QMusExt: A Minimal (Un)satisfiable Core Extractor for Quantified Boolean Formulas

Andreas Plank
Johannes Kepler Universität Linz, Austria

Martina Seidl
Johannes Kepler Universität Linz, Austria

Abstract

In this paper, we present QMusExt, a tool for the extraction of minimal unsatisfiable sets (MUS) from quantified Boolean formulas (QBFs) in prenex conjunctive normal form (PCNF). Our tool generalizes an efficient algorithm for MUS extraction from propositional formulas that analyses and rewrites resolution proofs generated by SAT solvers.

In addition to extracting unsatisfiable cores from false formulas in PCNF, we apply QMusExt also to obtain satisfiable cores from Q-resolution proofs of true formulas in prenex disjunctive normal form (PDNF).

2012 ACM Subject Classification Theory of computation → Logic and verification

Keywords and phrases Minimal Unsatisfiable Core, Quantified Boolean Formula

Digital Object Identifier 10.4230/LIPIcs.SAT.2023.20

Supplementary Material *Software (Source Code)*: `https://github.com/PlankAndreas/QMusExt` archived at `swh:1:dir:29b2426d2111cd93ebe0c5a9c41e9dc1f337fcce`

Funding This work has been supported by the LIT AI Lab funded by the State of Upper Austria.

1 Introduction

We present the tool QMusExt that computes a *minimal unsatisfiable set* (MUS), also called minimal unsatisfiable core, of a false quantified Boolean formula (QBF) $\Pi.\phi$ in prenex conjunctive normal form. An MUS is a subformula of $\phi' \subseteq \phi$ such that $\Pi.\phi'$ is false and removing any clause from ϕ' would make the formula true. Hence, an MUS describes a set of contradicting constraints from which no clause may be removed without eliminating the inconsistency as well. As this information is very important for understanding the reason of an inconsistency, many approaches have been presented to compute minimal unsatisfiable cores for propositional formulas [6]. In general, the MUS of a formula is not necessarily unique, a formula can have multiple MUSes and usually the smaller ones are preferred, i.e., size is a measure on the quality of the MUS extraction algorithm.

For QBFs, only few approaches for calculating MUSes exist so far, although MUS extraction is an important problem here as well. In [7, 6], theoretical properties of MUS extraction have been studied. An approach that extracts unsatisfiable cores which are not necessarily minimal is employed in [13] to validate the correctness of QBF solving results. In [4] the extraction of unsatisfiable cores is discussed in the context of the quantified MaxSAT problem. An approach to extract minimal unsatisfiable cores from false QBFs in PCNF was presented by Lonsing and Egly [8]. In this work, the solver DepQBF was equipped with an interface for incremental solving that provides a clause grouping feature. They showed that with this feature, the iterative clause set refinement approach with selector variables is straight-forward to implement for PCNF formulas. To the best of our knowledge, they provided the first available tool for MUS extraction. Most recently, Niskanen et al. presented an approach to find a smallest MUS of a false QBF based on implicit hitting sets [12].

The approach implemented in our tool QMusExt works differently. It applies a proof-based approach that was originally suggested for propositional formulas [2]. In particular, it reduces the set of initial clauses of a Q-resolution proof as produced by modern QBF solvers until an MUS is extracted. Therefore, the proof needs to be iteratively updated. By exploiting the duality of true and false QBFs, satisfiable cores of true QBFs can be obtained by employing our approach. Our tool QMusExt is implemented in C and is available under MIT license at

`https://github.com/PlankAndreas/QMusExt`.

2 Preliminaries

We consider QBFs of the form $\Pi.\phi$, where $\Pi = Q_1X_1...Q_nX_n$ is called the *quantifier prefix*, $X_1, \ldots, X_n$ are pairwise disjoint sets of Boolean variables, $Q_i \in \{\forall, \exists\}$, and $Q_i \neq Q_{i+1}$. The *matrix* ϕ is a propositional formula either in *prenex conjunctive normal form (PCNF)*, i.e., it is a conjunction of clauses, or in *prenex disjunctive normal form (PDNF)*, i.e., it is a disjunction of cubes. As usual, a *clause* is a disjunction of literals and a *cube* is a conjunction of literals. If convenient, we interpret clauses and cubes as sets of literals. A *literal* is a variable or a negated variable. We define $Var(l) = x$ if $l = x$ or $l = \bar{x}$ for any literal l. We say a literal is existential (universal), if its variable is existentially (universally) quantified. Further, $\bar{l} = x$ if $l = \bar{x}$ and $\bar{l} = \bar{x}$ if $l = x$. A quantifier prefix $Q_1X_1 \ldots Q_nX_n$ imposes an ordering $<$ on the variables: if $x_i \in X_i$, $x_j \in X_j$, and $i < j$, then $x_i < x_j$. For a propositional formula ϕ, ϕ_l denotes the formula obtained by setting variable x to true if l $=$ x and by setting x to false if $l = \bar{x}$. A QBF $\forall X\Pi.\phi$ is true iff $\forall X'\Pi.\phi_x$ and $\forall X'\Pi.\phi_{\bar{x}}$ are true where $X' = X \setminus \{x\}$. Respectively, a QBF $\exists X\Pi.\phi$ is true iff $\exists X'\Pi.\phi_x$ or $\exists X'\Pi.\phi_{\bar{x}}$ is true. For example $\forall x\exists y.(x \vee y) \wedge (\bar{x} \vee \bar{y})$ is true and $\exists x\forall y.(x \vee y) \wedge (\bar{x} \vee \bar{y})$ is false. Every false QBF $\Pi.\phi$ in PCNF can be refuted by Q-resolution [5] which consists of the following three clause-derivation rules:

