Query Answering in Propositional Circumscription

Mario Alviano
University of Calabria
alviano@mat.unical.it

Abstract

Propositional circumscription defines a preference relation over the models of a propositional theory, so that models being subset-minimal on the interpretation of a set of objective atoms are preferred. The complexity of several computational tasks increases by one level in the polynomial hierarchy due to such a preference relation; among them there is query answering, which amounts to decide whether there is an optimal model satisfying the query. A complete algorithm for query answering is obtained by searching for a model, not necessarily an optimal one, that satisfies the query, and such that no model unsatisfying the query is more preferred. If the query or its complement are among the objective atoms, the algorithm has a simpler behavior, which is also described in the paper. Moreover, an incomplete algorithm is obtained by searching for a model satisfying both the query and an objective atom being unit-implied by the theory extended with the complement of the query. A prototypical implementation is tested on instances from the 2nd International Competition on Computational Models of Argumentation.

1 Introduction

Circumscription [McCarthy, 1980] enforces the minimization of the extension of some predicates via a second order semantics. In the propositional case, such a minimization essentially selects subset minimal models. A revised version introduced the possibility to specify a set of atoms for grouping interpretations, and another set of atoms subject to minimization [Lifschitz, 1986]. In this form, circumscription is suitable for modeling several practical problems that are characterized by a preference relation over admissible solutions. Among them there are minimal diagnosis of faulty systems [Pereira *et al.*, 1993; Jannach *et al.*, 2016] and minimal correction subsets [Junker, 2004; Marques-Silva *et al.*, 2013; Mencía *et al.*, 2015].

The enumeration problem associated with propositional circumscription was recently addressed by modifying ONE [Alviano *et al.*, 2015b; Alviano and Dodaro, 2016; 2017], an algorithm for MaxSAT based on unsatisfiable core analysis [Morgado *et al.*, 2013]. Such an algorithm was implemented in CIRCUMSCRIPTINO and used by PYGLAF [Alviano, 2017a; 2017c] to solve several problems of the 2nd International Competition on Computational Models of Argumentation (ICCMA'17) [Gaggl *et al.*, 2016] by means of linear translations into the framework of propositional circumscription. Among these problems there are credulous and skeptical acceptance of arguments, which essentially amount to decide whether a given argument belongs to respectively some or all extensions of the graph in input.

Concerning acceptance problems, the *contra* of the strategy implemented by PYGLAF is that an enumeration algorithm provides a negative answer to credulous acceptance problems only after producing all models; a similar observation holds for positive answers to skeptical acceptance problems. This is in contrast with propositional logic and answer set programming, for which acceptance problems can be addressed efficiently by several algorithms [Alviano *et al.*, 2014; 2018] implemented in WASP [Dodaro *et al.*, 2011; Alviano *et al.*, 2013; 2015a]. The *pro* of PYGLAF is that the underlying solver for circumscription is used as a black box, which means that a better solution for some computational problems of circumscription can be easily used also for argumentation reasoning.

Motivated by the above analysis, the present paper provides algorithms for deciding the existence of a circumscribed model of the input theory satisfying a given query. More specifically, the framework is simplified in the form of a propositional theory associated with a set of objective literals subject to maximization (Section 2). A complete algorithm is obtained by searching for a model satisfying the query, and then by verifying that no model in which the query is unsatisfied is more preferred (Section 3); if this last check fails, a countermodel is identified, and the search for the next model satisfying the query is constrained to bypass the inhibition provided by the countermodel. An interesting aspect of this algorithm is that neither models nor countermodels are necessarily optimal models, and therefore queries may be answered without computing any optimal model. Moreover, the algorithm can be efficiently implemented by means of two SAT solvers, the first for producing models, and the second for producing countermodels (Section 6); the SAT solvers process propositional theories that are extended during the computation, and possibly subject to assumption literals, so that learned clauses are possibly reused in different computations.

The fact that the algorithm searches for a model and verifies the nonexistence of countermodels is justified by the complexity of query answering, which is in general Σ_2^P-complete for propositional circumscription [Eiter and Gottlob, 1993]. However, there are restricted cases for which the complexity of the problem drops by one level in the polynomial hierarchy. One of such cases is the occurrence of the query in the set of objective literals: any model I satisfying the query is sufficient to conclude the existence of an optimal model satisfying the query, as any model unsatisfying the query cannot be a countermodel of I. In this case, the second solver used by the algorithm immediately detects the nonexistence of any countermodel, and therefore does not add any overhead to the computation.

A generalization of the previous case is obtained by noting that any objective literal ℓ whose complement is unit-implied by the theory extended with the complement of the query is such that the theory extended with ℓ entails the query. Hence, any model satisfying ℓ is sufficient to conclude the existence of an optimal model satisfying the query. Based on this observation, an incomplete algorithm is presented (Section 4), and possibly combined with the complete algorithm.

Acceptance problems of abstract argumentation framework are mapped to query answering (Section 5), and the algorithms are tested empirically on instances of ICCMA'17 (Section 7). Specifically, preferred and semistable extensions are considered. The implemented solver is compared with ARGSEMSAT [Cerutti *et al.*, 2013] and CEGARTIX [Dvorák *et al.*, 2014], reporting very positive results.

2 Background

Let $\mathcal{A}$ be a fixed, countable set of *atoms*. A *literal* is an atom possibly preceded by the connective $\neg$. For a literal ℓ, let $\overline{\ell}$ denote its *complementary literal*, that is, $\overline{p} = \neg p$ and $\overline{\neg p} = p$ for all $p \in \mathcal{A}$; for a set L of literals, let $\overline{L}$ be $\{\overline{\ell} \mid \ell \in L\}$. *Formulas* are defined as usual by combining atoms and the connectives $\top$, $\bot$, $\neg$, $\wedge$, $\vee$, $\rightarrow$, and $\leftrightarrow$. A *theory* is a set Γ of formulas; the set of atoms occurring in Γ is denoted by $atoms(\Gamma)$.

