cmMUS: A Tool for Circumscription-Based MUS Membership Testing

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

[1] University College Dublin, Ireland
[2] INESC-ID, Lisbon, Portugal

Abstract. This article presents cmMUS—a tool for deciding whether a clause belongs to some minimal unsatisfiable subset (MUS) of a given formula. The MUS-membership has a number of practical applications, related with understanding the causes of unsatisfiability. However, it is computationally challenging as it is Σ_2^P-complete. cmMUS solves the problem by translating needs to propositional circumscription, another well-known Σ_2^P-complete problem from the area of non-monotonic reasoning. The tool constantly outperforms other approaches to the problem, which is demonstrated on a variety of benchmarks.

1 Introduction

Unsatisfiable formulas, representing refutation-proofs or inconsistencies, appear in various areas of automated reasoning. This article presents a tool that helps us to understand why a certain formula is unsatisfiable. To understand why a formula in the conjunctive normal form (CNF), is unsatisfiable, it is sufficient to consider only some of its subsets of clauses. More precisely, a set of clauses is called a minimally unsatisfiable subset (MUS) if it is unsatisfiable and any of its subsets is satisfiable. cmMUS determines whether a given clause belongs to *some* MUS. This is referred to as the MUS-MEMBERSHIP problem.

The MUS-MEMBERSHIP problem is important when one wants to *restore consistency* of a formula: removing a clause that is *not* part of any MUS, will certainly not restore consistency. Restoring consistency is an active area of research in the area of *product configuration* [14,15]. For example, when configuring a product, some sets of features result in an inconsistent configuration. Approaches for resolving conflicting features often involves user intervention, e.g. to decide which features to remove. Clearly, it is preferable to allow the user to deselect features relevant for the inconsistency.

2 Background

Throughout this paper, ϕ and ψ denote Boolean formulas, defined on a set of variables $X = \{x_1, \ldots, x_n\}$. Where necessary, additional variables can be considered. A Boolean formula ϕ in Conjunctive Normal Form (CNF) is a conjunction of disjunctions of literals. Each disjunction of literals is called a *clause*, and it is preferably represented by ω. Where appropriate, a CNF formula is interpreted as a set of clauses.

A *truth assignment* μ_X is a mapping from a set of variables X to $\{0,1\}$, $\mu_X : X \rightarrow \{0,1\}$. A truth assignment is represented by the set M_X of true variables in μ_X, $M_X = \{x_i \in X \mid \mu_X(x_i) = 1\}$. In what follows, truth assignments will be represented by the set of true variables, since the definition of μ_X is implicit, given X and M_X. Moreover, $M_X \models \phi$ is used to denote that truth assignment μ_X is a *model* of ϕ, i.e. that μ_X satisfies all clauses in ϕ. Truth assignments will also be defined for other sets of variables, as needed, e.g. M_S, M_{S_a}, M_{S_b}. When a formula is defined over distinct sets of variables, e.g. X and S, $M_S, M_X \models \phi$ denotes that the truth assignment to the variables in S and the variables in X satisfies ϕ. Finally, a truth assignment represented by M_X implicitly denotes that $M_X \subseteq X$. Similarly, M_R implicitly denotes that $M_R \subseteq R$. To simplify the notation, the set containment relation will be omitted in all formulas.

A *QBF formula* is a Boolean formula where variables can be universally or existentially quantified. We write QBF$_{k,\exists}$ to denote the class of formulas of the form $\exists X_1 \forall Y_1 \ldots \exists X_k \forall Y_k.\ \phi$. An important result from the complexity theory is that the validity of a formula in QBF$_{k,\exists}$ is Σ_k^P-complete [13].

A *Disjunctive Logic Program* (*DLP*) is a set of rules of the form $a_1 \vee \cdots \vee a_n \leftarrow b_1, \ldots, b_m, {\sim}c_1, \ldots, {\sim}c_k$, where a_j's, b_i's, c_l's are propositional atoms. The part $a_1 \vee \cdots \vee a_n$ is called the *head* and is viewed as a disjunction. The part right of $\leftarrow$ is called the *body* and is viewed as a conjunction. The symbol $\sim$ is the *default negation* (the failure to prove). The empty head is denoted as $\perp$. If a program comprises only rules with $k = 0$, the program is called *positive*. The *stable model semantics* is assumed for disjunctive logic programs [3,4].

Circumscription was introduced by McCarthy as a form of nonmonotonic reasoning [12]. While the original definition of circumscription is for first-order logic, for the purpose of this article we consider its propositional version. For a set of variables R, a model M of the formula ϕ is an *R-minimal model* if it is minimal with respect to the point-wise ordering on the variables R. The *circumscription inference problem* is the problem of deciding whether a formula ψ holds in all R-minimal models of a formula ϕ. If a formula ψ holds in all R-minimal models of a formula ϕ we write $\phi \models_R^{circ} \psi$.

