TLP **17** (5–6): 708–725, 2017. © Cambridge University Press 2017
doi:10.1017/S1471068417000278 First published online 22 August 2017

*Model enumeration in propositional circumscription via unsatisfiable core analysis**

MARIO ALVIANO

Department of Mathematics and Computer Science, University of Calabria, Rende, Italy
(*e-mail:* alviano@mat.unical.it)

submitted 6 July 2017; revised 20 July 2017; accepted 21 July 2017

Abstract

Many practical problems are characterized by a preference relation over admissible solutions, where preferred solutions are minimal in some sense. For example, a preferred diagnosis usually comprises a minimal set of reasons that is sufficient to cause the observed anomaly. Alternatively, a minimal correction subset comprises a minimal set of reasons whose deletion is sufficient to eliminate the observed anomaly. Circumscription formalizes such preference relations by associating propositional theories with minimal models. The resulting enumeration problem is addressed here by means of a new algorithm taking advantage of unsatisfiable core analysis. Empirical evidence of the efficiency of the algorithm is given by comparing the performance of the resulting solver, CIRCUMSCRIPTINO, with HCLASP, CAMUS_MCS, LBX and MCSLS on the enumeration of minimal models for problems originating from practical applications.

KEYWORDS: Circumscription, minimal model enumeration, minimal correction subsets, minimal intervention strategies, unsatisfiable core analysis

1 Introduction

Circumscription (McCarthy 1980) is a nonmonotonic logic formalizing common sense reasoning by means of a second-order semantics. Intuitively, circumscription allows to express that some things are as expected unless otherwise specified, a property that cannot be expressed in monotonic languages such as first order logic. More specifically, the idea of circumscription is to minimize the extension of some predicates. In the special case of propositional theories, which are the focus of the present paper, the simplest form of circumscription essentially selects subset minimal models. In the form introduced by Lifschitz (1986), instead, some atoms are used to group interpretations, and other atoms are subject to minimization.

* This research has been partially supported by the Italian Ministry for Economic Development (MISE) under project "PIUCultura – Paradigmi Innovativi per l'Utilizzo della Cultura" (no. F/020016/01-02/X27), and under project "Smarter Solutions in the Big Data World (S2BDW)" (no. F/050389/01-03/X32) funded within the call "HORIZON2020" PON I&C 2014-2020, and by Gruppo Nazionale per il Calcolo Scientifico (GNCS-INdAM).

https://doi.org/10.1017/S1471068417000278 Published online by Cambridge University Press

Many practical problems are characterized by a preference relation over admissible solutions. For example, when analyzing a faulty system, several diagnoses are usually possible, and the debugging process can be improved by focusing on those diagnoses comprising a minimal set of reasons that is sufficient to cause the observed anomaly (Pereira *et al.* 1993; Jannach *et al.* 2016). In this case, the preference relation is given by the subset containment relation. The same preference relation can be used to recover the faulty system: a correction subset is a set of reasons whose deletion is sufficient to eliminate the observed anomaly, and intuitively the debugging process has to focus on *minimal correction subsets* (Junker 2004; Marques-Silva *et al.* 2013). It turns out that such practical problems have a natural representation in the framework of circumscription.

The notion of a minimal correction subset received particular attention from the scientific community in recent years, in particular in the context of propositional logic (Marques-Silva and Previti 2014; Mencía *et al.* 2015, 2016). In this case, a minimal correction subset of an unsatisfiable propositional theory T is a subset S of T such that $T \setminus S$ is satisfiable, while $T \setminus S'$ is still unsatisfiable for all $S' \subset S$. A natural representation of this problem in circumscription is obtained by replacing each formula $\phi \in T$ with $\phi \vee x_\phi$, where x_ϕ is a fresh variable subject to minimization. Models of the obtained circumscribed theory represent minimal correction subsets of T: in fact, the minimal correction subset associated with a model I is $\{\phi \in T \mid x_\phi \in I\}$.

It is therefore not a surprise that the enumeration of models satisfying some minimality condition has been already addressed in the literature (Liffiton and Sakallah 2008; Kaminski *et al.* 2013; Faber *et al.* 2016). In particular, Kaminski *et al.* focused on the computation of minimal intervention strategies in logical signaling networks representing biological scenarios; among the several techniques they presented, the most efficient takes advantage of the domain heuristics supported by the solver HCLASP. Liffiton and Sakallah instead focused on the enumeration of minimal correction subsets in order to subsequently compute minimal unsatisfiable subsets (or minimal unsatisfiable cores); the first computational task is accomplished by the solver CAMUS_MCS by enumerating models of size n that do not contain previously reported models, for increasing values of n. Finally, Faber *et al.* presented a general purpose algorithm to achieve minimal model enumeration by iteratively enumerating cardinality minimal models of the input theory that do not contain previously reported models; enumeration of cardinality minimal models is usually achieved by computing a first model of minimal cardinality, and then by enumerating all models of that size.

Cardinality minimal models of propositional theories are solutions of the computational problem known as MaxSAT. For problems originating from practical applications, the most efficient MaxSAT algorithms are based on unsatisfiable core analysis (Morgado *et al.* 2013); among them are OLL (Andres *et al.* 2012) and ONE (Alviano *et al.* 2015). In particular, ONE can be seen as a simplification of OLL, and essentially modifies the processed theory by enforcing the satisfaction of all but one formulas in the analyzed unsatisfiable core. A natural question is therefore whether these algorithms can be directly adapted to achieve minimal model enumeration. In

https://doi.org/10.1017/S1471068417000278 Published online by Cambridge University Press

particular, such an algorithm would not rely on any global condition on the size of the computed models, which is an advantage because global constraints of this kind may exponentially deteriorate the performance of a solver (Bacchus and Narodytska 2014).

This paper provides a positive answer to the above question: an algorithm for the enumeration of models of a circumscribed theory is presented; the algorithm takes advantage of the unsatisfiable core analysis provided by ONE, and enumerates models sorted by size. This is a property shared with the approaches proposed by Liffiton and Sakallah, and Faber *et al.* However, differently from them, and like the strategy adopted by Kaminski *et al.*, the new algorithm smoothly runs on an incremental solver, meaning that new formulas are introduced during its execution, but none of them need to be subsequently removed.

A prototype solver implementing the proposed algorithm is also presented. It is called CIRCUMSCRIPTINO, and relies on the incremental SAT solver GLUCOSE (Audemard and Simon 2009) for computing models and unsatisfiable cores of the processed propositional theory. The prototype is evaluated empirically on four test cases representing the logical signaling networks analyzed by Kaminski *et al.*: in order to enumerate minimal intervention strategies with CIRCUMSCRIPTINO, the instances processed by HCLASP are translated into classical propositional theories by means of the convenient tool LP2SAT (Janhunen and Niemelä 2011). Three of the tested instances are solved by HCLASP and CIRCUMSCRIPTINO in a few seconds. Many other testcases are obtained from the SAT Solver Competitions (Järvisalo *et al.* 2012). In particular, the prototype is evaluated on the enumeration of minimal correction subsets for unsatisfiable instances of the MUS Special track. On these instances, the performance of CIRCUMSCRIPTINO is superior than those of the specialized solvers CAMUS_MCS (Liffiton and Sakallah 2008), LBX (Mencía *et al.* 2015) and MCSLS (Marques-Silva *et al.* 2013), as well as of HCLASP (Gebser *et al.* 2013).

2 Background

Let $\mathscr{A}$ be a fixed, countable set of *atoms* including $\bot$. 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 $p \in \mathscr{A}$; for a set L of literals, let $\overline{L}$ be $\{\overline{\ell} \mid \ell \in L\}$. Moreover, for a set L of literals and a set A of atoms, the *restriction* of L to symbols in A is $L|_A := L \cap (A \cup \overline{A})$.