An *assignment* is a set A of literals such that $A \cap \overline{A} = \emptyset$. An *interpretation* for a theory Γ is an assignment I such that $(I \cup \overline{I}) \cap \mathcal{A} = atoms(\Gamma)$. Relation $\models$ is defined as usual: $I \models \top$; $I \not\models \bot$; for $p \in \mathcal{A}$, $I \models p$ if $p \in I$; for ϕ and ψ formulas, $I \models \neg\phi$ if $I \not\models \phi$, $I \models \phi \wedge \psi$ if $I \models \phi$ and $I \models \psi$, $I \models \phi \vee \psi$ if $I \models \phi$ or $I \models \psi$, $I \models \phi \rightarrow \psi$ if $I \models \psi$ whenever $I \models \phi$, and $I \models \phi \leftrightarrow \psi$ if either $I \models \phi$ and $I \models \psi$, or $I \not\models \phi$ and $I \not\models \psi$. I is a *model* of a theory Γ if $I \models \Gamma$. Let $models(\Gamma)$ denote the set of models of Γ.

Circumscription applies to a theory Γ and disjoint sets P, Z of atoms; atoms in P are subject to minimization, while atoms in Z are irrelevant. Formally, relation $\leq^{PZ}$ is defined as follows: for I, J interpretations of Γ, $I \leq^{PZ} J$ if both $(\mathcal{A} \setminus (P \cup Z)) \cap I = (\mathcal{A} \setminus (P \cup Z)) \cap J$ and $P \cap I \subseteq P \cap J$. $I \in models(\Gamma)$ is a *circumscribed model* of Γ with respect to $\leq^{PZ}$ if there is no $J \in models(\Gamma)$ such that $I \not\leq^{PZ} J$ and $J \leq^{PZ} I$. Let $CIRC(\Gamma, P, Z)$ denote the set of circumscribed models of Γ with respect to $\leq^{PZ}$.

Algorithm 1: AnwerQuery(Γ, O, q)

```
1  $I' := \emptyset$;
2  loop
3      $I := solve(\Gamma \cup \{q\} \cup (O \cap I'))$;
4      if $I \neq \bot$ then
5          $I' := solve(\Gamma \cup \{\overline{q}\} \cup \{\bigvee O \setminus I\} \cup (O \cap I))$;
6          if $I' = \bot$ then return YES;
7      else
8          if $I' = \emptyset$ then return NO;
9          $\Gamma := \Gamma \cup \{\bigvee O \setminus I'\}$;
10         $I' := \emptyset$;
```

Example 1. Let Γ_1 be $\{\neg r \rightarrow a, \neg z \rightarrow a\}$. Hence, $CIRC(\Gamma_1, \{a\}, \{z\})$ contains $\{r, \neg a, z\}$, $\{\neg r, a, z\}$, and $\{\neg r, a, \neg z\}$. Note that $\{r, a, z\}$ is a model of Γ_1, but $\{r, \neg a, z\} \leq^{\{a\}\{z\}} \{r, a, z\}$; similar for $\{r, a, \neg z\}$. ■

The framework of circumscription is simplified by focusing on a theory Γ and on a set O of *objective literals* subject to maximization. A model I of Γ is *optimal* with respect to O if there is no $I' \in models(\Gamma)$ such that $O \cap I' \supset O \cap I$. Abusing of notation, let $models(\Gamma, O)$ denote the set of optimal models of Γ with respect to O. Note that Γ_1 from Example 1 satisfies $models(\Gamma_1, \{\neg a, r, \neg r\}) = CIRC(\Gamma_1, \{a\}, \{z\})$, and such a result holds in general.

Proposition 1. *For any theory* Γ, *and any disjoint sets* P, Z *of atoms,* $CIRC(\Gamma, P, Z) = models(\Gamma, \overline{P} \cup R \cup \overline{R})$, *where* R *is* $atoms(\Gamma) \setminus (P \cup Z)$.

The computational problem addressed in this paper, referred to as *query answering*, is the following: Given a theory Γ, a set O of objective literals, and a literal q called *query*, decide whether there is an optimal model $I \in models(\Gamma, O)$ such that $I \models q$.

Example 2. Let Γ_{run} be the theory $\{\varphi_1 \vee \varphi_2 \vee \varphi_3 \vee \varphi_4 \vee \varphi_5\}$, where φ_i are the following subformulas:

$$\begin{array}{ll} \varphi_1 := \neg a \wedge \neg b \wedge \neg c \wedge q & \varphi_4 := a \wedge b \wedge \neg c \wedge q \\ \varphi_2 := \neg a \wedge b \wedge \neg c \wedge \neg q & \varphi_5 := \neg a \wedge \neg b \wedge c \wedge \neg q \\ \varphi_3 := \neg a \wedge b \wedge \neg c \wedge q & \end{array}$$

The models of Γ_{run} are $I_1 = \{\neg a, \neg b, \neg c, q\}$, $I_2 = \{\neg a, b, \neg c, \neg q\}$, $I_3 = \{\neg a, b, \neg c, q\}$, $I_4 = \{a, b, \neg c, q\}$, and $I_5 = \{\neg a, \neg b, c, \neg q\}$. Let O_{run} be $\{a, b, c\}$. Hence, $models(\Gamma_{run}, O_{run})$ is $\{I_4, I_5\}$. The answer to query q over Γ_{run} and O_{run} is YES because of I_4. ■

3 Complete Algorithm for Query Answering

The proposed strategy to address query answering is reported in Algorithm 1. The idea is to compute a model I of the the input theory Γ such that $I \models q$, and then check whether there exists no model I' of Γ such that both $O \cap I' \supset O \cap I$ and $I' \not\models q$, where O is the set of objective literals. If such a model I' does exist, a new model I is searched, this time with the additional requirement that $O \cap I \supseteq O \cap I'$. This process is repeated until no such an I can be found, witnessing that this branch of the search space does not contain any optimal model making the query true. Hence, the branch is discarded by means of a blocking clause based on the last computed I', so that the algorithm can restart from a different branch. The algorithm can conclude the falsity of the query when no I is found after blocking a branch of computation.