We say that a set of clauses $\psi \subseteq \phi$ is a *Maximally Satisfiable Subformula (MSS)* iff ψ is unsatisfiable and any set $\psi' \subsetneq \psi$ is satisfiable. Dually, we say that a set of clauses $\psi \subseteq \phi$ is a *Minimally Unsatisfiable Subformula (MUS)* iff ψ is unsatisfiable and any set $\psi' \subsetneq \psi$ is satisfiable. The definition of MUSes yields the following problem.

Name: MUS-MEMBERSHIP
Given: A CNF formula ϕ and a clause ω.
Question: Is there an MUS ψ of ϕ such that $\omega \in \psi$?

Name: MUS-OVERLAP
Given: CNF formulas ϕ and γ.
Question: Is there an MUS ψ of ϕ such that $\gamma \cap \psi \neq \emptyset$?

Observe that MUS-MEMBERSHIP is a special case of MUS-OVERLAP if only a single clause is considered, and, that MUS-OVERLAP can be expressed as a disjunction of k instances of MUS-MEMBERSHIP, where k is the number of clauses in the formula γ. However, cmMUS solves directly the more general problem MUS-OVERLAP.

It has been shown that MUS-MEMBERSHIP, and therefore MUS-OVERLAP, is Σ_2^P-complete [10].

3 cmMUS Description

cmMUS[3] solves MUS-OVERLAP by translating it to propositional circumscription. It accepts a formula in the DIMACS format and a list of indices representing the clauses tested for overlap. To decide MUS-OVERLAP for a formula ϕ and a set of clauses γ the tool performs the following steps.

1. It introduces the *relaxed form* of the formula $\phi^* = \{\omega \vee r_\omega \mid \omega \in \phi\}$, where r_ω are fresh variables.
2. It generates the circumscription entailment problem $\phi^* \models_R^{circ} \bigvee_{\omega\in\gamma} \neg r_\omega$.
3. It solves the entailment by a dedicated algorithm based on counterexample guided abstraction refinement [8].
4. If the answer to the entailment problem is “valid”, then there is no overlap between MUSes of ϕ and the clauses γ. If the answer to the entailment problem is “invalid”, then there is an overlap. Further, if r_ω has the value 0 in a counterexample to the entailment, the clause ω overlaps with some MUS of ϕ.

Apart from the “yes”/“no” answer to the given MUS-OVERLAP problem, in the case of an overlap (“yes”), the tool outputs a formula $\phi' \subseteq \phi$ such that ϕ' is unsatisfiable and any of its MUSes overlaps with γ. Details of the translation are explained in the pertaining technical report [9].

4 Experimental Results

The following tools where considered for the experimental evaluation in addition to cmMUS.

look4MUS is a tool dedicated to MUS-MEMBERSHIP based on MUS enumeration, guided by heuristics based on a measure of inconsistency [6].

Quantified Boolean Formula (QBF) The problem was expressed as a QBF [9] and inputted to the QBF solver `QuBE7.1`[4]. The solver was chosen because it solved the most instances in the 2QBF track of QBF Evaluation 2010[5]. The solver was invoked with all its preprocessing techniques [5] (using the `-all` switch).

MSS enum. A clause appears in some MUS if there exists an MSS that does not contain it. The tool `CAMUS` [11] was used to enumerate MSSes of the given formula. The enumeration stops if it encounters an MSS that does not contain at least one of the clauses in γ.

Disjunctive Logic Programming The translation to disjunctive logic programming (DLP) was performed in a sequence of steps.

1. Translate the relaxed version ϕ^* into a positive disjunctive logic program by putting positive atoms in the head and negative in the body.

[3] Available at http://sat.inesc-id.pt/~mikolas/sw/cmmus
[4] Available at www.star.dist.unige.it/~qube/.
[5] http://www.qbflib.org/

	cmMUS	look4MUS	MSS enum.	QBF	DLP
Nemesis (bf) (223)	223	223	31	8	0
Daimler-Chrysler (84)	46	13	49	0	0
dining philosophers (22)	18	17	4	0	0
dimacs (87)	87	82	51	48	0
ezfact (41)	21	11	11	0	0
crafted (24)	24	14	13	12	5
total (481)	419	360	159	68	5

Table 1. Number of solved instances by the different approaches

2. Apply the tool `circ2dlp` [7] to produce a disjunctive logic program whose stable models correspond to the R-minimal models of the given formula.
3. To find out whether the set of clauses $\omega_1, \omega_2, ..., \omega_n$ overlaps with any MUS of ϕ^*, add to this program the rule $\perp \leftarrow {\sim}r_{\omega_1}, {\sim}r_{\omega_2}, ..., {\sim}r_{\omega_n}$ which disables the stable models where none of the clauses in question are relaxed. The resulting program has at least one model iff there exists an MSS such that at least one of the clauses in question is relaxed, an approach suggested in [2].
4. Run the DLP solver `claspD` [1] to decide whether it has at least one model are not.