- **Axiom**: Any clause of ϕ can be derived.
- **Resolution**: From already derived clauses $C \vee x$ and $D \vee \bar{x}$, a clause $C \vee D$ can be derived if there is no literal l with $l, \bar{l} \in C \cup D$, $x \notin D$, $\bar{x} \notin C$, and x is existentially quantified.
- **Universal Reduction**: From an already derived clause $C \vee l$, a clause C can be derived if l is universal and there is no existential literal $k \in C$ with $l < k$.

A QBF is false iff the empty clause $\Box$ can be derived via Q-resolution. Dually, every true QBF $\Pi.\phi$ in PCNF can be proven by Q-resolution [3] which consists of the following three cube-derivation rules:

- **Axiom**: Let σ be a satisfying assignment of ϕ. Then cube $\bigwedge_{l \in \sigma} l$ can be derived.
- **Resolution**: From already derived cubes $C \wedge x$ and $D \wedge \bar{x}$, a cube $C \wedge D$ can be derived if there is no literal l with $l, \bar{l} \in C \cup D$, $x \notin D$, $\bar{x} \notin C$, and x is universally quantified.
- **Existential Reduction**: From an already derived cube $C \wedge l$, a cube C can be derived if l is existential and there is no universal literal $k \in C$ with $l < k$.

A QBF is true iff the empty cube can be derived via Q-resolution. A clause/cube derived via the resolution rule is called *resolvent*, while the parent clauses/cubes are called *antecedents*. Clauses with no antecedents are called *initial clauses*. Respectively, cubes without antecedents are called *initial cubes*. While initial clauses are directly available from the given PCNF formula, initial cubes have to be found by the QBF solver. Q-resolution proofs can be described in terms of *resolution graphs*. For a false QBF $\Pi.\phi$, a *resolution graph* $P = (V, E)$ is a directed acyclic graph (DAG). The set of vertices $V = V^{\mathsf{i}} \cup V^{\mathsf{d}}$ consists of initial clauses $V^{\mathsf{i}} \subseteq \phi$ and derived clauses V^{d}. Edges connect two antecedents and their resolvent, or a clause C and a clause C' that is obtained by universally reducing C. The only sink vertex is the empty clause denoted by $\Box$. Resolution graphs for true QBFs are defined respectively.

A vertex D is called *reachable* in resolution graph P from a vertex C iff there is a path from vertex C to vertex D. With $cone(P, C)$ we denote the set of all vertices reachable from vertex C in a resolution graph P (the *cone* of a clause C). Dually the set $unRe(P, C)$ contains all clauses not reachable from clause C. Finally, we define a resolution graph P to be *non-redundant* if all vertices are connected. In the following, we will use Q-resolution proofs to detect minimal (un)satisfiable cores which are defined as follows.

▶ **Definition 1** (Minimal Unsatisfiable Core). *For a false QBF $\Phi = \Pi.\phi$ in PCNF, a sub-formula $\phi' \subseteq \phi$ is a* minimal unsatisfiable core *of Φ, if $\Pi.\phi'$ is false and $\Pi.\phi' \backslash \{C\}$ is true for all $C \in \phi'$.*

The size of an unsatisfiable core is the number of its clauses. In general, minimal unsatisfiable cores are not unique and they can be of different size. For example, the formula $\exists x \forall y \exists z.((x \vee y \vee z) \wedge (\bar{x} \vee \bar{y}) \wedge (\bar{z}) \wedge (x \vee y \vee \bar{z}) \wedge (z \vee y))$ has minimal unsatisfiable cores $((x \vee y \vee z) \wedge (\bar{x} \vee \bar{y}) \wedge (\bar{z}))$, $((\bar{z}) \wedge (z \vee y))$, and $((x \vee y \vee z) \wedge (\bar{x} \vee \bar{y}) \wedge (x \vee y \vee \bar{z}))$.