Formulas are defined as usual by combining atoms and the connectives $\neg$, $\wedge$, $\vee$, $\rightarrow$. In addition, a formula can be a *cardinality constraint* of the following form:

$$\ell_1 + \cdots + \ell_n \geqslant k \tag{2.1}$$

where $\ell_1, \ldots, \ell_n$ are literals, $n \geqslant 1$ and $k \geqslant 0$. A *theory* is a set T of formulas including $\neg\bot$; the set of atoms occurring in T is denoted by $atoms(T)$.

https://doi.org/10.1017/S1471068417000278 Published online by Cambridge University Press

Example 1
The following theories will be used as running examples:

$$T_1 := \{\neg\bot, \quad a \vee x_0, \quad \neg a \vee b \vee x_1, \quad \neg a \vee \neg b \vee x_2\};$$
$$T_2 := T_1 \cup \{r \rightarrow x_0 \wedge x_1 \wedge x_2\};$$
$$T_3 := T_1 \cup \{\neg x_0 + \neg x_1 + \neg x_2 + y_1 + y_2 \geqslant 2, \quad y_2 \rightarrow y_1\}.$$

In particular, T_3 will be used in Section 3 as an example of unsatisfiable core analysis.

An *assignment* is a set A of literals such that $A \cap \overline{A} = \emptyset$. An *interpretation* for a theory T is an assignment I such that $(I \cup \overline{I}) \cap \mathscr{A} = atoms(T)$. Relation $\models$ is defined as usual: for $p \in \mathscr{A}$, $I \models p$ 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 $I \models \psi$, and $I \models \phi \rightarrow \psi$ if $I \models \psi$ whenever $I \models \phi$; for ϕ of the form (2.1), $I \models \phi$ if $|I \cap \{\ell_1, \ldots, \ell_n\}| \geqslant k$; for a theory T, $I \models T$ if $I \models \phi$ for all $\phi \in T$. I is a *model* of a theory T if $I \models T$. Let $models(T)$ denote the set of models of T. (Models will be also represented by the set of their atoms, as their negative literals are implicit.)

Example 2 (*Continuing Example 1*)
T_1 has 16 models, including the following: $\{x_0\}$, $\{x_0, b\}$, $\{x_1, a\}$, $\{x_2, a, b\}$, $\{x_0, x_1\}$, $\{x_0, x_1, a\}$, and $\{x_0, x_1, b\}$. These are also models of T_2 (where r is false); T_2 additionally admits, for any $X \subseteq \{a, b\}$, $\{x_0, x_1, x_2, r\} \cup X$. Regarding T_3, variables y_1, y_2 can be used to constrain the number of false atoms among $\{x_0, x_1, x_2\}$. For example, models extending the assignment $\{\neg y_1, \neg y_2\}$ must assign false to at least two atoms in $\{x_0, x_1, x_2\}$; these models are precisely $\{x_0\}$, $\{x_0, b\}$, $\{x_1, a\}$, and $\{x_2, a, b\}$. Finally, note that T_1, T_2, and T_3 have no models extending the assignment $\{\neg x_0, \neg x_1, \neg x_2\}$.

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

Example 3 (*Continuing Example 2*)
Let P be $\{x_0, x_1, x_2\}$, and $Z \subseteq \{a, b\}$. $CIRC(T_1, P, Z)$ and $CIRC(T_2, P, Z \cup \{r\})$ are $\{\{x_0\}, \{x_0, b\}, \{x_1, a\}, \{x_2, a, b\}\}$, while $CIRC(T_2, P, Z)$ additionally includes $\{x_0, x_1, x_2, r\} \cup X$, for all $X \subseteq \{a, b\}$. Indeed, if r is not irrelevant, then $I \not\leqslant^{PZ} \{x_0, x_1, x_2, r\} \cup X$ for all $I \in \{\{x_0\}, \{x_0, b\}, \{x_1, a\}, \{x_2, a, b\}\}$, and all $X \subseteq \{a, b\}$. Regarding T_3, note that $CIRC(T_1, \{y_1, y_2\}, Z)$ is again $\{\{x_0\}, \{x_0, b\}, \{x_1, a\}, \{x_2, a, b\}\}$, for any $Z \subseteq \{a, b\}$.

3 Model enumeration

The computational problem addressed in this paper is the enumeration of models of a circumscribed theory, that is, the enumeration of $CIRC(T, P, Z)$. The proposed

https://doi.org/10.1017/S1471068417000278 Published online by Cambridge University Press

Algorithm 1: Model enumeration for $CIRC(T,P,Z)$

1 $O := P$; $V := atoms(T) \setminus \{\bot\}$; $R := V \setminus (P \cup Z)$;
2 **repeat**
3 $(sat, I, -, C) := solve(T, \overline{O})$; // try to falsify all objective literals
4 **if** *sat* **then** // minimal model found
5 $enumerate(T, V, I|_{P \cup R} \cup (O \setminus P))$; // fix P, R, and disable new constraints
6 $T := T \cup \{\bigwedge I|_R \rightarrow \bigvee \overline{P \cap I}\}$; // add blocking clause
7 **else if** $C \neq \emptyset$ **then** // unsatisfiable core analysis
8 Let C be $\{\neg x_0, \ldots, \neg x_n\}$ ($n \geqslant 0$), and $y_1, \ldots, y_n$ be fresh variables;
9 $O := (O \setminus \{x_0, \ldots, x_n\}) \cup \{y_1, \ldots, y_n\}$;
10 $T := T \cup \{\neg x_0 + \cdots + \neg x_n + y_1 + \cdots + y_n \geqslant n\} \cup \{y_i \rightarrow y_{i-1} \mid i \in [2..n]\}$;
11 **until not** *sat* **and** $C = \emptyset$ // empty unsatisfiable core implies no more models;

algorithm takes advantage of modern solvers for checking satisfiability of propositional theories. This is achieved by means of function *solve*, whose input is a theory T and a set A of literals called *assumptions*. The function searches for a model I of T such that $A \subseteq I$. If such an I exists, tuple $(true, I, B, -)$ is returned, where $B \subseteq I$ is the set of branching literals used to compute I. Otherwise, a tuple $(false, -, -, C)$ is returned, where $C \subseteq A$ is such that $T \cup \{\ell \mid \ell \in C\}$ has no models; in this case, C is called *unsatisfiable core*.

Example 4 (*Continuing Example 3*)
The result of $solve(T_1, \{\neg x_0, \neg x_1, \neg x_2\})$ is $(false, -, -, \{\neg x_0, \neg x_1, \neg x_2\})$, that is, set $\{\neg x_0, \neg x_1, \neg x_2\}$ is an unsatisfiable core. On the other hand, $solve(T_3, \{\neg y_1, \neg y_2\})$ returns $(true, I, B, -)$, where $I \in \{\{x_0\}, \{x_0, b\}, \{x_1, a\}, \{x_2, a, b\}\}$ (and $B \subseteq I$).

Algorithm 1 implements model enumeration for $CIRC(T,P,Z)$. The following sets are used by the algorithm: O for the objective literals, that is, those to minimize, initially P; V for the visible atoms, that is, those in the input theory T; R for the (other) relevant atoms, that is, those not in $P \cup Z$. The algorithm iteratively searches for a model of T falsifying all objective literals. If a nonempty unsatisfiable core C is returned, it is processed according to the ONE algorithm (lines 8–10): objective literals in the unsatisfiable core are replaced by $|C| - 1$ new objective literals (not already occurring in T), and the theory T is extended with new formulas enforcing the truth of at least $|C| - i$ literals in C whenever the ith new objective literal is false. On the other hand, if a model is found, say I, it is guaranteed to be minimal with respect to P. In this case, the interpretation of atoms in P and R is fixed, and all formulas introduced in line 10 are satisfied by assuming the truth of all objective literals introduced in line 9. All models extending these assumptions are then enumerated, for example, by means of the polyspace algorithm introduced by Gebser *et al.* (2007), here given in terms of assumptions (Alviano and Dodaro

https://doi.org/10.1017/S1471068417000278 Published online by Cambridge University Press

Procedure enumerate(T, V, A)

1 $push(A, \neg\bot)$; $F := \emptyset$; // initialize assumptions and flipped literals
2 **while** $top(A) \neq \bot$ **do** // there are still assumptions to be flipped
3 $(sat, I, B, C) := solve(T, A)$; // search $I \in models(T)$ such that $A \subseteq I$
4 **if** *sat* **then** // found I using branching literals B
5 **print** $I \cap V$; // report model
6 **for** $\ell \in B \setminus A$ **do** $push(A, \ell)$; // extend A with new branching literals
7 **else** // found unsatisfiable core $C \subseteq A$
8 **while** $top(A) \neq \neg\bot$ ***and*** $top(A) \notin C$ **do** $F := F \setminus \{pop(A)\}$; // backjump
9 **while** $top(A) \in F$ **do** $F := F \setminus \{pop(A)\}$; // remove flipped assumptions
10 $push(A, \overline{pop(A)})$; $F := F \cup \{top(A)\}$; // flip top assumption

2016a). After that, a blocking clause is added to T, so to discard all interpretations J such that $I \leqslant^{PZ} J$ (including I itself): intuitively, any model J such that $J|_R = I|_R$ is forced to falsify at least one atom of P that is interpreted as true by I. The algorithm terminates as soon as an empty unsatisfiable core is returned by function *solve*, meaning that all models in $CIRC(T, P, Z)$ have been computed.