More in detail, model searches are performed by means of function $solve$, whose input is a theory Γ, and whose output is either $\bot$ if $models(\Gamma) = \emptyset$, or a model $I \in models(\Gamma)$. A blocking clause for a set of objective literals O and an interpretation I is the clause $\bigvee O \setminus I$; such a clause enforces the satisfaction of at least one objective literal being false according to I. The blocking clause can be combined with the set $O \cap I$ of formulas in order to enforce the search of a model I' such that $O \cap I' \supset O \cap I$; indeed, note that $O \cap I$ alone enforces the search of a model I' such that $O \cap I' \supseteq O \cap I$.

Summing up, Algorithm 1 initially sets I' to the empty set (line 1), so that a model of $\Gamma \cup \{q\}$ is searched at line 3. If a model I is found (line 4), a countermodel is searched, that is, a model I' of Γ such that $O \cap I' \supset O \cap I$ and $I' \not\models q$ (line 5). If function $solve$ returns $\bot$, then the algorithm terminates providing a positive answer (line 6); note that I is not necessarily an optimal model of Γ, but any model I' of Γ such that $O \cap I' \supset O \cap I$ must necessarily be such that $I' \models q$. Otherwise, function $solve$ returns a model I', and the algorithm continues from line 3 by searching for a model of $\Gamma \cup \{q\}$ such that $O \cap I \supseteq O \cap I'$. When the search for I terminates with $\bot$, the algorithm distinguishes two cases: if $I' = \emptyset$, no model of $\Gamma \cup \{q\}$ does exist, and a negative answer is provided (line 8); otherwise, the current branch of computation is discarded by the blocking clause $\bigvee O \setminus I'$ (line 9), I' is reset (line 10), and the algorithm restarts from line 3.

Example 3 (Continuing Example 2)**.** The execution of $\text{AnswerQuery}(\Gamma_{run}, O_{run}, q)$ initially searches for a model I of Γ_{run} such that $I \models q$ (line 3); say that I_1 is returned. The call to function $solve$ at line 5 returns either I_2 or I_5; say that I_2 is returned, that is, I' is I_2 in the next check at line 3. Hence, $O_{run} \cap I' = \{b\}$, and either I_3 or I_4 is returned; say that I_3 is returned, so that theory $\Gamma_{run} \cup \{\neg q\} \cup \{a \vee c\} \cup \{b\}$ is processed at line 5. Since $\bot$ is returned, the algorithm terminates at line 6 answering YES. Note that I_3 is not optimal, but there is sufficient evidence that an optimal model satisfying the query does exist (in this case, I_4).

As for an alternative execution, consider the case in which the first call to $solve$ at line 3 returns I_1, and the first call at line 5 returns I_5. Hence, I' is I_5 in the next check at line 3, and $\bot$ is returned because no model of Γ_{run} satisfies both q and c. Therefore, the theory is extended with $\bigvee O_{run} \setminus I_5 = a \vee b$, and I' is reset (lines 9–10). The next check at line 3 returns either I_3 or I_4, and the algorithm continues as in the previous execution. ■

Theorem 1. *Let Γ be a theory, O be a set of literals, and q be a literal. Algorithm 1 terminates, and returns* YES *if there is $I \in models(\Gamma, O)$ such that $I \models q$, and* NO *otherwise.*

Proof. We shall show the following properties: (P1) no optimal model of Γ is discarded by the new clauses added at line 9; (P2) if YES is returned, there is $I^* \in models(\Gamma, O)$ such that $I^* \models q$; (P3) if NO is returned, there is no $I^* \in models(\Gamma, O)$ such that $I^* \models q$; (P4) one of the return instructions at lines 6 and 8 is eventually executed.

Concerning (P1), line 9 is executed only if $I' \in models(\Gamma)$ is such that $I' \models \overline{q}$, and for all $I \in models(\Gamma)$ such that $O \cap I \supseteq O \cap I'$, it holds that $I \not\models q$; stated differently, any model I containing q is such that $O \cap I \not\supseteq I'$, and therefore $I \models \bigvee O \setminus I'$. Hence, $\text{AnswerQuery}(\Gamma, O, q) = \text{AnswerQuery}(\Gamma \cup \{\bigvee O \setminus I'\}, O, q)$.

Concerning (P2), YES is returned at line 6 only if there is no $I' \in models(\Gamma)$ such that $I' \models \overline{q}$, and $O \cap I' \supset O \cap I$, where $I \in models(\Gamma)$ is such that $I \models q$. Hence, $I' \models q$ holds for all $I' \in models(\Gamma)$ such that $O \cap I' \supset O \cap I$. If $I \in models(\Gamma, O)$, the claim holds (for $I^* = I$). Otherwise, there is $I^* \in models(\Gamma)$ such that $O \cap I^* \supset O \cap I$, and therefore $I^* \models q$.

Concerning (P3), NO is returned at line 8 only if $I' = \emptyset$, and there is no $I \in models(\Gamma)$ such that $I \models q$. Since $models(\Gamma, O) \subseteq models(\Gamma)$, the claim holds.