▶ **Definition 2** (Minimal Satisfiable Core). *For a true QBF $\Phi = \Pi.\phi$ in PDNF, a sub-formula $\phi' \subseteq \phi$ is a* minimal satisfiable core *of Φ, if $\Pi.\phi'$ is true and $\Pi.\phi' \backslash \{C\}$ is false for all $C \in \phi'$.*

3 QMusExt

Our tool QMusExt extracts unsatisfiable cores from Q-resolution refutations of false QBFs. Further, it extracts satisfiable cores from Q-resolution satisfaction proofs of true QBFs. In both cases, QMusExt processes Q-resolution proofs in the QRP-format[1] and repeatedly calls the QBF solver DepQBF [9] in version 6.03 for deciding PCNF formulas and for producing proofs of modified formulas. In the following, we first introduce the algorithm implemented in QMusExt for extracting unsatisfiable cores, and then discuss the extraction of satisfiable cores.

Algorithm 1 Minimal Unsatisfiable Core Extraction.

```
   Data: False QBF $\Pi.\phi$ in PCNF
   Result: Minimal Unsat Core $V^{i}$
1  (False, $P$) ← QBFSolver ($\Pi.\phi$) with $P = (V^{i} \cup V^{d}, E)$;
2  $P$ ← trim ($P$);
3  while unmarked clauses exists in $V^{i}$ do
4      $C^{i}$ ← pickUnmarkedClause($V^{i}$);
5      (Val, $P'$) ← QBFSolver ($\Pi.unRe(P, C^{i})$);
6      if Val == True then
7          mark $C^{i}$ as a MUS member;
8      else
9          $P''$ ← rebuildProof ($P$, $P'$);
10         $P$ ← trim ($P''$);
11     end
12 end
```

[1] http://fmv.jku.at/qbfcert/qrp.format

3.1 Basic Algorithm for the Extraction of Unsatisfiable Cores

Our tool QMusExt is based on the algorithm for the extraction of minimal unsatisfiable cores for propositional formulas presented in [2]. Whereas the original approach relies on propositional resolution proofs, QMusExt processes Q-resolution proofs. The latter contain not only applications of the axiom and the resolution rule, but also universal reductions. While this has some impacts on the implementation of QMusExt, conceptually the original algorithm is similar.

The approach implemented by QMusExt is summarized in Algorithm 1. The input is a false QBF $\Phi = \Pi.\phi$ in PCNF. First a QBF solver like DepQBF is called. We assume that the QBF solver returns a pair (`Val,` P) where `Val` is the truth value of the solved formula and $P = (V^{\mathsf{i}} \cup V^{\mathsf{d}}, E)$ is a Q-resolution refutation of Φ. In order to ensure that P is non-redundant, the function `trim` is called. Next, QMusExt checks if there is an unmarked clause in V^{i}. If a clause is unmarked it has not been checked so far if it belongs to the MUS. As long as there is one unmarked clause in V^{i}, such a clause C^{i} is selected. Then the QBF $\Pi.\phi'$ is solved where ϕ' consists of the clauses of P (initial and derived clauses) that are not in the cone of C^{i}, i.e., those clauses of P that are unreachable from C^{i}. If $\Phi.\phi'$ is true, then C^{i} is marked as MUS member. Otherwise, the solver returns a refutation P' of $\Pi.\phi'$. This proof does not contain C^{i} as initial clause, but P' could contain clauses from V^{d} as initial clauses which are not part of V^{i} and therefore also not part of the original clause set ϕ. Hence, it is not a proof of $\Pi.\phi$ in general. Based on information from proof P, P' can be modified to a proof P'' such that it contains only initial clauses from V^{i}. In consequence, P'' is a proof of $\Pi.\phi$. This proof P'' contains at least one initial clause less than P (clause C^{i}), but in many cases other clauses other than C^{i} from V^{i} are no initial clauses of P'' as well, because they are not needed to justify initial clauses from P'. In this way, multiple clauses not belonging to the MUS can be eliminated in one step. For the next iteration, P is replaced by a trimmed version of P'', i.e., all clauses that are not connected are removed to ensure that P is non-redundant. The approach is illustrated by the following example.

▶ **Example 3.** Consider the following QBF $\Phi_1 = \Pi.\phi$ which has seven clauses

$$\exists a, b \forall x, y \exists c, d.(\bar{b} \vee x \vee c) \wedge (b \vee x \vee c) \wedge (b \vee \bar{y} \vee \bar{c} \vee \bar{d}) \wedge (\bar{b} \vee y \vee d) \wedge (\bar{a} \vee x) \wedge (a \vee x \vee \bar{c} \vee \bar{d}) \wedge (a \vee x \vee c).$$

The resolution proof P_1 in Figure 1 witnesses that this formula is false. For convenience, the initial clauses V^{i} are labeled by C^{i}_j (with $1 \leq j \leq 7$) and the derived clauses V^{d} are labeled by C^{d}_k (with $1 \leq k \leq 10$). In this resolution graph with $V^{\mathsf{i}} = \phi$ all clauses are connected to the empty clause. Hence, it is already non-redundant.