Example 5 (*Continuing Example 4*)
Let T be T_1, P be $\{x_0, x_1, x_2\}$, and Z be empty. Algorithm 1 starts by setting O and R respectively to $\{x_0, x_1, x_2\}$ and $\{a, b\}$. The first call to function *solve* returns $(false, -, -, \{\neg x_0, \neg x_1, \neg x_2\})$, and therefore the unsatisfiable core $\{\neg x_0, \neg x_1, \neg x_2\}$ is analyzed (lines 7–10): set O becomes $\{y_1, y_2\}$, where y_1 and y_2 are fresh variables, and T is extended with $\neg x_0 + \neg x_1 + \neg x_2 + y_1 + y_2 \geqslant 2$, and $y_2 \rightarrow y_1$. Note that T is now T_3. The second call to function *solve* then returns $(true, \{y_1, y_2\} \cup I, -, -)$, where $I \in \{\{x_0\}, \{x_0, b\}, \{x_1, a\}, \{x_2, a, b\}\}$. Say that I is $\{x_0\}$; the enumeration procedure is called with assumptions $\{x_0, \neg x_1, \neg x_2, \neg a, \neg b, y_1, y_2\}$ (recall that negative literals are implicit in I, hence $I|_{P \cup R}$ is $\{x_0, \neg x_1, \neg x_2, \neg a, \neg b\}$). In this case, model I is computed again (in linear time with modern solvers), and the enumeration procedure terminates. Theory T is extended with the blocking clause $\neg a \wedge \neg b \rightarrow \neg x_0$, and a new model is computed, say $\{x_0, b\}$. Again, since all atoms are relevant, the enumeration procedure terminates reporting only $\{x_0, b\}$ itself. Theory T is extended with the blocking clause $\neg a \wedge b \rightarrow \neg x_0$, and a new model is computed, say $\{x_1, a\}$. Theory T is extended with the blocking clause $a \wedge \neg b \rightarrow \neg x_1$, and model $\{x_2, a, b\}$ is computed. Finally, the blocking clause $a \wedge b \rightarrow \neg x_2$ is added, and the empty unsatisfiable core is returned by function *solve*. Hence, all models of $CIRC(T_1, \{x_0, x_1, x_2\}, \emptyset)$ were reported, and the algorithm terminates.

For Z being $\{a, b\}$, R is empty and the algorithm behaves differently starting from the call to the enumeration procedure. Indeed, for $I = \{x_0\}$, the assumptions are $\{x_0, \neg x_1, \neg x_2, y_1, y_2\}$, and the procedure reports two models, namely $\{x_0\}$ and $\{x_0, b\}$. Moreover, the blocking clause added to T is $\neg x_0$, so that the next model returned by function *solve* must be either $\{x_1, a\}$ or $\{x_2, a, b\}$. The associated blocking clauses

https://doi.org/10.1017/S1471068417000278 Published online by Cambridge University Press

are $\neg x_1$ and $\neg x_2$, and after adding them, the empty unsatisfiable core is returned by function *solve*, so that the algorithm can terminate.

For T being T_2, and Z being $\{a, b\}$, R is $\{r\}$ and the algorithm behaves differently starting from the second call to function *solve*. Indeed, in this case, the returned model may also be $\{x_0, x_1, x_2, r\} \cup X$, for $X \subseteq \{a, b\}$. Say that I is $\{x_0, x_1, x_2, r\}$; the enumeration procedure is called with assumptions $\{x_0, x_1, x_2, r, y_1, y_2\}$, and models $\{x_0, x_1, x_2, r\} \cup X$, for $X \subseteq \{a, b\}$, are reported. After that, the blocking clause $r \rightarrow \neg x_0 \vee \neg x_1 \vee \neg x_2$ is added to T, so that a model $I \in \{\{x_0\}, \{x_0, b\}, \{x_1, a\}, \{x_2, a, b\}\}$ can be returned by function *solve*. From this point, the algorithm continues as for $CIRC(T_1, \{x_0, x_1, x_2\}, \{a, b\})$, with the only difference that the added blocking clauses are $\neg r \rightarrow \neg x_0$, $\neg r \rightarrow \neg x_1$, and $\neg r \rightarrow \neg x_2$.

3.1 Correctness

The following main theorem is proved in this section.

Theorem 1
Let T be a theory, and P, Z be sets of atoms. Algorithm 1 enumerates all models in $CIRC(T, P, Z)$, and the number of iterations of the repeat-until loop is bounded by $|models(T)| + |P|$.

The above theorem is proved by showing that the algorithm is correct at each iteration: when lines 4–6 are executed, models of the processed theory are either reported or satisfy the added blocking clause; when lines 7–10 are executed, all models are preserved.

First of all, recall that a model is possibly represented as the set of its atoms (i.e., negative literals are ignored). Hence, for sets S, S' of models, we will write $S = S'$ if $\{I \cap \mathscr{A} \mid I \in S\} = \{I \cap \mathscr{A} \mid I \in S'\}$, even if S and S' are models of theories with different atoms. The following lemma states that procedure $enumerate(T, V, A)$ computes all models of T extending the assignment A.

Lemma 1
Let T be a theory, V be a set of atoms, and A be a set of literals. Procedure $enumerate(T, V, A)$ computes $\{I \cap V \mid I \in models(T \cup \{\ell \mid \ell \in A\})\}$.

Proof
The procedure given in this paper extends the one presented by Alviano and Dodaro (2016a) with the possibility of providing in input a set A of assumptions. In order to extend the correctness of the enumeration procedure presented by Alviano and Dodaro, we have only to note that assumptions in A are protected by literal $\neg\bot$ (pushed on line 1 of the procedure), so that they are never flipped or removed by the procedure. All models of T extending the provided assumptions are therefore reported, only printing true atoms among those in V. □

Among the assumptions passed to procedure *enumerate* are the variables introduced by ONE. They are assumed true so to restore the original theory.

https://doi.org/10.1017/S1471068417000278 Published online by Cambridge University Press

Lemma 2
Let T be a theory, and T' be $T \cup \{\neg x_0 + \cdots + \neg x_n + y_1 + \cdots + y_n \geqslant n\} \cup \{y_i \rightarrow y_{i-1} \mid i \in [2..n]\}$, for $n \geqslant 0$. If $y_i \notin atoms(T)$ for $i \in [1..n]$, $models(T) = \{I|_{atoms(T)} \mid I \in models(T' \cup \{y_i \mid i \in [1..n]\})\}$.

Proof
If I is a model of T, then $I \cup \{y_i \mid i \in [1..n]\}$ is a model of $T' \cup \{y_i \mid i \in [1..n]\}$. Moreover, any model J of $T' \cup \{y_i \mid i \in [1..n]\}$ satisfies $J \models T$ (because $T \subseteq T'$). □

Hence, when procedure *enumerate* is invoked on line 5 of Algorithm 1, since the initial assumptions comprise $O \setminus P$, and because of Lemma 2, all models of T extending the assignment $I|_{P \cup R}$ are computed. These models are then discarded by the blocking clause added in line 6, as formalized by the following claim.