Concerning (P4), let I_i, I'_i be the values of variables I, I' at iteration $i \geq 0$. If the return instructions are not reached at iteration i, then the algorithm is in one of two possible cases. In the first case, $I_i = \bot$ and $I_i \neq \emptyset$. Since the new clause added to Γ discards at least one model, and the number of models of Γ is finite, this case occurs finitely many times during any execution. In the second case, $I_i \neq \bot$ and $I'_i \neq \bot$. Hence, either $O \cap I_{i+1} \supseteq O \cap I'_i \supset O \cap I_i$ (which is possible at most $|O|$ consecutive times), or $I_{i+1} = \bot$ (and the first case can occur finitely many times). □

Algorithm 2: AnwerQuery(Γ, $O \cup \{q\}$, q)

```
1 if $solve(\Gamma \cup \{q\}) \neq \bot$ then return YES;
2 return NO;
```

Algorithm 1 addresses query answering in the general setting. Interestingly, if either q or $\overline{q}$ is part of the objective literals, many calls to function $solve$ are completed immediately because both q and $\overline{q}$ are formulas of the tested theory, which is therefore unsatisfiable. Specifically, if $q \in O$, the call to function $solve$ at line 5 returns $\bot$: $q \in I$ because of line 3, and therefore $q \in O \cap I$; hence, both q and $\overline{q}$ are part of the tested theory, which is trivially unsatisfiable. It turns out that in this case query answering can be achieved by a single call to function $solve$, as reported in Algorithm 2. Correctness of the algorithm follows from Theorem 1.

Corollary 1. *Let Γ be a theory, O be a set of literals, and q be a literal. Algorithm 2 terminates, and returns* YES *if there is $I \in models(\Gamma, O \cup \{q\})$ such that $I \models q$, and* NO *otherwise.*

Example 4 (Continuing Example 3)**.** The execution of $\text{AnswerQuery}(\Gamma_{run}, O_{run} \cup \{q\}, q)$ initially searches for a model I of Γ_{run} such that $I \models q$; as in the previous example, say that I_1 is returned. The algorithm can already answer YES; in fact, Algorithm 1 would process the theory $\Gamma_{run} \cup \{\neg q\} \cup \{a \vee b \vee c\} \cup \{q\}$, which is unsatisfiable. ■

On the other hand, if $\overline{q} \in O$ and $I' \neq \emptyset$, the call to function $solve$ at line 3 of Algorithm 1 returns $\bot$: $\overline{q} \in I'$ because of line 5, and therefore $\overline{q} \in O \cap I'$; hence, both q and $\overline{q}$

Algorithm 3: AnwerQuery(Γ, $O \cup \{\overline{q}\}$, q)

```
loop
    $I := solve(\Gamma \cup \{q\})$;
    if $I = \bot$ then return NO;
    $I' := solve(\Gamma \cup \{\overline{q}\} \cup (O \cap I))$;
    if $I' = \bot$ then return YES;
    $\Gamma := \Gamma \cup \{\bigvee O \setminus I'\}$;
```

Algorithm 4: IncompleteAnwerQuery(Γ, O, q)

```
for $\ell \in O$ such that $solve(\Gamma \cup \{\overline{q}\} \cup \{\ell\}) = \bot$ do
    if $solve(\Gamma \cup \{q\} \cup \{\ell\}) \neq \bot$ then return YES;
return UNKNOWN;
```

are part of the tested theory, which is trivially unsatisfiable. It turns out that blocking clauses can be immediately added after a countermodel is computed. Additionally, the theory processed at line 5 of Algorithm 1 can be simplified by removing the disjunction $\bigvee O \setminus I$, which necessarily contains $\overline{q}$; similarly, blocking clauses at line 9 of Algorithm 1 necessarily contain $\overline{q}$, which is necessarily false at line 3 and can be therefore removed. Such simplifications are reported in Algorithm 3, whose correctness follows from Theorem 1.

Corollary 2. *Let Γ be a theory, O be a set of literals, and q be a literal. Algorithm 3 terminates, and returns* YES *if there is $I \in models(\Gamma, O \cup \{\overline{q}\})$ such that $I \models q$, and* NO *otherwise.*

Example 5 (Continuing Example 3)**.** The execution of $\text{AnswerQuery}(\Gamma_{run}, O_{run} \cup \{\neg q\}, q)$ initially searches for a model I of Γ_{run} such that $I \models q$; as in the previous examples, say that I_1 is returned. Say that the call to $solve(\Gamma_{run} \cup \{\neg q\} \cup \emptyset)$ returns I_2. The algorithm can already extend the theory with $a \vee c$; in fact, Algorithm 1 would process the theory $\Gamma_{run} \cup \{q\} \cup \{b, \neg q\}$, which is unsatisfiable. After that, I_4 is assigned to I, so that the next check at line 4 of Algorithm 3 returns $\bot$, and the algorithm answers YES. ■

4 Incomplete Algorithm for Query Answering

The incomplete algorithm is based on the following property.

Lemma 1. *Let Γ be a theory, O be a set of literals, and q be a literal. If there is $O' \subseteq O$ such that $models(\Gamma \cup O' \cup \{q\}) \neq \emptyset$ and $models(\Gamma \cup O' \cup \{\overline{q}\}) = \emptyset$, then there is $I^* \in models(\Gamma, O)$ such that $I^* \models q$.*

Proof. Let $I \in models(\Gamma \cup O' \cup \{q\})$. If $I \in models(\Gamma, O)$, the claim holds (for $I^* = I$). Otherwise, consider any $I^* \in models(\Gamma, O)$ such that $O \cap I^* \supset O \cap I \supseteq O$. Since $models(\Gamma \cup O' \cup \{\overline{q}\}) = \emptyset$ by assumption, and $I^* \models \Gamma \cup O'$, we conclude that $I^* \models q$. □

Intuitively, the lemma above highlights a sufficient but not necessary condition to conclude that the query is true: if there is a set O' of objective literals that can be extended to a model of Γ, and such that any extension to a model of Γ contains the query q, then there is also an optimal model of Γ with respect to O containing q.

The incomplete algorithm is reported as Algorithm 4. Since checking all subsets of O is impractical, the algorithm focuses on singletons. As will be clarified in Section 6, the algorithm can be further constrained to check only singletons $\{\ell\}$ such that $models(\Gamma \cup \{\overline{q}\} \cup \{\ell\}) = \emptyset$ is verifiable in polynomial time.