A variety of unsatisfiable formulas was selected from the benchmarks used for SAT competitions[6] and from well-known applications of SAT (namely ATPG and product configuration). The selected formulas are relatively easy for modern SAT solvers because MUS-MEMBERSHIP is significantly harder than satisfiability. Even so, instances with tens of thousands of clauses were used (e.g. dining philosophers).

For each of these formulas the MUS-OVERLAP was computed using the various approaches. The 1st, 3rd, 5th, and 7th clauses in the formula's representation were chosen as the set γ for which the overlap was to be determined—this testing methodology was also used in [6].

All experimental results were obtained on an Intel Xeon 5160 3GHz with 4GB of memory. The experiments were obtained with a time limit of 1,000 seconds. The results of the measurements are presented by Table 1 and Figure 1. Table 1 presents the number of solved instances by each of the approaches for each set of benchmarks. Figure 1 presents the computation times with *cactus plots*—the horizontal axis represents the number of instances that were solved within the time represented by the vertical axis.

Out of the presented approaches, the cmMUS is the most robust one: it has solved the most instances (419) and except for one class of benchmarks it exhibits the shortest overall running times. The set of benchmarks where cmMUS came second are the Daimler-Chrysler. In these benchmarks the simple MSS enumeration solved 3 more instances.

The dedicated algorithm `look4MUS` came second in terms of number of the solved instances (360) and it solved a number of benchmarks in a short time (Nemesis-bf), although slower than cmMUS. However, it turned out not to be robust (e.g. a small number of instances were solved in suite Daimler-Chrysler and ezfact).

[6] http://www.satcompetition.org/

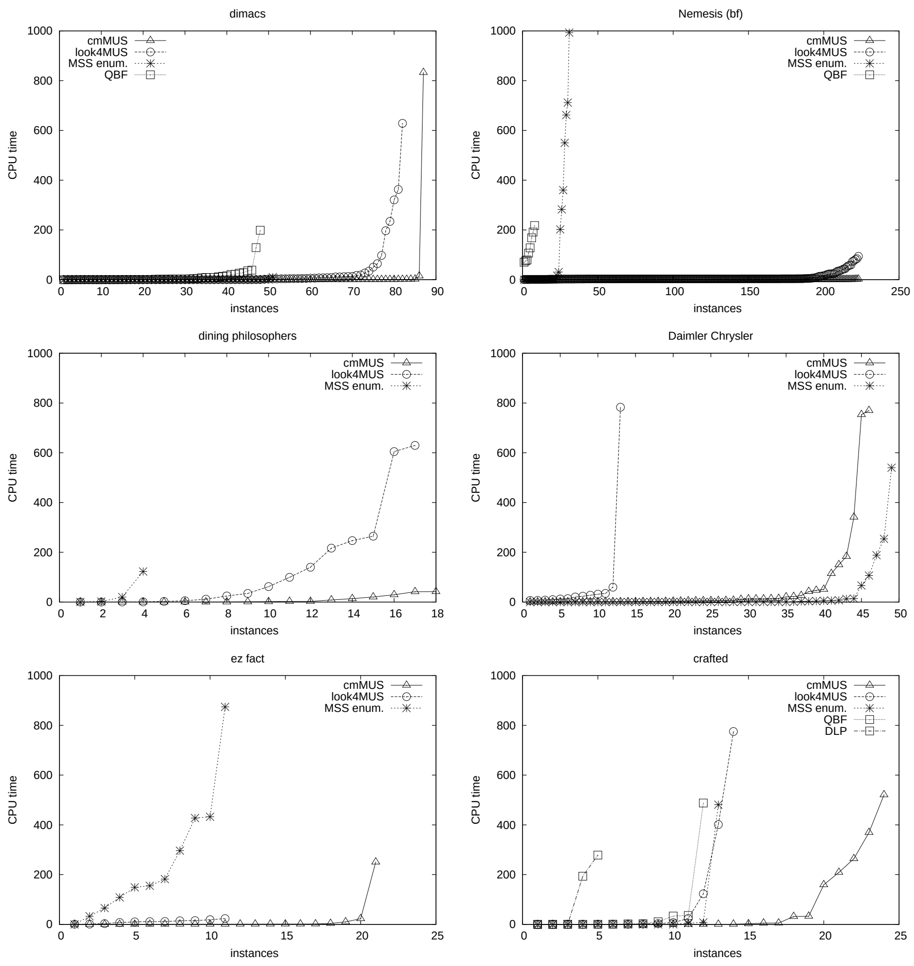

Fig. 1. Cactus plots for the measurements (number of instances x solved in less than y seconds)