Lemma 3
Let I be a model in $CIRC(T, P, Z)$, R be $atoms(T) \setminus (P \cup Z \cup \{\bot\})$, and ϕ be $\bigwedge I|_R \rightarrow \bigvee \overline{P \cap I}$. It holds that $CIRC(T, P, Z) = CIRC(T \cup \{\phi\}, P, Z) \cup models(T \cup \{\ell \mid \ell \in I|_{P \cup R}\})$.

Proof
($\subseteq$) Consider $J \in CIRC(T, P, Z)$. We distinguish two cases:

(1) $J \models \phi$. Hence, $J \in models(T \cup \{\phi\})$. Let $J' \in models(T \cup \{\phi\})$ be such that $J' \leqslant^{PZ} J$. Since $J \in CIRC(T, P, Z)$ by assumption, $J \leqslant^{PZ} J'$ holds, and therefore $J \in CIRC(T \cup \{\phi\}, P, Z)$.
(2) $J \not\models \phi$. Hence, $J \models \bigwedge I|_R$ and $J \not\models \bigvee \overline{P \cap I}$. Note that $I|_R = I|_{atoms(T) \setminus (P \cup Z \cup \{\bot\})} = I|_{\mathscr{A} \setminus (P \cup Z)}$, and therefore $J \models \bigwedge I|_R$ implies $I|_{\mathscr{A} \setminus (P \cup Z)} = J|_{\mathscr{A} \setminus (P \cup Z)}$. Moreover, $J \not\models \bigvee \overline{P \cap I}$ implies $I \cap P \subseteq J \cap P$. From $I|_{\mathscr{A} \setminus (P \cup Z)} = J|_{\mathscr{A} \setminus (P \cup Z)}$ and $I \cap P \subseteq J \cap P$, we have $I \leqslant^{PZ} J$. Since $J \in CIRC(T, P, Z)$ by assumption, $J \leqslant^{PZ} I$ holds. Hence, $J|_{P \cup R} = I|_{P \cup R}$, which implies $J \in models(T \cup \{\ell \mid \ell \in I|_{P \cup R}\})$.

($\supseteq$) We distinguish two cases:

(1) $J \in CIRC(T \cup \{\phi\}, P, Z)$. We have $J \in models(T)$. Let $J' \in models(T)$ be such that $J' \leqslant^{PZ} J$. We shall show that $J' \in models(T \cup \{\phi\})$, which implies $J \leqslant^{PZ} J'$ and therefore $J \in CIRC(T, P, Z)$. $J' \leqslant^{PZ} J$ implies $J'|_{\mathscr{A} \setminus (P \cup Z)} = J|_{\mathscr{A} \setminus (P \cup Z)}$ and $J' \cap P \subseteq J \cap P$. From $J \models \phi$, either $J \not\models \bigwedge I|_R$, or $J \models \bigvee \overline{P \cap I}$. Hence, $J' \not\models \bigwedge I|_R$, or $J' \models \bigvee \overline{P \cap I}$, that is, $J' \models \phi$ and then $J' \in models(T \cup \{\phi\})$.
(2) $J \in models(T \cup \{\ell \mid \ell \in I|_{P \cup R}\})$. Since $J \models \bigwedge I|_{P \cup R}$, the symmetric difference of I and J is a subset of Z, which implies $J \leqslant^{PZ} I$ (as well as $I \leqslant^{PZ} J$). Since $I \in CIRC(T, P, Z)$ by assumption, we can conclude that $J \in CIRC(T, P, Z)$.

This conclude the proof of the lemma. □

The following lemma states that the application of ONE transforms the original problem into an equivalent problem.

https://doi.org/10.1017/S1471068417000278 Published online by Cambridge University Press

Lemma 4
Let T be a theory, and P, Z be sets of atoms. If $\{x_0, \ldots, x_n\} \subseteq P$ $(n \geqslant 0)$ is such that $models(T \cup \{\neg x_i \mid i \in [0..n]\}) = \emptyset$, then $CIRC(T, P, Z) = CIRC(T', P', Z')$, where $T' = T \cup \{\neg x_0 + \cdots + \neg x_n + y_1 + \cdots + y_n \geqslant n\} \cup \{y_i \rightarrow y_{i-1} \mid i \in [2..n]\}$, $P' = (P \setminus \{x_0, \ldots, x_n\}) \cup \{y_1, \ldots, y_n\}$, $Z' = Z \cup \{x_0, \ldots, x_n\}$, and $y_1, \ldots, y_n$ are fresh variables.

Proof
Let $ext(I)$ be $I \cup \{y_i \mid i \in [1..n], |I \cap \{x_0, \ldots, x_n\}| > i\}$, and $red(I)$ be $I|_{atoms(T)}$.

$(\subseteq)$ Let $I \in CIRC(T, P, Z)$. By construction, $ext(I) \models T'$. Let J be such that $J \models T'$ and $J \leqslant^{P'Z'} ext(I)$. In order to have $ext(I) \in CIRC(T', P', Z')$, we shall show that $ext(I) \leqslant^{P'Z'} J$. We have $red(J) \leqslant^{PZ} I$, and combining with $I \in CIRC(T, P, Z)$ we conclude $I \leqslant^{PZ} red(J)$. The previous finally implies $ext(I) \leqslant^{P'Z'} J$, and we are done.

$(\supseteq)$ Let $I \in CIRC(T', P', Z')$. By construction, $red(I) \models T$. Let J be such that $J \models T$ and $J \leqslant^{PZ} red(I)$. In order to have $red(I) \in CIRC(T, P, Z)$, we shall show that $red(I) \leqslant^{PZ} J$. We have $ext(J) \leqslant^{P'Z'} I$, and combining with $I \in CIRC(T', P', Z')$ we conclude $I \leqslant^{P'Z'} ext(J)$. The previous finally implies $red(I) \leqslant^{PZ} J$, and we are done. □

Finally, termination of the algorithm is guaranteed because Algorithm 1 executes lines 7–10 unless there is $I \in CIRC(T, P, Z)$ such that $|P \cap I| = |P| - |O|$; otherwise, lines 4–6 are executed, and at least one model is discarded by the added blocking clause. This argument also provides the desired bound on the iterations of the repeat-until loop.

Proof of Theorem 1
Let R be $atoms(T) \setminus (P \cup Z \cup \{\bot\})$, as in Algorithm 1. Let T_i, O_i $(i \geqslant 0)$ be the values of variables T, O at iteration i of Algorithm 1. Let Z_i be $Z \cup (O_i \setminus P)$. We use induction on i to show the following proposition:

$$\text{If } J \in CIRC(T, P, Z) \text{ and } J \models T_i \text{ then } J \in CIRC(T_i, O_i, Z_i). \tag{3.2}$$

The base case is trivial as $T_0 = T$, $O_0 = P$ and $Z_0 = Z$. Assume the proposition for $i \geqslant 0$ in order to show that it holds for $i + 1$. Let $J \in CIRC(T, P, Z)$ be such that $J \models T_{i+1}$. Note that $J \models T_i$ as well, and therefore $J \in CIRC(T_i, O_i, Z_i)$ because of the induction hypothesis. We now distinguish two cases depending on the outcome of function $solve(T_i, \overline{O_i})$:

(1) T_{i+1} is $T_i \cup \{\bigwedge I|_R \rightarrow \bigvee \overline{P \cap I}\}$, for some model I. Hence, $I \in CIRC(T_i, O_i, Z_i)$ because $I \cap O_i = \emptyset$. Moreover, $O_{i+1} = O_i$ and $Z_{i+1} = Z_i$, so that we can apply Lemma 3 to conclude $J \in CIRC(T_{i+1}, O_{i+1}, Z_{i+1}) \cup models(T_i \cup \{\ell \mid \ell \in I|_{P \cup R}\})$. But $J \notin models(T_i \cup \{\ell \mid \ell \in I|_{P \cup R}\})$ because $J \models T_{i+1}$ by assumption. Hence, $J \in CIRC(T_{i+1}, O_{i+1}, Z_{i+1})$.
(2) T_{i+1} is $T_i \cup \{\neg x_0 + \cdots + \neg x_n + y_1 + \cdots + y_n \geqslant n\} \cup \{y_j \rightarrow y_{j-1} \mid j \in [2..n]\}$, for some unsatisfiable core $\{\neg x_0, \ldots, \neg x_n\}$.Hence, $O_{i+1} := (O_i \setminus \{x_0, \ldots, x_n\}) \cup$

https://doi.org/10.1017/S1471068417000278 Published online by Cambridge University Press

$\{y_1, \ldots, y_n\}$, and we can apply Lemma 4 to conclude $CIRC(T_i, O_i, Z_i) = CIRC(T_{i+1}, O_{i+1}, Z_{i+1})$. Therefore, $J \in CIRC(T_{i+1}, O_{i+1}, Z_{i+1})$.