In the first step, we remove clause $C^{\mathsf{i}}_1 = (\bar{b} \vee x \vee c)$ as well as its cone clauses $C^{\mathsf{d}}_6 = (x \vee c)$, $C^{\mathsf{d}}_8 = (x \vee \bar{y} \vee d)$, $C^{\mathsf{d}}_9 = (x \vee \bar{y})$, and $C^{\mathsf{d}}_{10} = \Box$. Those clauses are highlighted in Figure 1. Now a QBF solver is invoked on all remaining clauses, i.e., on the QBF $\Phi_2 = \Pi.V^{\mathsf{i}} \setminus \{C^{\mathsf{i}}_1\} \cup V^{\mathsf{d}} \setminus \{C^{\mathsf{d}}_6, C^{\mathsf{d}}_8, C^{\mathsf{d}}_9, C^{\mathsf{d}}_{10}\}$.

Also Φ_2 is false. Therefore, we can conclude that the clause $C^{\mathsf{i}}_1 = (\bar{b} \vee x \vee c)$ is not part of the minimal unsatisfiable core and can be removed permanently. The resolution proof P_2 of Φ_2 (the highlighted part of the proof shown in Figure 2 is not a resolution proof of Φ_1 as it contains initial clauses that do not occur in ϕ. However, the proof of Φ_1 can be used to rewrite the proof of Φ_2 such such that it becomes a proof of Φ_1 without using C^{i}_1. In particular, we need to introduce derivations for initial clauses of P_2 that are from V^{d}. These derivations are obtained from P_1. For example, the derivation for C^{d}_4 needs to be added. In the new proof, clause C^{i}_2 is not needed for proving Φ_1. Therefore, it is also not part of the MUS. The new proof has only five initial clauses. In the next five iterations, we find out that none of them can be removed, i.e., all of them are part of the MUS.