The QBF and DLP approaches turned out to be the least successful ones. In the case of DLP this is most likely attributed to the relatively small number of variables on which the circumscription is being minimized (the set P). This weakness has already been highlighted by the authors of `circ2dlp` [7]. However, to our knowledge, the solver `claspD` does not use such extensive preprocessing techniques as `Qube 7.1`. Hence, this could be investigated in the future.

5 Summary

This article presents cmMUS—a tool for deciding the MUS-MEMBERSHIP problem, i.e. it decides whether a given clause belongs to some minimally unsatisfiable set. The tool translates the problem into entailment in propositional circumscription, on which it invokes a dedicated algorithm based on abstraction counterexample refinement [8].

A variety of benchmarks shows that the tool outperforms existing approaches to the problem.

Acknowledgement. This work is partially supported by SFI PI grant BEACON (09/IN.1/I2618), by FCT through grant ATTEST (CMU-PT/ELE/0009/2009), and by INESC-ID multiannual funding from the PIDDAC program funds.

References

1. Drescher, C., Gebser, M., Grote, T., Kaufmann, B., König, A., Ostrowski, M., Schaub, T.: Conflict-driven disjunctive answer set solving. In: Brewka, G., Lang, J. (eds.) KR. pp. 422–432. AAAI Press (2008)
2. Eiter, T., Gottlob, G.: On the computational cost of disjunctive logic programming: Propositional case. Annals of Mathematics and Artificial Intelligence 15, 289–323 (1995)
3. Gelfond, M., Lifschitz, V.: Classical negation in logic programs and disjunctive databases. New generation computing 9(3), 365–385 (1991)
4. Gelfond, M.: Handbook of Knowledge Representation, chap. Answer Sets. Elsevier (2008)
5. Giunchiglia, E., Marin, P., Narizzano, M.: An effective preprocessor for QBF pre-reasoning. In: 2nd International Workshop on Quantification in Constraint Programming (QiCP) (2008)
6. Grégoire, E., Mazure, B., Piette, C.: Does this set of clauses overlap with at least one MUS? In: Proceedings of the 22nd International Conference on Automated Deduction. pp. 100–115. CADE-22, Springer-Verlag, Berlin, Heidelberg (2009)
7. Janhunen, T., Oikarinen, E.: Capturing parallel circumscription with disjunctive logic programs. In: European Conf. on Logics in Artif. Intell. pp. 134–146 (2004)
8. Janota, M., Grigore, R., Marques-Silva, J.: Counterexample guided abstraction refinement algorithm for propositional circumscription. In: Proceeding of the 12th European Conference on Logics in Artificial Intelligence (JELIA) (2010)
9. Janota, M., Marques-Silva, J.: Models and algorithms for MUS membership testing. Tech. Rep. TR-07/2011, INESC-ID (January 2011)
10. Kullmann, O.: Constraint satisfaction problems in clausal form: Autarkies and minimal unsatisfiability. Electronic Colloquium on Computational Complexity (ECCC) 14(055) (2007)
11. Liffiton, M.H., Sakallah, K.A.: Algorithms for computing minimal unsatisfiable subsets of constraints. J. Autom. Reasoning 40(1), 1–33 (2008)
12. McCarthy, J.: Circumscription - a form of non-monotonic reasoning. Artif. Intell. 13(1-2), 27–39 (1980)
13. Meyer, A.R., Stockmeyer, L.J.: The equivalence problem for regular expressions with squaring requires exponential space. In: Switching and Automata Theory, 1972., IEEE Conference Record of 13th Annual Symposium on (October 1972)
14. O'Callaghan, B., O'Sullivan, B., Freuder, E.C.: Generating corrective explanations for interactive constraint satisfaction. In: van Beek, P. (ed.) CP. Lecture Notes in Computer Science, vol. 3709, pp. 445–459. Springer (2005)
15. Papadopoulos, A., O'Sullivan, B.: Relaxations for compiled over-constrained problems. In: Stuckey, P.J. (ed.) CP. Lecture Notes in Computer Science, vol. 5202, pp. 433–447. Springer (2008)