This completes the proof of (3.2).

We can also note that the number of iterations of Algorithm 1 is bounded by $|models(T)| + |P|$ because for all $i \geqslant 0$ either $|models(T_{i+1})| < |models(T_i)|$ (case 1 above) or $|O_{i+1}| < |O_i|$ (case 2 above). This guarantees termination of the algorithm.

To complete the proof of the theorem, we have only to note that for all $J \in CIRC(T, P, Z)$ such that $J \not\models T_{i+1} \setminus T_i$, $J \in models(T_i \cup \{\ell \mid \ell \in I|_{P \cup R}\})$, and therefore J is printed because of Lemmas 1 and 2. □

4 Implementation and experiments

Algorithm 1 is implemented on top of the SAT solver GLUCOSE-4.0 (Audemard and Simon 2009), which is extended to natively support cardinality constraints as a special case of the implementation of weight constraints described by Gebser *et al.* (2009). The resulting prototype solver is called CIRCUMSCRIPTINO (`http://alviano.com/software/circumscriptino/`). Relevant command line parameters are `-n` and `--circ-wit`, respectively, for limiting the number of models and witnesses to be computed, where a witness of a set A of assumptions is intended as a model I such that $A \subseteq I$. Intuitively, `--circ-wit` is used to limit the number of models computed by procedure *enumerate*. In the special case of `--circ-wit=1`, line 5 of Algorithm 1 is not executed, and model I is directly reported to the user. This is particularly useful for problems where the interpretation of atoms in Z is not particularly important. Moreover, the analysis of unsatisfiable cores is preceded by a progression based shrinking (Alviano and Dodaro 2016b).

The performance of the implemented prototype is compared with CAMUS_MCS-1.0.5 (Liffiton and Sakallah 2008), LBX (Mencía *et al.* 2015), MCSLS (with algorithms ELS and CLD; Marques-Silva *et al.* 2013) and HCLASP-1.1.5 (Gebser *et al.* 2013). CAMUS_MCS, LBX and MCSLS are solvers for the enumeration of minimal correction subsets (more details are provided in Section 5). HCLASP is a branch of CLASP (Gebser *et al.* 2012) introducing domain heuristics; it can enumerate minimal models if atoms of the form `_heuristic(p,false,1)` are introduced for each atom p subject to minimization, and if invoked with the command line parameters `--heuristic=domain --enum-mode=record`.

The experiments comprise two problems, namely the enumeration of minimal intervention strategies and the enumeration of minimal correction subsets. For the first problem, instances representing biological signaling networks are considered (Kaminski *et al.* 2013); these instances are translated into the input format of CIRCUMSCRIPTINO thanks to the tool chain LP2NORMAL-2.27+LP2ATOMIC-1.17+LP2SAT-1.24 (Janhunen and Niemelä 2011). Regarding the second problem, instances from the SAT Solver Competitions (MUS Special Track) are tested (Järvisalo *et al.* 2012). The experiments were run on an Intel Xeon 2.4 GHz with 16GB of memory, and time and memory were limited to 10 min and 15GB, respectively.

https://doi.org/10.1017/S1471068417000278 Published online by Cambridge University Press

Table 1. *Enumeration of minimal intervention strategies: execution time in seconds (* T.O. *for timeout), memory consumption in MB, and number of reported models*

	CIRCUMSCRIPTINO			HCLASP		
Instance	Time	Mem	Models	Time	Mem	Models
EGFR	0.00	0	21	0.00	0	21
EGFR MULTIPLE	0.44	36	83	0.10	15	83
TCR	8.33	16	13,016	2.16	9	13,016
TBH6B	T.O.	95	153,405	T.O.	115	758,887

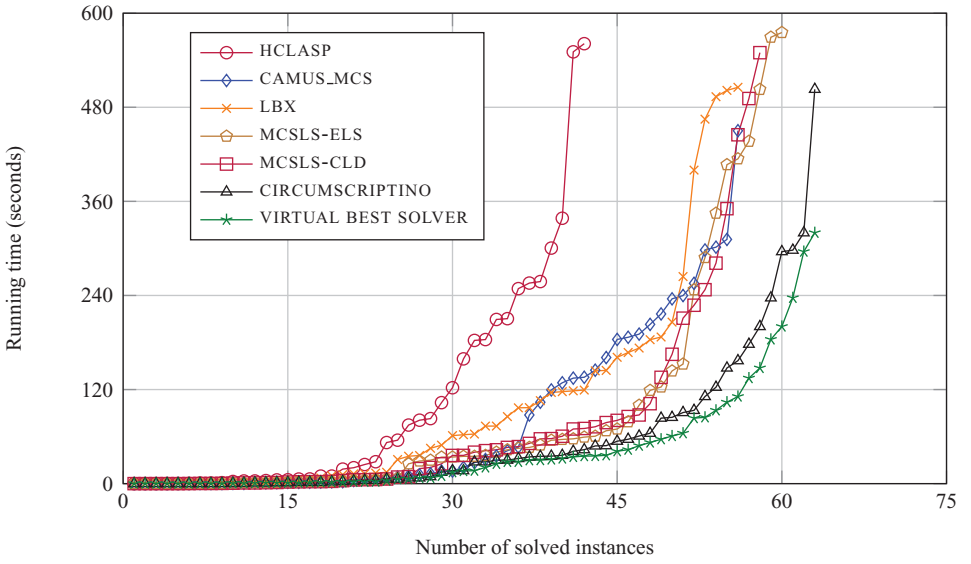


Fig. 1. Enumeration of minimal correction subsets: solved instances within a time budget.

Experimental results on the enumeration of minimal intervention strategies are reported in Table 1. Only four instances are available, one of which cannot be solved by the tested solvers; the other three instances, instead, are solved in less than 10 s. For all instances, memory consumption is very low, and HCLASP appears to be 4–5 times faster than CIRCUMSCRIPTINO. (CAMUS_MCS, LBX and MCSLS are not tested on these instances because their translation is not immediate, and also not the focus of this paper.)

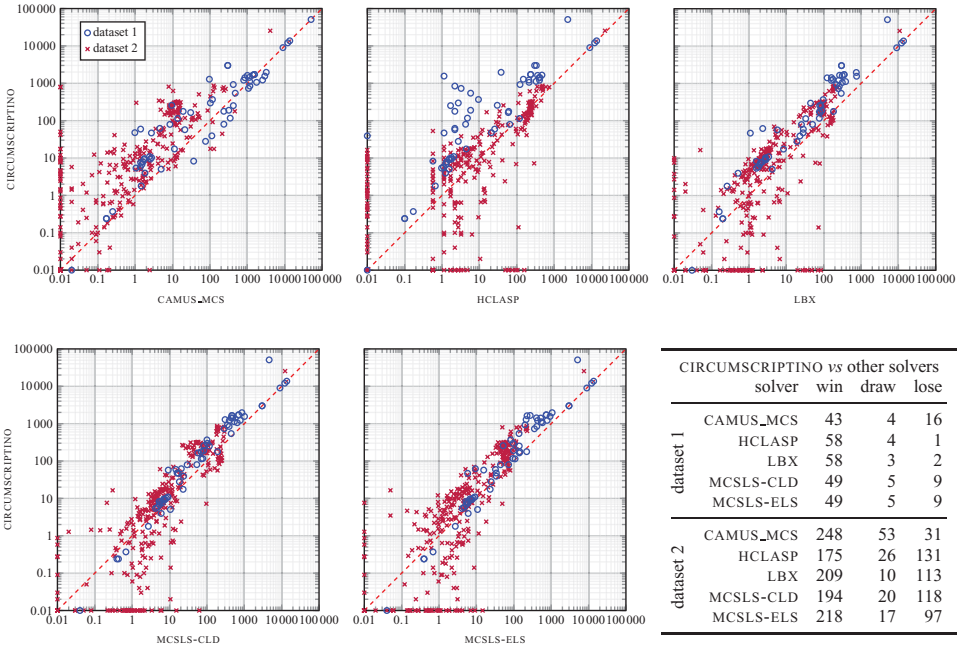
Concerning the enumeration of minimal correction subsets, 395 instances are tested. A cactus plot of the execution time of the tested solvers is reported in Figure 1. The cactus plot also shows the performance of the *virtual best solver*, which is almost aligned with CIRCUMSCRIPTINO: only a few seconds and no solved instances are gained when CIRCUMSCRIPTINO is replaced with the best performant solver in each tested instance. The good result of CIRCUMSCRIPTINO is also confirmed by Table 2, reporting aggregated results for instances for which at least one solver enumerated all models (dataset 1), and for the remaining instances (dataset 2). In particular, results for dataset 1 confirm that CIRCUMSCRIPTINO solves more instances

https://doi.org/10.1017/S1471068417000278 Published online by Cambridge University Press