Theorem 2. *Let Γ be a theory, O be a set of literals, and q be a literal. Algorithm 4 terminates, and if it returns* YES *then there is $I \in models(\Gamma, O)$ such that $I \models q$.*

Proof. The algorithm performs at most $2 \cdot |O|$ satisfiability checks, so termination is guaranteed. It returns YES if there is $\ell \in O$ such that $models(\Gamma \cup \{\overline{q}\} \cup \{\ell\}) = \emptyset$, and $models(\Gamma \cup \{q\} \cup \{\ell\}) \neq \emptyset$. Hence, correctness follows from Lemma 1 for $O' = \{\ell\}$. □

Example 6 (Continuing Example 3)**.** The execution of $\text{IncompleteAnswerQuery}(\Gamma_{run}, O_{run}, q)$ detects that $solve(\Gamma_{run} \cup \{\neg q\} \cup \{a\}) = \bot$, and $solve(\Gamma_{run} \cup \{q\} \cup \{a\}) = I_4$. Hence, the algorithm returns YES. ■

5 Argumentation Acceptance Problems

An *abstract argumentation framework* (AF) is a directed graph G whose nodes $arg(G)$ are arguments, and whose arcs $att(G)$ represent an attack relation. An *extension* E is a set of arguments, and its *range* is $E^+ := E \cup \{x \mid \exists yx \in att(G)$ with $y \in E\}$. Let $\mathbf{CO}(G)$ contain any extension E of G having the following properties: there are no $x, y \in E$ with $xy \in att(G)$ (*conflict-free*); for all $yx \in att(G)$ such that $x \in E$, there is $zy \in att(G)$ such that $z \in E$ (*admissible*); if $x \in arg(G)$ is such that for all $yx \in att(G)$ there is $zy \in att(G)$ with $z \in E$, then $x \in E$ (*complete*). E is *preferred* if $E \in \mathbf{CO}(G)$, and there is no $E' \in \mathbf{CO}(G)$ such that $E' \supset E$. E is *semi-stable* if $E \in \mathbf{CO}(G)$, and there is no $E' \in \mathbf{CO}(G)$ such that $E'^+ \supset E^+$. Let $\mathbf{PR}(G)$ and $\mathbf{SST}(G)$ denote respectively the set of preferred and semi-stable extensions of G. An argument $x \in arg(G)$ is *credulously accepted* in a set S of extensions, denoted $S \models_c x$, if there is $E \in S$ such that $x \in E$. Argument x is *skeptically accepted* in S, denoted $S \models_s x$, if $x \in E$ for all $E \in S$.

For an AF G, let $pr(G) := \{\neg x \vee \neg y \mid xy \in att(G)\} \cup \Big\{a_x \leftrightarrow \bigvee_{yx \in att(G)} y,\ x \leftrightarrow \bigwedge_{yx \in att(G)} a_y \;\Big|\; x \in arg(G)\Big\}$, $sst(G) := pr(G) \cup \Big\{r_x \to x \vee \bigvee_{yx \in att(G)} y \;\Big|\; x \in arg(G)\Big\}$.

Proposition 2. *For any AF G, $\mathbf{PR}(G) = \{I \cap arg(G) \mid I \in models(pr(G), arg(G))\}$, and $\mathbf{SST}(G) = \{I \cap arg(G) \mid I \in models(sst(G), \{r_x \mid x \in arg(G)\})$.*

Credulous acceptance of $x \in arg(G)$ in $\mathbf{PR}(G)$ can be tested by a call to $\text{AnswerQuery}(pr(G), arg(G), x)$; note that the query is an objective literal, and therefore Corollary 1 applies. Skeptical acceptance of $x \in arg(G)$ in $\mathbf{PR}(G)$ can be tested by a call to $\text{AnswerQuery}(pr(G), arg(G), \neg x)$, and by properly reverting the answer; note that $\overline{\neg x} = x$ is an objective literal, and therefore Corollary 2 applies. Similarly, credulous acceptance of $x \in arg(G)$ in $\mathbf{SST}(G)$ can be tested by a call to $\text{AnswerQuery}(sst(G), \{r_x \mid x \in arg(G)\}, x)$, while skeptical acceptance of $x \in arg(G)$ in $\mathbf{SST}(G)$ can be tested by a call to $\text{AnswerQuery}(sst(G), \{r_x \mid x \in arg(G)\}, \neg x)$, and by properly reverting the answer.

6 Implementation

Algorithm 1 has been implemented in the solver CIRCUMSCRIPTINO [Alviano, 2017b]. Model searches (i.e., function *solve*) are performed by means of the SAT solver GLUCOSE 4.1 [Audemard and Simon, 2009], with its default configuration. More in detail, two instances of the SAT solver are used, one for the calls performed at line 3, and one for the calls performed at line 5. Specifically, the first solver processes the theory $\Gamma \cup \{q\}$ and the set of assumption literals $O \cap I'$; clause $\bigvee O \setminus I'$ is added to the theory at line 9. The second solver, instead, processes the theory $\Gamma \cup \{\overline{q}\} \cup \{\bigvee O \setminus I\}$ and the set of assumption literals $O \cap I$; clause $\bigvee O \setminus I$ is not removed after completing the model search, as the next call either adds $\bigvee O \setminus I''$ for $I'' \supset I$ (if $I' \neq \emptyset$), or is preceded by the addition of $\bigvee O \setminus I'$ at line 9 for I' such that $O \cap I' \supset O \cap I$.