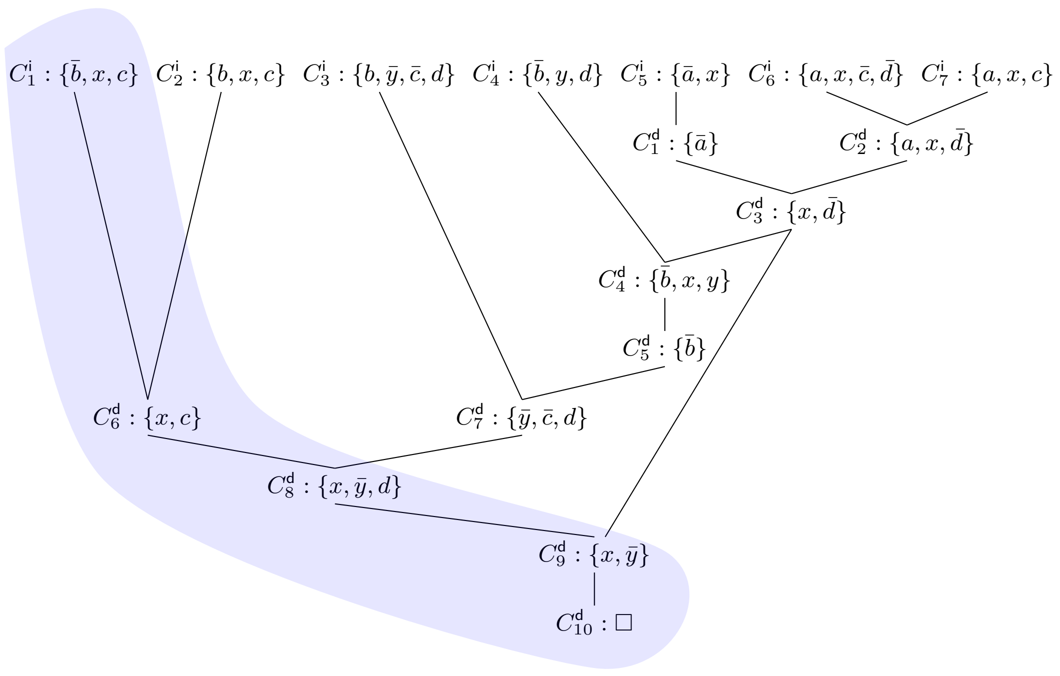

$C^{\mathsf{i}}_1 : \{\bar{b}, x, c\}$
$C^{\mathsf{i}}_2 : \{b, x, c\}$
$C^{\mathsf{i}}_3 : \{b, \bar{y}, \bar{c}, d\}$
$C^{\mathsf{i}}_4 : \{\bar{b}, y, d\}$
$C^{\mathsf{i}}_5 : \{\bar{a}, x\}$
$C^{\mathsf{i}}_6 : \{a, x, \bar{c}, \bar{d}\}$
$C^{\mathsf{i}}_7 : \{a, x, c\}$
$C^{\mathsf{d}}_1 : \{\bar{a}\}$
$C^{\mathsf{d}}_2 : \{a, x, \bar{d}\}$
$C^{\mathsf{d}}_3 : \{x, \bar{d}\}$
$C^{\mathsf{d}}_4 : \{\bar{b}, x, y\}$
$C^{\mathsf{d}}_5 : \{\bar{b}\}$
$C^{\mathsf{d}}_6 : \{x, c\}$
$C^{\mathsf{d}}_7 : \{\bar{y}, \bar{c}, d\}$
$C^{\mathsf{d}}_8 : \{x, \bar{y}, d\}$
$C^{\mathsf{d}}_9 : \{x, \bar{y}\}$
$C^{\mathsf{d}}_{10} : \Box$

Figure 1 Initial resolution refutation for QBF Φ of Example 3 as returned by the QBF solver. Assume that the clause C^{i}_1 is tested for its MUS membership. The highlighted clauses are in the cone of C^{i}_1. Only those clauses which are not highlighted are passed to the QBF solver in the next iteration.

3.2 Extraction of Satisfiable Cores

In contrast to SAT where only unsatisfiable formulas have resolution proofs, also true QBFs have resolution proofs. As QBFs are usually in PCNF, the solver has to provide initial cubes that are satisfying assignments of the matrix, i.e., for a true QBF $\Pi.\phi$ in PCNF, the solver provides a PDNF representation $\Pi.\psi$ from which the empty cube is derived by using the resolution rule and the existential reduction rule. We can now ask the question what is a minimal satisfiable core of $\Pi.\psi$? Our tool can directly answer this question by processing the Q-resolution satisfaction proof in a similar manner as discussed above. Minimal satisfiable cores might be used to find smaller proofs for true formulas. For true formulas, the clausal representation of the input formula is disadvantageous in general, leading to large initial cubes and large proofs. As an effect, the proofs are often very large and also the Skolem functions, the solutions that are extracted from the proofs according to approaches as presented in [1], are large as well.

4 Evaluation

In this section, we evaluate our tool QMusExt on false (true) instances to extract minimal unsatisfiable cores and minimal satisfiable cores. In our implementation we used hash maps as the data structure to store the resolution refutation. This design choice causes a slightly higher memory usage compared to arrays, however tests showed a significant speed up in computation time, due to efficient proof manipulations during the iterations. We also decided to closely interact with DepQBF via API calls, reducing the time needed for the required