Table 2. *Enumeration of minimal correction subsets: solved instances, average execution time in seconds on solved instances, average memory consumption in MB, number of reported models, and average velocity*

	Dataset 1 (63 instances)					Dataset 2 (332 instances)		
Solver	Sol	Time	Mem	Models	Vel	Mem	Models	Vel
CIRCUMSCRIPTINO	63	59.3	166	534,293	1,875	598	15,548,418	77
CAMUS_MCS	56	78.6	411	366,909	1,778	1,442	2,508,394	12
HCLASP	42	99.5	356	351,785	666	1,174	13,988,100	69
LBX	56	98.0	146	477,758	749	636	6,604,809	33
MCSLS-CLD	58	75.7	125	489,240	895	612	7,538,687	37
MCSLS-ELS	60	91.0	125	474,296	892	594	4,595,938	23

Instances in the first dataset are those solved by at least one solver, while instances in the second dataset are those for which all solvers ran out of time or memory.



CIRCUMSCRIPTINO *vs* other solvers

	solver	win	draw	lose
dataset 1	CAMUS_MCS	43	4	16
	HCLASP	58	4	1
	LBX	58	3	2
	MCSLS-CLD	49	5	9
	MCSLS-ELS	49	5	9
dataset 2	CAMUS_MCS	248	53	31
	HCLASP	175	26	131
	LBX	209	10	113
	MCSLS-CLD	194	20	118
	MCSLS-ELS	218	17	97

Fig. 2. Enumeration of minimal correction subsets: instance-by-instance comparison in terms of velocity (velocity 0 normalized to 0.01). Instances in the first dataset are those solved by at least one solver, while instances in the second dataset are those for which all solvers ran out of time or memory.

and has the lowest average running time. Concerning dataset 2, since all solvers ran out of time or memory, a comparison is obtained in terms of *velocity*, defined as the number of reported models per second of computation (Liffiton and Sakallah 2008). This metric shows that HCLASP produces models with a velocity close to that of CIRCUMSCRIPTINO, while other solvers are 2–6 times slower. (The same metric is also applied to dataset 1, where however the average is influenced by instances solved in

https://doi.org/10.1017/S1471068417000278 Published online by Cambridge University Press

less than 1 s.) An instance-by-instance comparison in terms of velocity is shown in the plots in Figure 2. Axes are in logarithmic scale, so velocity 0 is normalized to 0.01. It can be observed that in all cases the majority of points is above the diagonal, meaning that the velocity of CIRCUMSCRIPTINO is higher than the velocity of other solvers in the majority of instances. Finally, concerning memory usage, only two memory outs were recorded for CAMUS_MCS and HCLASP on the same instance.

5 Related work

Circumscription formalizes a preference relation over models of logic theories. Such a preference relation is essentially subset minimality. Within this respect, this work is related to many articles in the literature introducing algorithms for computing minimal models. Among them is the OPTSAT algorithm (Giunchiglia and Maratea 2006), which is very similar to the algorithm used by HCLASP (Kaminski *et al.* 2013) in our experiment. Indeed, OPTSAT essentially modifies the standard heuristic of a SAT solver by selecting the atoms subject to minimization as first branching literals, so to force the search procedure to return a minimal model. The difference with the approach implemented by HCLASP is that OPTSAT fixes an order for the atoms subject to minimization, while the heuristic of HCLASP can select any of these atoms; indeed, the only constraint that the heuristic of HCLASP has to satisfy is that all atoms subject to minimization have to be assigned before branching on any other atom.

The main similarity between OPTSAT, HCLASP and CIRCUMSCRIPTINO is that the search starts by trying to falsifying all atoms subject to minimization. However, as soon as no model falsifying all these atoms exists, the algorithms behave differently: OPTSAT and HCLASP backtrack and flip some of the objective literals, while CIRCUMSCRIPTINO alters the problem itself so that models falsifying all but one of the original atoms subject to minimization can be searched.

The modification strategy described above is actually the one implemented by many MaxSAT algorithms based on unsatisfiable core analysis (Morgado *et al.* 2013). In Algorithm 1, the unsatisfiable core analysis is performed according to ONE (Alviano *et al.* 2015). This design choice is motivated by the fact that the fresh variables $y_1, \ldots, y_n$ can be later assumed true in order to trivially satisfy the cardinality constraint and the implications introduced by the unsatisfiable core analysis, a feature not required for computing a single solution for a given MaxSAT instance. Eventually, the algorithm can be adapted to use different unsatisfiable core analysis techniques, in particular PMRES (Narodytska and Bacchus 2014) and K (Alviano *et al.* 2015).

The algorithm implemented by CAMUS_MCS (Liffiton and Sakallah 2008) is specifically conceived to address minimal correction subset enumeration, which is also considered in our experimental analysis. CAMUS_MCS adds to the input theory a cardinality constraint in order to compute models of bounded size; such a bound is iteratively increased until all minimal correction sets are computed. It turns out that CAMUS_MCS cannot run smoothly on an incremental solver, and in fact some of the learned clauses have to be eliminated when the bound of the cardinality constraint is changed. Such a drawback also affects the more general algorithm introduced

https://doi.org/10.1017/S1471068417000278 Published online by Cambridge University Press

by Faber *et al.* (2016): an external solver is used to enumerate cardinality minimal solutions of the input problem, and blocking clauses are then added to the theory so that the external solver can be invoked again for enumerating cardinality minimal solutions of the new theory; the process is repeated until the theory becomes unsatisfiable.

Concerning LBX (Mencía *et al.* 2015), and the algorithms ELS and CLD implemented by MCSLS (Marques-Silva *et al.* 2013), all of them follow an iterative approach, where models are improved by performing several calls to a SAT solver. Specifically, these algorithms start with any assignment, which is used to partition clauses into satisfied S and unsatisfied U. After that, these algorithms iteratively search for a new model satisfying all clauses in S and at least one clause in U. When no further improvement is possible, the last computed model is an MCS of the input theory, which is reported to the user and blocked by means of a blocking clause. The three algorithms differ in how they enforce an improvement in the current model: ELS checks the satisfiability of the theory $S \cup \{c\}$, for some $c \in U$; CLD checks the satisfiability of the theory $S \cup \{d\}$, where d is the disjunction of all literals occurring in U; LBX checks the satisfiability of the theory $S \cup \{\ell\}$, where ℓ is some literal occurring in U. The three algorithms additionally take advantage of a few enhancements, such as disjoint core analysis and backbone literals computation.

Another difference between the mentioned algorithms and the one implemented by CIRCUMSCRIPTINO is represented by the blocking clauses added to the input theory. In fact, since CIRCUMSCRIPTINO addresses model enumeration for circumscribed theories with grouping atoms (Lifschitz 1986), the assignment of non-grouping atoms (those in set R) has to be taken into account in the construction of the blocking clause associated with a computed model.