Possibly, the complete algorithm is aided by the incomplete algorithm. However, checking $solve(\Gamma \cup \{\overline{q}\} \cup \{\ell\}) = \bot$ and $solve(\Gamma \cup \{q\} \cup \{\ell\}) \neq \bot$ for all $\ell \in O$ may require significant computational resources. Hence, a weaker version of the algorithm is implemented by checking $solve(\Gamma \cup \{q\} \cup \{\ell\}) \neq \bot$ only for those literals $\ell \in O$ such that $models(\Gamma\{\overline{q}\} \cup \{\ell\}) = \emptyset$ is easily detectable; intuitively, ℓ is considered only if $\overline{\ell}$ is derived by (unit) propagation on $\Gamma \cup \{\overline{q}\}$. For example, if Γ is $\{q \vee a, q \vee \neg a \vee \neg b\}$, a and $\neg b$ are derived by (unit) propagation on $\Gamma \cup \{\overline{q}\}$; if O is $\{a, b\}$, then $solve(\Gamma \cup \{q\} \cup \{b\}) \neq \bot$ is checked. On the other hand, if Γ is $\{q \vee a \vee \neg b, q \vee \neg a \vee \neg b\}$, no literal is derived by (unit) propagation on $\Gamma \cup \{\overline{q}\}$, even if $models(\Gamma \cup \{\overline{q}\} \cup \{b\}) = \emptyset$.

Since the underlying theory Γ is modified during the execution of the complete algorithm, it may be the case that some new literal is derived by (unit) propagation. For this reason, the incomplete algorithm can be run either once at the beginning of the computation, or each time I' is reset, giving two different strategies to address query answering.

7 Experiment

The experiment comprises instances from ICCMA'17 related to credulous (DC) and skeptical (DS) acceptance problems for preferred (PR) and semi-stable (SST) semantics, and was run on an Intel Xeon 2.4 GHz with 16 GB of memory, and time and memory were limited to 10 minutes and 15 GB, respectively. Each benchmark comprises 350 testcases. The new version of CIRCUMSCRIPTINO is used by PYGLAF, and the execution time of the full pipeline is measured.

A cactus plot of the overall performance is shown in Figure 1. The best performance is reached by PYGLAF with the incomplete algorithm disabled. Specifically, it solved 71 instances more than ARGSEMSAT and 127 more than CEGARTIX. The plot also highlights an initial overhead of PYGLAF with respect to ARGSEMSAT due to the construction of the propositional theory, which is implemented in Python.

The scatter plots in Figure 2 provide an instance by instance comparison in terms of execution time. The comparison with ARGSEMSAT and CEGARTIX shows that there are several instances for which PYGLAF is not the fastest system, but it is still able to provide the correct answer within the allotted resources. Distinguishing on the different computational problems, the main advantage of PYGLAF emerges on the semi-stable semantics, which stimulates all features of Algorithm 1. Figure 2 also compares the different variants of PYGLAF, highlighting that the addition of the incomplete algorithm deteriorates the performance of the system for skeptical acceptance over preferred semantics, and otherwise it does not provide any influence.

Finally, a summary of the experiment is reported on Table 1, where the number of instances solved by each tested system within the allotted resources is shown. The table confirms that the main advantage of Algorithm 1 emerges on the semi-stable semantics, and that the addition of the incomplete algorithm deteriorated the performance of the system only for skeptical acceptance over preferred extensions.

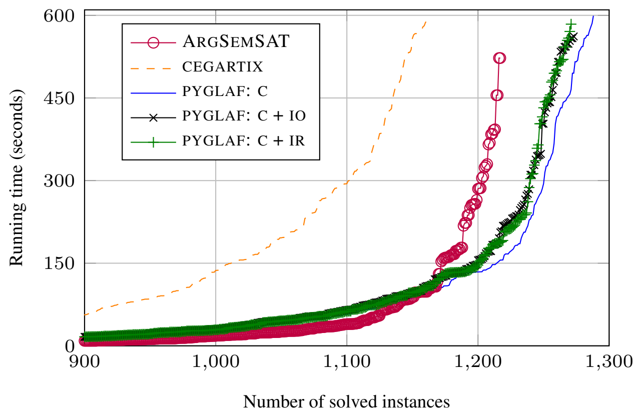

Figure 1: Solved instances within a time budget; C is the complete algorithm, IO is the incomplete algorithm used once, and IR is the incomplete algorithm reiterated.

Solver	DC PR	DC SST	DS PR	DS SST	Overall
ARGSEMSAT	**331**	291	306	289	1217
	(11.3)	(24.2)	(19.3)	(20.0)	(18.5)
CEGARTIX	322	311	264	264	1161
	(33.9)	(54.8)	(63.8)	(62.0)	(52.7)
PYGLAF: C	324	323	**316**	**325**	**1288**
	(31.3)	(35.0)	**(36.4)**	**(34.7)**	**(34.4)**
PYGLAF: C+IO	326	324	298	325	1273
	(31.2)	(33.4)	(35.3)	(35.1)	(33.7)
PYGLAF: C+IR	326	**324**	296	**325**	1271
	(30.6)	**(32.7)**	(32.1)	**(34.7)**	(32.5)

Table 1: Solved instances and average execution time (in seconds, shown in parenthesis), and best performance emphasized in bold; C is the complete algorithm, IO is the incomplete algorithm used once, and IR is the incomplete algorithm reiterated.

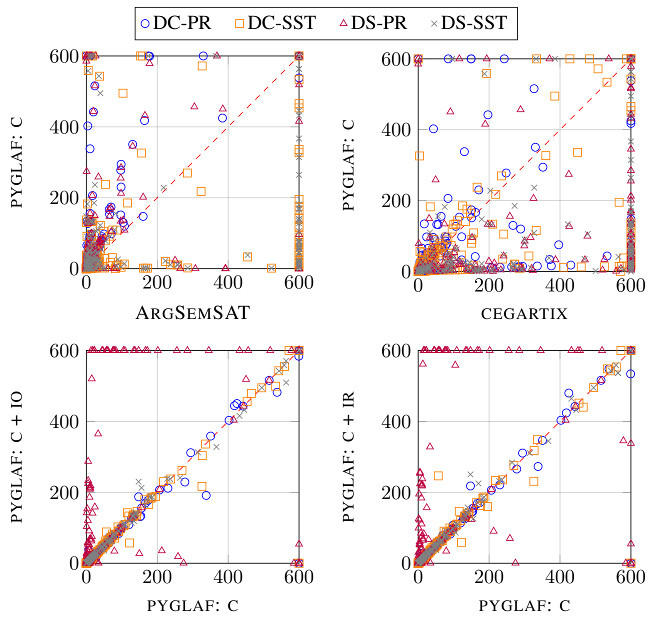

Figure 2: Instance by instance comparison on execution time; C is the complete algorithm, IO is the incomplete algorithm used once, and IR is the incomplete algorithm reiterated.

