$, timeout)
        if success = true then
            $G \leftarrow G \cup \{x\}$
            if outc = true then
                $F = F \cup \{x\}$
                $\phi = \phi \land \neg x$
    $X \leftarrow X \setminus G$
    timeout $\leftarrow k \times$ timeout
    return $F$
```

Algorithm 3. Computing the set of variables that are free for negation

Algorithm 3 summarizes these ideas in pseudocode. The algorithm described in the previous section is represented by the function Free-For-Negation, which returns a pair of values. The first value in the pair indicates whether the algorithm terminated before the given timeout or not. The second value of the pair indicates whether the given variable is free for negation or not. The timeout is gradually multiplied by some constant coefficient k. In the actual implementation there is a maximum timeout for which the algorithm stops and returns an approximation of the set of variables free for negation.

7 Evaluation

Algorithm 3 was implemented in Java using SAT4J as the underlying SAT solver while availing of its incremental interface [23]. The implementation was evaluated on a benchmark of 260 tests[1]. A majority of these are valid software configurations (motivated by [13]). A few tests are from the SAT '09 competition—relatively easy instances were chosen as the computed problem is significantly harder than satisfiability. The results appear in Table 1. An instance is considered solved if the answer is given in less than 30 s. The time given in the table is the average for the solved instances.

The alternative we tried was based on the tool `circ2dlp` [20], which transforms circumscription into a disjunctive logic program, and `gnt` [11], which lists all models of that program. From the list of models it is easy and fast to construct the set of variables that are free for negation. We also tried using a QBF solver along with (3), but that implementation solved *none* of the 260 tests.

[1] Available at `http://logos.ucd.ie/confs/jelia10/jelia10-bench.tgz`

Table 1. Experimental evaluation

		Algorithm 3		circ2dlp+gnt	
	tests	solved	time[s]	solved	time[s]
e-shop	174	174	2.1	95	2.4
BerkeleyDB	30	30	0.9	30	< 0.1
model-transf	41	41	1.1	35	2.8
SAT2009	15	3	7.6	2	2.5

8 Summary and Future Work

This paper proposes an algorithm for deduction under the set of minimal models of a propositional formula. This algorithm enables us to reason under the propositional versions of close world assumption or circumscription. The algorithm hinges on an application of a SAT solver but more importantly on counterexample guided abstraction refinement (CEGAR). While CEGAR has been amply used in software verification [3,8],we are not aware of its application in nonmonotonic reasoning.

The deduction problem under the set of minimal models can be formulated as QBF [5] or as a DLP [15,12]. The experimental results suggest that current QBF solvers are not practical for this problem. The comparison to the DLP-based solution indicates that our dedicated algorithm enables solving more instances. Nevertheless, the DLP-based solution was faster for some instances.

The promising experimental results indicate that the ideas behind the presented algorithms have potential for further work. The evaluation was performed for the computation of variables free for negation defining the closure of a theory in GCWA, hence, further evaluations should be performed on other types of problems in this domain. On a more general scale, it is well known that minimal models can be seen as optima with respect to the pertaining ordering [2,22]. This opens possibilities to investigate generalizations of the presented algorithms for different orderings than the one used for minimal models. Last but not least, the comparison with the DLP-based solution indicates that it would be beneficial to investigate approaches tackling the problem with hybrid techniques.