Lee and Lin (2006) studied theoretical properties of the computational problem associated with circumscription. In particular, they showed that models of circumscribed theories can be computed by adding *loop formulas* to the input theory, where the notion of loop formula is adapted from answer set programming (Lin and Zhao 2004; Lee and Lifschitz 2003). Within this respect, the algorithm implemented by CIRCUMSCRIPTINO requires less additions to the incremental SAT solver.

Finally, subset minimality is among the preferences natively supported in the language of ASPRIN (Brewka *et al.* 2015a, 2015b; Romero *et al.* 2016), a versatile framework built on top of CLINGO (Gebser *et al.* 2017). The algorithm implemented by ASPRIN is also iterative, meaning that better and better models are computed until an inconsistency arises. Differently from other iterative algorithms, however, the improvement on the current model is enforced by means of a *preference program*, which is possibly specified by the user in case of custom preferences.

6 Conclusion

Many practical problems require a preference relation over admissible solutions. When such a preference amounts to minimize a set of properties, the problem can be naturally represented in circumscription. Prominent examples of these problems have been considered in our experiments, namely the enumeration of minimal

https://doi.org/10.1017/S1471068417000278 Published online by Cambridge University Press

intervention strategies and the enumeration of minimal correction subsets. The proposed algorithm takes advantage of unsatisfiable core analysis, and showed to be very efficient in many cases. As a final remark, we stress here that the algorithm presented in this paper can be nicely combined with the tool LP2SAT in order to enumerate models of circumscribed answer set programming theories (under the restriction that answer set existence can be checked in NP; Alviano and Faber 2013); within this respect, completion of disjunctive logic programs (Alviano and Dodaro 2016c), specialized techniques to handle recursive aggregates (Alviano and Leone 2015; Alviano *et al.* 2015; Alviano 2016), and other algorithms implemented in efficient solvers such as CMODELS (Giunchiglia *et al.* 2004, 2006), CLASP (Gebser *et al.* 2012, 2015), DLV (Leone *et al.* 2006; Alviano *et al.* 2017) and WASP (Dodaro *et al.* 2011; Alviano *et al.* 2015) may also been employed.

References

Alviano, M. 2016. Evaluating answer set programming with non-convex recursive aggregates. *Fundamenta Informaticae 149,* 1–2, 1–34.

Alviano, M., Calimeri, F., Dodaro, C., Fuscà, D., Leone, N., Perri, S., Ricca, F., Veltri, P. and Zangari, J. 2017. The ASP system DLV2. In *Proc. of 14th International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR '17)*, July 3–6, Espoo, Finland, M. Balduccini and T. Janhunen, Eds., Lecture Notes in Computer Science, vol. 10377. Springer, 215–221.

Alviano, M. and Dodaro, C. 2016a. Answer set enumeration via assumption literals. In *Proc. of 25th International Conference of the Italian Association for Artificial Intelligence on Advances in Artificial Intelligence (AI*IA '16)*, November 29–December 1, Genova, Italy, G. Adorni, S. Cagnoni, M. Gori and M. Maratea, Eds., Lecture Notes in Computer Science, vol. 10037. Springer, 149–163.

Alviano, M. and Dodaro, C. 2016b. Anytime answer set optimization via unsatisfiable core shrinking. *Theory and Practice of Logic Programming 16,* 5–6, 533–551.

Alviano, M. and Dodaro, C. 2016c. Completion of disjunctive logic programs. In *Proc. of 25th International Joint Conference on Artificial Intelligence (IJCAI '16)*, July 9–15, New York, NY, USA, S. Kambhampati, Ed., IJCAI/AAAI Press, 886–892.

Alviano, M., Dodaro, C., Leone, N. and Ricca, F. 2015. Advances in WASP. In *Proc. of 13th International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR '15)*, September 27–30, Lexington, KY, USA, F. Calimeri, G. Ianni and M. Truszczynski, Eds., Lecture Notes in Computer Science, vol. 9345. Springer, 40–54.

Alviano, M., Dodaro, C. and Ricca, F. 2015. A maxsat algorithm using cardinality constraints of bounded size. In *Proc. of 24th International Joint Conference on Artificial Intelligenc (IJCAI '15)*, July 25–31, Buenos Aires, Argentina, Q. Yang and M. Wooldridge, Eds., AAAI Press, 2677–2683.

Alviano, M. and Faber, W. 2013. The complexity boundary of answer set programming with generalized atoms under the FLP semantics. In *Proc. of 12th International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR '13)*, September 15–19, Corunna, Spain, P. Cabalar and T. C. Son, Eds., Lecture Notes in Computer Science, vol. 8148. Springer, 67–72.

Alviano, M., Faber, W. and Gebser, M. 2015. Rewriting recursive aggregates in answer set programming: Back to monotonicity. *Theory and Practice of Logic Programming 15,* 4–5, 559–573.

https://doi.org/10.1017/S1471068417000278 Published online by Cambridge University Press

Alviano, M. and Leone, N. 2015. Complexity and compilation of GZ-aggregates in answer set programming. *Theory and Practice of Logic Programming 15,* 4–5, 574–587.

Andres, B., Kaufmann, B., Matheis, O. and Schaub, T. 2012. Unsatisfiability-based optimization in clasp. In *Proc. of Technical Communications of the 28th International Conference on Logic Programming (ICLP '12)* , September 4–8, Budapest, Hungary, A. Dovier and V. S. Costa, Eds., LIPIcs, vol. 17, Schloss Dagstuhl – Leibniz-Zentrum fuer Informatik, 211–221.

Audemard, G. and Simon, L. 2009. Predicting learnt clauses quality in modern SAT solvers. In *Proc. of 21st International Joint Conference on Artificial Intelligence (IJCAI '09)*, July 11–17, Pasadena, CA, USA, C. Boutilier, Ed., 399–404.

Bacchus, F. and Narodytska, N. 2014. Cores in core based maxsat algorithms: An analysis. In *Proc. of 17th International Conference on Theory and Applications of Satisfiability Testing (SAT '14), Held as Part of the Vienna Summer of Logic (VSL '04)*, July 14–17, Vienna, Austria, C. Sinz and U. Egly, Eds., Lecture Notes in Computer Science, vol. 8561. Springer, 7–15.

Brewka, G., Delgrande, J. P., Romero, J. and Schaub, T. 2015a. asprin: Customizing answer set preferences without a headache. In *Proc. of 29th AAAI Conference on Artificial Intelligence*, January 25–30, Austin, TX, USA, B. Bonet and S. Koenig, Eds. AAAI Press, 1467–1474.

Brewka, G., Delgrande, J. P., Romero, J. and Schaub, T. 2015b. Implementing preferences with asprin. In *Proc. of 13th International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR '15)*, September 27–30, Lexington, KY, USA, F. Calimeri, G. Ianni and M. Truszczynski, Eds., Lecture Notes in Computer Science, vol. 9345. Springer, 158–172.

Dodaro, C., Alviano, M., Faber, W., Leone, N., Ricca, F. and Sirianni, M. 2011. The birth of a WASP: preliminary report on a new ASP solver. In *Proc. of 26th Italian Conference on Computational Logic*, August 31–September 2, Pescara, Italy, F. Fioravanti (Ed.), CEUR Workshop Proceedings, vol. 810. CEUR-WS.org, 99–113.

Faber, W., Vallati, M., Cerutti, F. and Giacomin, M. 2016. Solving set optimization problems by cardinality optimization with an application to argumentation. In *Proc. of 22nd European Conference on Artificial Intelligence (ECAI '16) – Including Prestigious Applications of Artificial Intelligence (PAIS '16)*, 29 August–2 September, The Hague, The Netherlands, G. A. Kaminka, M. Fox, P. Bouquet, E. Hüllermeier, V. Dignum, F. Dignum and F. van Harmelen, Eds., Frontiers in Artificial Intelligence and Applications, vol. 285. IOS Press, 966–973.

Gebser, M., Kaminski, R., Kaufmann, B., Romero, J., and Schaub, T. 2015. Progress in clasp series 3. In *Proc. of 13th International Conference Logic Programming and Nonmonotonic Reasoning (LPNMR '05)*, September 27–30, Lexington, KY, USA, F. Calimeri, G. Ianni and M. Truszczynski, Eds., Lecture Notes in Computer Science, vol. 9345, 2015. Springer, 368–383.