8 Related Work

The notion of query answering considered in this paper is the one addressed by ASPRIN [Romero *et al.*, 2016], a system for knowledge bases subject to arbitrary preference relations, where knowledge bases and preference relations are expressed by means of answer set programs. Four different algorithms were implemented in ASPRIN, all of them based on the enumeration of optimal models, or on the enumeration of models followed by an optimality check. Hence, those algorithms are different from the algorithms proposed in this paper, whose main characteristic is essentially to not focus on optimal models. Answer set programming was also used to implement circumscription [Oikarinen and Janhunen, 2005] and argumentation frameworks [Egly *et al.*, 2010].

ARGSEMSAT searches for optimal models of a propositional theory encoding the target semantics until one satisfies the query [Cerutti *et al.*, 2013], and CEGARTIX searches for an optimal model of the theory extended with the query and being an optimal model of the original theory [Dvorák *et al.*, 2014]. Hence, again the optimality requirement is the main difference with the algorithms proposed in this paper.

CEGARTIX also implements some semantic shortcuts, as for example for skeptical acceptance over preferred extensions the system first checks the existence of a complete extension attacking the query argument, which is sufficient to conclude the existence of a preferred extension violating the query. Indeed, in this case any extension being more preferred must also attack the query argument, which therefore cannot be part of the extension. Such a shortcut is obtained and generalized by the incomplete algorithm introduced in Section 4, as in fact arguments attacking the query argument are unit-implied at line 1, possibly among other arguments. On the other hand, the shortcut of CEGARTIX only requires a single consistency check, which may be convenient in some cases.

9 Conclusion

Querying models of circumscribed propositional theories is a computational task in the second level of the polynomial hierarchy, and therefore naturally addressed by procedures involving two SAT solvers. However, the two SAT solvers can be combined in several ways, and previous techniques in the literature focused on the computation of optimal models. On the other hand, the complete algorithm proposed in this paper does not enforce the search of optimal models, but focuses on models satisfying the query and (counter)models not satisfying the query. Empirical evidence of the performance gain of the proposed approach is given. The same empirical analysis highlights that the incomplete algorithm instead deteriorates the performance of the complete algorithm for skeptical acceptance over preferred extensions. Such a behavior suggests that the incomplete algorithm has to be run as a heuristic approach, and therefore subject to some restriction, as for example within a budget on the number of conflicts. The development of a heuristic strategy for limiting the execution of the incomplete algorithm is left as future work.

Acknowledgments

This work was partially supported by the POR CALABRIA FESR 2014-2020 project “DLV Large Scale” (CUP J28C17000220006), by the EU H2020 PON I&C 2014-2020 project “S2BDW” (CUP B28I17000250008), and by GNCS-INdAM.

References

[Alviano and Dodaro, 2016] Mario Alviano and Carmine Dodaro. Anytime answer set optimization via unsatisfiable core shrinking. *TPLP*, 16(5-6):533–551, 2016.

[Alviano and Dodaro, 2017] Mario Alviano and Carmine Dodaro. Unsatisfiable core shrinking for anytime answer set optimization. In Carles Sierra, editor, *IJCAI 2017, Melbourne, Australia, August 19-25, 2017*, pages 4781–4785, 2017.

[Alviano *et al.*, 2013] Mario Alviano, Carmine Dodaro, Wolfgang Faber, Nicola Leone, and Francesco Ricca. WASP: A native ASP solver based on constraint learning. In Pedro Cabalar and Tran Cao Son, editors, *LPNMR 2013, Corunna, Spain, September 15-19, 2013. Proceedings*, volume 8148 of *LNCS*, pages 54–66. Springer, 2013.

[Alviano *et al.*, 2014] Mario Alviano, Carmine Dodaro, and Francesco Ricca. Anytime computation of cautious consequences in answer set programming. *TPLP*, 14(4-5):755–770, 2014.

[Alviano *et al.*, 2015a] Mario Alviano, Carmine Dodaro, Nicola Leone, and Francesco Ricca. Advances in WASP. In Francesco Calimeri, Giovambattista Ianni, and Miroslaw Truszczynski, editors, *LPNMR 2015, Lexington, KY, USA, September 27-30, 2015. Proceedings*, volume 9345 of *LNCS*, pages 40–54. Springer, 2015.

[Alviano *et al.*, 2015b] Mario Alviano, Carmine Dodaro, and Francesco Ricca. A maxsat algorithm using cardinality constraints of bounded size. In Qiang Yang and Michael Wooldridge, editors, *IJCAI 2015, Buenos Aires, Argentina, July 25-31, 2015*, pages 2677–2683. AAAI Press, 2015.

[Alviano *et al.*, 2018] Mario Alviano, Carmine Dodaro, Matti Järvisalo, Marco Maratea, and Alessandro Previti. Cautious reasoning in ASP via minimal models and unsatisfiable cores. *CoRR*, abs/1804.08480, 2018.

[Alviano, 2017a] Mario Alviano. The ingredients of the argumentation reasoner pyglaf: Python, circumscription, and glucose to taste. In Marco Maratea and Ivan Serina, editors, *RCRA 2017, Bari, Italy, November 14-15, 2017*, volume 2011 of *CEUR Workshop Proceedings*, pages 1–16. CEUR-WS.org, 2017.