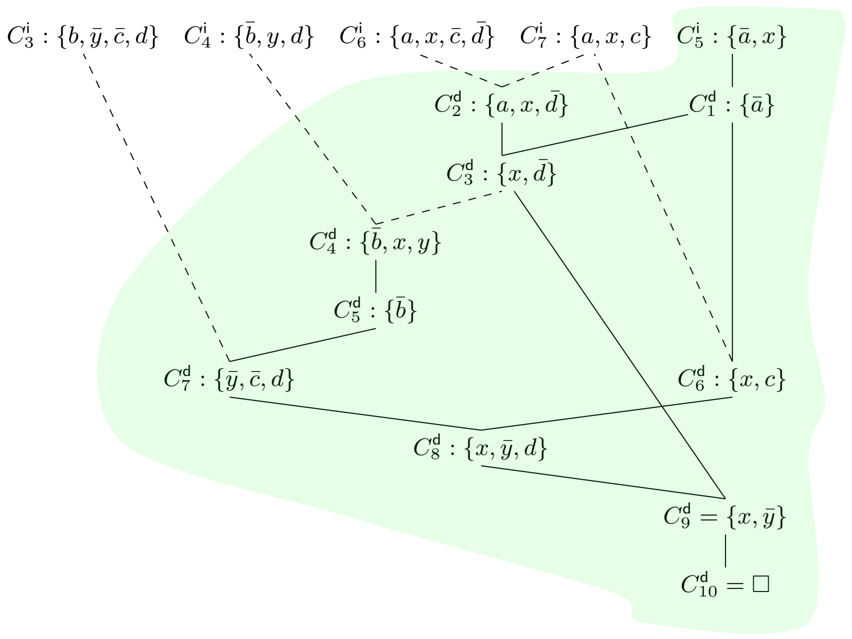

$C^{\mathsf{i}}_3 : \{b, \bar{y}, \bar{c}, d\}$
$C^{\mathsf{i}}_4 : \{\bar{b}, y, d\}$
$C^{\mathsf{i}}_6 : \{a, x, \bar{c}, \bar{d}\}$
$C^{\mathsf{i}}_7 : \{a, x, c\}$
$C^{\mathsf{i}}_5 : \{\bar{a}, x\}$
$C^{\mathsf{d}}_2 : \{a, x, \bar{d}\}$
$C^{\mathsf{d}}_1 : \{\bar{a}\}$
$C^{\mathsf{d}}_3 : \{x, \bar{d}\}$
$C^{\mathsf{d}}_4 : \{\bar{b}, x, y\}$
$C^{\mathsf{d}}_5 : \{\bar{b}\}$
$C^{\mathsf{d}}_7 : \{\bar{y}, \bar{c}, d\}$
$C^{\mathsf{d}}_6 : \{x, c\}$
$C^{\mathsf{d}}_8 : \{x, \bar{y}, d\}$
$C^{\mathsf{d}}_9 = \{x, \bar{y}\}$
$C^{\mathsf{d}}_{10} = \square$

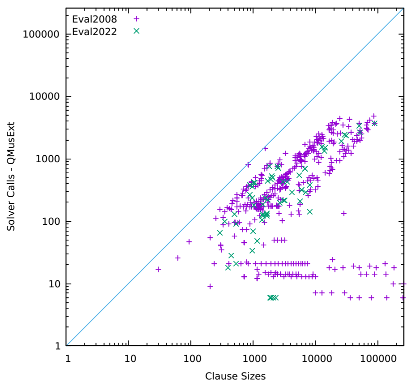
Figure 2 Rewritten resolution refutation after one iteration. The highlighted part is the proof for the formula that contains the clauses not reachable from C^{i}_1 (the clauses not highlighted in the proof above). Hence, this is not a proof of Φ. The dashed edges and vertices from the matrix of Φ are added in order to obtain a proof for Φ. This proof does not include C^{i}_1 as initial clause. Further, it does not include C^{i}_2.

solver calls. All experiments were run on a cluster of dual-socket AMD EPYC 7313 @ 16 $\times$ 3.7GHz machines with 4GB memory limit and 1800 seconds as timeout. All experimental data is available at the webpage of our tool.

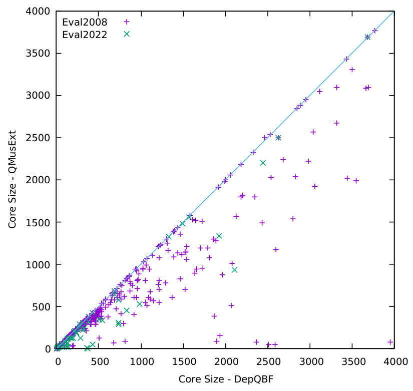
4.1 Extraction of Unsatisfiable Cores

For MUS extraction, we consider the formulas of the PCNF track of the QBFEval 2022 and of the QBFEval 2008. All formulas are available at QBFLib.[2] To identify false formulas we run DepQBF [9] in standard configuration and selected all false formulas that could be solved within a time limit of 1800 seconds. Out of 1141 formulas of the eval2008 benchmark set (resp. 259 of the eval2022 benchmark set) 683 (resp. 137) formulas were found to be false. For the 2008 benchmarks, QMusExt could find the MUSes of 436 formulas with an average size of 533.50 clauses while the proofs contain 650.58 initial clauses and the original PCNFs 16903.04 clauses on average. For the 2022 benchmarks, QMusExt could find the MUSes of 62 formulas with an average size of 406.63 clauses while the proofs contain 500.53 initial clauses and the original PCNFs 12270.57 clauses on average. Hence we observe an decrease of 96.84 % and 96.68 % of used clauses compared to the initial clauses and a reduction in proof clauses of 18.00 % and 18.76 %. The reductions are summarized in Table 2. The average runtime for successful executions was 120.67 seconds for the 2008 benchmarks and 131.01 seconds for the 2022 benchmarks. On average 444.70 and 265.81 solver calls were needed

[2] `http://www.qbflib.org`

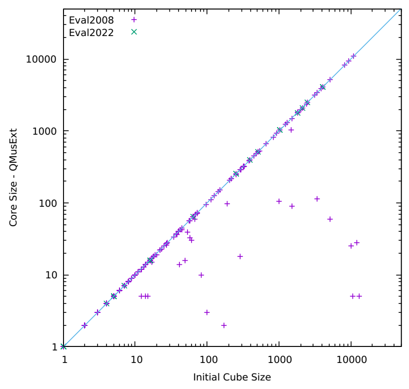


(a) Number of solver calls per formula within QMusExt.