References

1. Cadoli, M., Lenzerini, M.: The complexity of closed world reasoning and circumscription. In: AAAI Conference on Artificial Intelligence, pp. 550–555 (1990)
2. Castell, T., Cayrol, C., Cayrol, M., Berre, D.L.: Using the Davis and Putnam procedure for an efficient computation of preferred models. In: European Conference on Artificial Intelligence, pp. 350–354 (1996)
3. Clarke, E.M., Grumberg, O., Jha, S., Lu, Y., Veith, H.: Counterexample-guided abstraction refinement. In: Emerson, E.A., Sistla, A.P. (eds.) CAV 2000. LNCS, ch. 1855, pp. 154–169. Springer, Heidelberg (2000)
4. Dix, J., Furbach, U., Niemelä, I.: Nonmonotonic reasoning: Towards efficient calculi and implementations. In: Voronkov, A., Robinson, A. (eds.) Handbook of Automated Reasoning, vol. 19, pp. 1241–1354. North-Holland, Amsterdam (2001)

5. Egly, U., Eiter, T., Tompits, H., Woltran, S.: Solving advanced reasoning tasks using quantified boolean formulas. In: AAAI Conference on Artificial Intelligence, pp. 417–422 (2000)
6. Eiter, T., Gottlob, G.: Propositional circumscription and extended closed-world reasoning are Π_2^P-complete. Theor. Comput. Sci. 114(2), 231–245 (1993)
7. Eiter, T., Ianni, G., Lukasiewicz, T., Schindlauer, R., Tompits, H.: Combining answer set programming with description logics for the Semantic Web. Artif. Intell. 172(12-13), 1495–1539 (2008)
8. Flanagan, C., Qadeer, S.: Predicate abstraction for software verification. In: Principles of programming languages (POPL), pp. 191–202. ACM, New York (2002)
9. Gelfond, M., Przymusinska, H., Przymusinski, T.C.: On the relationship between circumscription and negation as failure. Artif. Intell. 38(1), 75–94 (1989)
10. Giunchiglia, E., Maratea, M.: Solving optimization problems with DLL. In: European Conference on Artificial Intelligence, pp. 377–381 (2006)
11. Janhunen, T., Niemelä, I., Seipel, D., Simons, P., You, J.H.: Unfolding partiality and disjunctions in stable model semantics. ACM Trans. Comput. Log. 7(1), 1–37 (2006)
12. Janhunen, T., Oikarinen, E.: Capturing parallel circumscription with disjunctive logic programs. In: European Conf. on Logics in Artif. Intell., pp. 134–146 (2004)
13. Janota, M., Botterweck, G., Grigore, R., Marques-Silva, J.: How to complete an interactive configuration process? In: Conference on Current Trends in Theory and Practice of Computer Science, pp. 528–539 (2010)
14. Janota, M., Grigore, R., Marques-Silva, J.: Counterexample guided abstraction refinement algorithm for propositional circumscription. Tech. Rep. TR-32-2010, INESC-ID Lisboa (2010)
15. Lifschitz, V.: Foundations of logic programming. In: Principles of Knowledge Representation, pp. 69–127 (1996)
16. McCarthy, J.: Circumscription - a form of non-monotonic reasoning. Artif. Intell. 13(1-2), 27–39 (1980)
17. McCarthy, J.: Applications of circumscription to formalizing common-sense knowledge. Artif. Intell. 28(1), 89–116 (1986)
18. Minker, J.: On indefinite databases and the closed world assumption. In: Loveland, D.W. (ed.) CADE 1982. LNCS, vol. 138, pp. 292–308. Springer, Heidelberg (1982)
19. Niemelä, I.: Implementing circumscription using a tableau method. In: European Conference on Artificial Intelligence, pp. 80–84 (1996)
20. Oikarinen, E., Janhunen, T.: circ2dlp — translating circumscription into disjunctive logic programming. In: Baral, C., Greco, G., Leone, N., Terracina, G. (eds.) LPNMR 2005. LNCS (LNAI), vol. 3662, pp. 405–409. Springer, Heidelberg (2005)
21. Przymusinski, T.C.: An algorithm to compute circumscription. Artif. Intell. 38(1), 49–73 (1989)
22. Rosa, E.D., Giunchiglia, E., Maratea, M.: Solving satisfiability problems with preferences. Constraints. An International Journal (2010) (in press)
23. SAT4j, `http://www.sat4j.org`
24. Tseitin, G.S.: On the complexity of derivation in propositional calculus. Studies in constructive mathematics and mathematical logic 2(115-125), 10–13 (1968)
25. Yahya, A.H., Henschen, L.J.: Deduction in non-Horn databases. Journal of Automated Reasoning 1(2), 141–160 (1985)