Gebser, M., Kaminski, R., Kaufmann, B. and Schaub, T. 2009. On the implementation of weight constraint rules in conflict-driven ASP solvers. In *Proc. of 25th International Conference on Logic Programming (ICLP '09)*, July 14–17, Pasadena, CA, USA, P. M. Hill and D. S. Warren, Eds., Lecture Notes in Computer Science, vol. 5649. Springer, 250–264.

Gebser, M., Kaminski, R., Kaufmann, B. and Schaub, T. 2017. Multi-shot ASP solving with clingo. *CoRR abs/1705.09811*.

Gebser, M., Kaufmann, B., Neumann, A. and Schaub, T. 2007. Conflict-driven answer set enumeration. In *Proc. of 9th International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR '07)*, May 15–17, Tempe, AZ, USA, C. Baral, G. Brewka and J. S. Schlipf, Eds., Lecture Notes in Computer Science, vol. 4483. Springer, 136–148.

https://doi.org/10.1017/S1471068417000278 Published online by Cambridge University Press

Gebser, M., Kaufmann, B., Romero, J., Otero, R., Schaub, T. and Wanko, P. 2013. Domain-specific heuristics in answer set programming. In *Proc. of 27th AAAI Conference on Artificial Intelligence*, July 14–18, Bellevue, Washington, USA, M. desJardins and M. L. Littman, Eds. AAAI Press.

Gebser, M., Kaufmann, B. and Schaub, T. 2012. Conflict-driven answer set solving: From theory to practice. *Artificial Intelligence 187*, 52–89.

Giunchiglia, E., Lierler, Y. and Maratea, M. 2004. Sat-based answer set programming. In *Proc. of 19th National Conference on Artificial Intelligence, Sixteenth Conference on Innovative Applications of Artificial Intelligence*, July 25–29, San Jose, CA, USA, D. L. McGuinness and G. Ferguson, Eds. AAAI Press/MIT Press, 61–66.

Giunchiglia, E., Lierler, Y. and Maratea, M. 2006. Answer set programming based on propositional satisfiability. *Journal of Automated Reasoning 36,* 4, 345–377.

Giunchiglia, E. and Maratea, M. 2006. optsat: A tool for solving SAT related optimization problems. In *Proc. of 10th European Conference on Logics in Artificial Intelligence (JELIA '06)*, September 13–15, Liverpool, UK, M. Fisher, W. van der Hoek, B. Konev and A. Lisitsa, Eds. Lecture Notes in Computer Science, vol. 4160. Springer, 485–489.

Janhunen, T. and Niemelä, I. 2011. Compact translations of non-disjunctive answer set programs to propositional clauses. In *Logic Programming, Knowledge Representation, and Nonmonotonic Reasoning – Essays Dedicated to Michael Gelfond on the Occasion of His 65th Birthday*, M. Balduccini and T. C. Son, Eds., Lecture Notes in Computer Science, vol. 6565. Springer, 111–130.

Jannach, D., Schmitz, T. and Shchekotykhin, K. M. 2016. Parallel model-based diagnosis on multi-core computers. *Journal of Artificial Intelligence Research 55*, 835–887.

Järvisalo, M., Berre, D. L., Roussel, O. and Simon, L. 2012. The international SAT solver competitions. *AI Magazine 33,* 1, 89–94.

Junker, U. 2004. QUICKXPLAIN: Preferred explanations and relaxations for over-constrained problems. In *Proc. of 19th National Conference on Artificial Intelligence, 16th Conference on Innovative Applications of Artificial Intelligence*, July 25–29, San Jose, CA, USA, D. L. McGuinness and G. Ferguson, Eds. AAAI Press/MIT Press, 167–172.

Kaminski, R., Schaub, T., Siegel, A. and Videla, S. 2013. Minimal intervention strategies in logical signaling networks with ASP. *Theory and Practice of Logic Programming 13,* 4–5, 675–690.

Lee, J. and Lifschitz, V. 2003. Loop formulas for disjunctive logic programs. In *Proc. of 19th International Conference on Logic Programming (ICLP '03)*, December 9–13, Mumbai, India, C. Palamidessi, Ed. Lecture Notes in Computer Science, vol. 2916. Springer, 451–465.

Lee, J. and Lin, F. 2006. Loop formulas for circumscription. *Artificial Intelligence 170,* 2, 160–185.

Leone, N., Pfeifer, G., Faber, W., Eiter, T., Gottlob, G., Perri, S. and Scarcello, F. 2006. The DLV system for knowledge representation and reasoning. *ACM Transactions on Computational Logic 7,* 3, 499–562.

Liffiton, M. H. and Sakallah, K. A. 2008. Algorithms for computing minimal unsatisfiable subsets of constraints. *Journal of Automated Reasoning 40,* 1, 1–33.

Lifschitz, V. 1986. On the satisfiability of circumscription. *Artificial Intelligence 28,* 1, 17–27.

Lin, F. and Zhao, Y. 2004. ASSAT: Computing answer sets of a logic program by SAT solvers. *Artificial Intelligence 157,* 1–2, 115–137.

Marques-Silva, J., Heras, F., Janota, M., Previti, A. and Belov, A. 2013. On computing minimal correction subsets. In *Proc. of 23rd International Joint Conference on Artificial Intelligence (IJCAI '13)*, August 3–9, Beijing, China, F. Rossi, Ed. IJCAI/AAAI, 615–622.

https://doi.org/10.1017/S1471068417000278 Published online by Cambridge University Press

Marques-Silva, J. and Previti, A. 2014. On computing preferred MUSes and MCSes. In *Proc. of 17th International Conference on Theory and Applications of Satisfiability Testing (SAT '14), Held as Part of the Vienna Summer of Logic (VSL '14)*, July 14–17, Vienna, Austria, C. Sinz and U. Egly, Eds. Lecture Notes in Computer Science, vol. 8561. Springer, 58–74.

McCarthy, J. 1980. Circumscription – A form of non-monotonic reasoning. *Artificial Intelligence 13,* 1–2, 27–39.

Mencía, C., Ignatiev, A., Previti, A. and Marques-Silva, J. 2016. MCS extraction with sublinear oracle queries. In *Proc. of 19th International Conference on Theory and Applications of Satisfiability Testing (SAT '16)*, July 5–8, Bordeaux, France, N. Creignou and D. L. Berre, Eds. Lecture Notes in Computer Science, vol. 9710. Springer, 342–360.

Mencía, C., Previti, A. and Marques-Silva, J. 2015. Literal-based MCS extraction. In *Proc. of 24th International Joint Conference on Artificial Intelligence (IJCAI '15)*, July 25–31, Buenos Aires, Argentina, Q. Yang and M. Wooldridge, Eds. AAAI Press.

Morgado, A., Heras, F., Liffiton, M. H., Planes, J. and Marques-Silva, J. 2013. Iterative and core-guided maxsat solving: A survey and assessment. *Constraints 18,* 4, 478–534.

Narodytska, N. and Bacchus, F. 2014. Maximum satisfiability using core-guided maxsat resolution. In *Proc. of 28th AAAI Conference on Artificial Intelligence*, July 27–31, Québec City, Québec, Canada, C. E. Brodley and P. Stone, Eds., AAAI Press, 2717–2723.

Pereira, L. M., Damásio, C. V. and Alferes, J. J. 1993. Debugging by diagnosing assumptions. In *Proc. of 1st International Workshop Automated and Algorithmic Debugging (AADEBUG '93)*, May 3–5, Linköping, Sweden, P. Fritszon (Ed.), Lecture Notes in Computer Science, vol. 749. Springer, 58–74.

Romero, J., Schaub, T. and Wanko, P. 2016. Computing diverse optimal stable models. In *Proc. of Technical Communications of the 32nd International Conference on Logic Programming, (ICLP-TC '16)*, October 16–21, New York, NY, USA, M. Carro, A. King, N. Saeedloei and M. D. Vos, Eds. OASICS, vol. 52, Schloss Dagstuhl – Leibniz-Zentrum fuer Informatik, 3:1–3:14.

https://doi.org/10.1017/S1471068417000278 Published online by Cambridge University Press