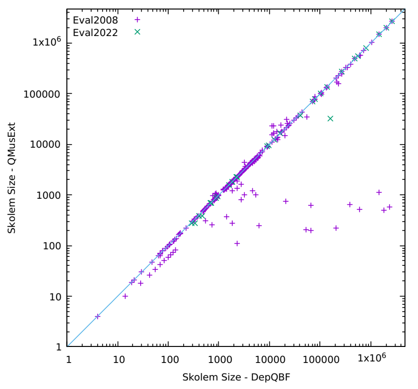


(b) Comparison of MUS sizes produced by DepQBF vs. MUS sizes produced by QMusExt.

Figure 3 Results for false formulas.



(a) Size reduction of the initial cubes from Q-resolution satisfaction proofs.



(b) Size comparison of Skolem functions from initial proof vs Skolem functions from satisfiable core proofs.

Figure 4 Results for true formulas.

to find MUSes. Figure 3a relates number of solver calls and clause sizes. In the worst case, only one clause is eliminated per iteration, i.e., the approach is linear in the formula size. In practice, fewer calls are need indicating the scalability of the approach.

We compared QMusExt to the approach implemented in DepQBF [8] and the recent approach SMUSer [12] that computes minimum unsatisfiable cores, i.e., a MUS with the smallest possible cardinality. As finding the smallest MUSes is a computationally harder problem than finding any MUS, it is not surprising that SMUSer finds fewer MUSes compared to the other two approaches within the given time limit. In particular, it finds the smallest MUS for 32 formulas form the 2008 benchmark set. For the 2022 benchmark set, we did not obtain any result from SMUSer. For all of the formulas for which SMUSer could find a result, also QMusExt and DepQBF found MUSes. For 24 of these, our tool found cores of the same size as the cores found by SMUSer. The others differ by 18 clauses at most. Table 1 summarizes the number of solved instances and the average core sizes.

DepQBF is able to find MUSes of 679 (benchmarks from 2008) and of 134 (formulas from 2022). It is not surprising that DepQBF finds more MUSes with the incremental approach than QMusExt, although QMusExt also relies on DepQBF internally. For the algorithm

Table 1 Comparison of the evaluation results.

	solved instances				average core size			
	FALSE		TRUE		FALSE		TRUE	
	eval08 683 inst.	eval22 137 inst.	eval08 458 inst.	eval22 122 inst.	eval08	eval22	eval08	eval22
QMusExt	436	62	253	31	534	407	316	589
DepQBF	679	134	–	–	2438	1695	–	–
SMUSer	32	–	–	–	146	–	–	–

implemented in QMusExt, proofs have to be generated, analyzed and rewritten. If proof generation is enabled, certain pruning techniques have to be disabled slowing down the solving process. Further, the proof size can be very large, requiring the implementation of efficient hashing techniques for finding nodes in the resolution graph. When we compare the sizes of the MUSes produced by DepQBF to the sizes of the MUSes produced by QMusExt as done in Figure 3b we see that the cores found by QMusExt are of equal size or smaller.

4.2 Extraction of Satisfiable Cores

We also applied QMusExt on true formulas and observe it it can also find minimal satisfiable cores of the PDNF. For our experiments, we selected those formulas from the 2022 formulas and, respectively, from the 2008 formulas, which could be solved by DepQBF in 1800 seconds. Out of 458 (122) true formulas, our tool could find satisfiable cores for 253 and, respectively, 31 formulas. Out of these, 28 could by decreased in average by 61.32 %. Figure 4a shows the reduction for the individual formulas. In the most extreme case, a PDNF with 10096 cubes could be reduced to a PDNF with 25 cubes. We also calculated the Skolem functions from the original set of initial cubes as well as from the formula reduced to a minimal satisfiable core. The result is shown in Figure 4b. The Skolem functions are extracted with the QBFCert framework [11] and represented as And-Inverter Graphs in the Aiger format.[3] We measure the size in terms of gate numbers. In some cases, we observe a slight increase in the size of the Skolem functions, while there are also cases where the size could be considerably decreased. In the most extreme case, the Skolem function could be reduced by 99.98 %. Details are summarized in Table 1 and Table 2.

Table 2 Average size of cores generated by QMusExt (core), average formula sizes (formula), average number of axiom clauses/cubes (proof), reductions when applying QMusExt and average run time of QMusExt.

		avg. size			reductions		
		formula	proof	core	formula size	proof	avg. run time (s)
FALSE	eval08	651	16904	534	96.84%	17.97%	120.67
	eval22	501	12271	407	96.68%	18.76%	131.01
TRUE	eval08	840	17950	316	98.24%	62.38%	439.11
	eval22	595	70213	589	99.16%	1.01%	179.15

[3] `http://fmv.jku.at/aiger/`

5 Conclusion

We presented QMusExt, the first tool that implements the extraction of unsatisfiable cores of false QBFs based on Q-resolution proofs. Originally, the approach was successfully applied for SAT [2]. Our experiments indicate that the approach is also promising for QBFs. In particular, we could observe that the number of necessary solver calls is smaller than the number of clauses of the input formula. Not surprisingly, an approach based on selector variables implemented with the incremental interface of the QBF solver DepQBF is more efficient in terms of runtime. However, the approach of QMusExt finds smaller unsatisfiable cores in many cases. Further, due to the duality of false and true QBFs, the tool can be be applied for the extraction of satisfiable cores from PDNF formulas as produced by solvers as well.