[Alviano, 2017b] Mario Alviano. Model enumeration in propositional circumscription via unsatisfiable core analysis. *TPLP*, 17(5-6):708–725, 2017.

[Alviano, 2017c] Mario Alviano. The pyglaf argumentation reasoner. In Ricardo Rocha, Tran Cao Son, Christopher Mears, and Neda Saeedloei, editors, *ICLP 2017, August 28 to September 1, 2017, Melbourne, Australia*, volume 58 of *OASICS*, pages 2:1–2:3. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2017.

[Audemard and Simon, 2009] Gilles Audemard and Laurent Simon. Predicting learnt clauses quality in modern SAT solvers. In Craig Boutilier, editor, *IJCAI 2009, Pasadena, California, USA, July 11-17, 2009*, pages 399–404, 2009.

[Cerutti *et al.*, 2013] Federico Cerutti, Paul E. Dunne, Massimiliano Giacomin, and Mauro Vallati. Computing preferred extensions in abstract argumentation: A sat-based approach. In Elizabeth Black, Sanjay Modgil, and Nir Oren, editors, *TAFA 2013, Beijing, China, August 3-5, 2013, Revised Selected papers*, volume 8306 of *LNCS*, pages 176–193. Springer, 2013.

[Dodaro *et al.*, 2011] Carmine Dodaro, Mario Alviano, Wolfgang Faber, Nicola Leone, Francesco Ricca, and Marco Sirianni. The birth of a WASP: preliminary report on a new ASP solver. In Fabio Fioravanti, editor, *CILC 2011, Pescara, Italy, August 31 - September 2, 2011*, volume 810 of *CEUR Workshop Proceedings*, pages 99–113. CEUR-WS.org, 2011.

[Dvorák *et al.*, 2014] Wolfgang Dvorák, Matti Järvisalo, Johannes Peter Wallner, and Stefan Woltran. Complexity-sensitive decision procedures for abstract argumentation. *Artif. Intell.*, 206:53–78, 2014.

[Egly *et al.*, 2010] Uwe Egly, Sarah Alice Gaggl, and Stefan Woltran. Answer-set programming encodings for argumentation frameworks. *Argument & Computation*, 1(2):147–177, 2010.

[Eiter and Gottlob, 1993] Thomas Eiter and Georg Gottlob. Propositional circumscription and extended closed-world reasoning are iip2-complete. *Theor. Comput. Sci.*, 114(2):231–245, 1993.

[Gaggl *et al.*, 2016] Sarah Alice Gaggl, Thomas Linsbichler, Marco Maratea, and Stefan Woltran. Introducing the second international competition on computational models of argumentation. In Matthias Thimm, Federico Cerutti, Hannes Strass, and Mauro Vallati, editors, *SAFA, Potsdam, Germany, September 13, 2016.*, volume 1672 of *CEUR Workshop Proceedings*, pages 4–9. CEUR-WS.org, 2016.

[Jannach *et al.*, 2016] Dietmar Jannach, Thomas Schmitz, and Kostyantyn M. Shchekotykhin. Parallel model-based diagnosis on multi-core computers. *J. Artif. Intell. Res. (JAIR)*, 55:835–887, 2016.

[Junker, 2004] Ulrich Junker. QUICKXPLAIN: preferred explanations and relaxations for over-constrained problems. In Deborah L. McGuinness and George Ferguson, editors, *Proceedings of the Nineteenth National Conference on Artificial Intelligence, Sixteenth Conference on Innovative Applications of Artificial Intelligence, July 25-29, 2004, San Jose, California, USA*, pages 167–172. AAAI Press / The MIT Press, 2004.

[Lifschitz, 1986] Vladimir Lifschitz. On the satisfiability of circumscription. *Artif. Intell.*, 28(1):17–27, 1986.

[Marques-Silva *et al.*, 2013] João Marques-Silva, Federico Heras, Mikolás Janota, Alessandro Previti, and Anton Belov. On computing minimal correction subsets. In Francesca Rossi, editor, *IJCAI 2013, Beijing, China, August 3-9, 2013*, pages 615–622. IJCAI/AAAI, 2013.

[McCarthy, 1980] John McCarthy. Circumscription - A form of non-monotonic reasoning. *Artif. Intell.*, 13(1-2):27–39, 1980.

[Mencía *et al.*, 2015] Carlos Mencía, Alessandro Previti, and João Marques-Silva. Literal-based MCS extraction. In Qiang Yang and Michael Wooldridge, editors, *IJCAI 2015, Buenos Aires, Argentina, July 25-31, 2015*, pages 1973–1979. AAAI Press, 2015.

[Morgado *et al.*, 2013] António Morgado, Federico Heras, Mark H. Liffiton, Jordi Planes, and João Marques-Silva. Iterative and core-guided maxsat solving: A survey and assessment. *Constraints*, 18(4):478–534, 2013.

[Oikarinen and Janhunen, 2005] Emilia Oikarinen and Tomi Janhunen. circ2dlp - translating circumscription into disjunctive logic programming. In Chitta Baral, Gianluigi Greco, Nicola Leone, and Giorgio Terracina, editors, *LPNMR 2005, Diamante, Italy, September 5-8, 2005*, volume 3662 of *LNCS*, pages 405–409. Springer, 2005.

[Pereira *et al.*, 1993] Luís Moniz Pereira, Carlos Viegas Damásio, and José Júlio Alferes. Debugging by diagnosing assumptions. In Peter Fritzson, editor, *AADEBUG'93, Linköping, Sweden, May 3-5, 1993, Proceedings*, volume 749 of *LNCS*, pages 58–74. Springer, 1993.

[Romero *et al.*, 2016] Javier Romero, Torsten Schaub, and Philipp Wanko. Computing diverse optimal stable models. In Manuel Carro, Andy King, Neda Saeedloei, and Marina De Vos, editors, *ICLP 2016 TCs, October 16-21, 2016, New York City, USA*, volume 52 of *OASICS*, pages 3:1–3:14. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2016.