In the future we plan to adapt optimizations of the basic algorithm as proposed in [10] for QBFs and combine Q-resolution based approaches with approaches based on selector variables. In addition, we further plan to investigate the pruning potential of proofs and function extraction.

This work has been supported by the LIT AI Lab funded by the State of Upper Austria.

References

1 Valeriy Balabanov and Jie-Hong R. Jiang. Unified QBF certification and its applications. *Formal Methods Syst. Des.*, 41(1):45–65, 2012. doi:10.1007/s10703-012-0152-6.

2 Nachum Dershowitz, Ziyad Hanna, and Alexander Nadel. A scalable algorithm for minimal unsatisfiable core extraction. In *Proc. of the 9th Int. Conf. on Theory and Applications of Satisfiability Testing (SAT 2006)*, volume 4121 of *Lecture Notes in Computer Science*, pages 36–41. Springer, 2006. doi:10.1007/11814948_5.

3 Enrico Giunchiglia, Massimo Narizzano, and Armando Tacchella. Clause/term resolution and learning in the evaluation of quantified boolean formulas. *J. Artif. Intell. Res.*, 26:371–416, 2006. doi:10.1613/jair.1959.

4 Alexey Ignatiev, Mikolás Janota, and João Marques-Silva. Quantified maximum satisfiability. *Constraints*, 21(2):277–302, 2016. doi:10.1007/s10601-015-9195-9.

5 Hans Kleine Büning, Marek Karpinski, and Andreas Flögel. Resolution for quantified boolean formulas. *Inf. Comput.*, 117(1):12–18, 1995. doi:10.1006/inco.1995.1025.

6 Hans Kleine Büning and Oliver Kullmann. Minimal unsatisfiability and autarkies. In *Handbook of Satisfiability – Second Edition*, volume 336 of *Frontiers in Artificial Intelligence and Applications*, pages 571–633. IOS Press, 2021.

7 Hans Kleine Büning and Xishun Zhao. Minimal false quantified boolean formulas. In Armin Biere and Carla P. Gomes, editors, *Proc. of the 9th Int. Conf. on Theory and Applications of Satisfiability Testing (SAT 2006)*, volume 4121 of *Lecture Notes in Computer Science*, pages 339–352. Springer, 2006.

8 Florian Lonsing and Uwe Egly. Incrementally computing minimal unsatisfiable cores of qbfs via a clause group solver API. In *Proc. of the 18th Int. Conf. on Theory and Applications of Satisfiabily Testing (SAT 2015)*, volume 9340 of *Lecture Notes in Computer Science*, pages 191–198. Springer, 2015. doi:10.1007/978-3-319-24318-4_14.

9 Florian Lonsing and Uwe Egly. Depqbf 6.0: A search-based QBF solver beyond traditional QCDCL. In *Proc. of the 26th Conf. on Automated Deduction (CADE 26)*, volume 10395 of *Lecture Notes in Computer Science*, pages 371–384. Springer, 2017. doi:10.1007/978-3-319-63046-5_23.

10 Alexander Nadel. Boosting minimal unsatisfiable core extraction. In *Proc. of 10th Int. Conf. on Formal Methods in Computer-Aided Design (FMCAD 2010)*, pages 221–229. IEEE, 2010. URL: https://ieeexplore.ieee.org/document/5770953/.

11 Aina Niemetz, Mathias Preiner, Florian Lonsing, Martina Seidl, and Armin Biere. Resolution-based certificate extraction for QBF – (tool presentation). In *Proc. of the 15th Int. Conf. on Theory and Applications of Satisfiability Testing (SAT 2012)*, volume 7317 of *Lecture Notes in Computer Science*, pages 430–435. Springer, 2012.

12 Andreas Niskanen, Jere Mustonen, Jeremias Berg, and Matti Järvisalo. Computing smallest muses of quantified boolean formulas. In *Proc. of the 16th Int. Conf. on Logic Programming and Nonmonotonic Reasoning (LPNMR 2022)*, volume 13416 of *Lecture Notes in Computer Science*, pages 301–314. Springer, 2022. doi:10.1007/978-3-031-15707-3_23.

13 Yinlei Yu and Sharad Malik. Validating the result of a quantified boolean formula (QBF) solver: theory and practice. In *Proc. of the 2005 Conf. on Asia South Pacific Design Automation, (ASP-DAC 2005)*, pages 1047–1051. ACM Press, 2005.