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) II.ϕ in prenex conjunctive normal form. An MUS is a subformula of ϕ′ ⊆ ϕ such that II.ϕ′ 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 straightforward 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].
2 Preliminaries

We consider QBFs of the form $\Pi \phi$, where $\Pi = Q_1 X_1 ... Q_n X_n$ 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 $\phi$ 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 conjunction 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 $\text{Var}(l) = x$ 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, $\hat{l} = x$ if $l = \bar{x}$ and $\hat{l} = \bar{x}$ if $l = x$. A quantifier prefix $Q_1 X_1 \ldots Q_n X_n$ imposes an ordering $\prec$ on the variables: if $x_i \in X_i$, $x_j \in X_j$, and $i < j$, then $x_i \prec x_j$. For a propositional formula $\phi$, $\phi_l$ 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 $\forall X'. \phi$ is true iff $\forall X'. \phi_x$ and $\forall X'. \phi_{\bar{x}}$ are true where $X' = X \setminus \{x\}$. Respectively, a QBF $\exists X'. \phi$ is true iff $\exists X'. \phi_x$ or $\exists X'. \phi_{\bar{x}}$ is true. For example $\forall x \exists y. (x \lor y) \land (\bar{x} \lor \bar{y})$ is true and $\exists x \forall y. (x \lor y) \land (\bar{x} \lor \bar{y})$ 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 $\phi$ can be derived.
- **Resolution:** From already derived clauses $C \lor x$ and $D \lor \bar{x}$, a clause $C \lor D$ can be derived if there is no literal $l$ with $l \in C \cup D$, $x \not\in D$, $\bar{x} \not\in C$, and $x$ is existentially quantified.
- **Universal Reduction:** From an already derived clause $C \lor l$, 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 $\square$ 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 $\sigma$ be a satisfying assignment of $\phi$. Then cube $\bigwedge_{l \in \sigma} l$ can be derived.
- **Resolution:** From already derived cubes $C \land x$ and $D \land \bar{x}$, a cube $C \land D$ can be derived if there is no literal $l$ with $l \in C \cup D$, $x \not\in D$, $\bar{x} \not\in C$, and $x$ is universally quantified.
- **Existential Reduction:** From an already derived cube $C \land l$, 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 $V = V^i \cup V^d$ consists of initial clauses $V^i \subseteq \phi$ 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 $\square$. Resolution graphs for true QBFs are defined respectively.
A vertex $D$ is called reachable in resolution graph $P$ from a vertex $C$ if 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 $\Phi$, if $\Pi.\phi'$ is false and $\Pi.\phi'\{C\}$ 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 $\exists x \forall y \exists z . (x \lor y \lor z) \land (\bar{x} \lor \bar{y} \lor \bar{z})$ has minimal unsatisfiable cores $((x \lor y \lor z) \land (\bar{x} \lor \bar{y} \lor \bar{z})), (\exists z) \land (z \lor y)$, and $((x \lor y \lor z) \land (\bar{x} \lor \bar{y} \lor (x \lor y \lor z))$.

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 $\Phi$, if $\Pi.\phi'$ is true and $\Pi.\phi'\{C\}$ 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 \hspace{1em} $C^i ←$ pickUnmarkedClause($V^i$);
5 \hspace{1em} (Val, $P'$) ← QBFSolver ($\Pi.\text{unRe}(P, C^i)$);
6 \hspace{1em} if Val == True then
7 \hspace{2em} mark $C^i$ as a MUS member;
8 \hspace{1em} else
9 \hspace{2em} $P'' ←$ rebuildProof ($P$, $P'$);
10 \hspace{2em} $P ←$ trim ($P''$);
11 end
12 end

\(^1\) [http://fmv.jku.at/qbfcert/qrp.format](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 \(P = (V^1 \cup V^d, E)\) is a Q-resolution refutation of \(\Phi\). In order to ensure that \(P\) is non-redundant, the function trim is called. Next, QMusExt checks if there is an unmarked clause in \(V^1\). 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 \(V^1\), such a clause \(C^i\) is selected. Then the QBF \(\Pi.\phi'\) is solved where \(\phi'\) 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 \(V^1\) and therefore also not part of the original clause set \(\phi\). 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 \(V^1\). 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 \(V^1\) 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

\[
\exists a, b \forall x, y \exists c, d. (b \lor x \lor c) \land (b \lor x \lor c) \land (b \lor y \lor c \lor d) \land (b \lor y \lor d) \land (a \lor x) \land (a \lor x \lor e \lor d) \land (a \lor x \lor e).
\]

The proof graph \(P_1\) in Figure 1 witnesses that this formula is false. For convenience, the initial clauses \(V^1\) are labeled by \(C^j\) (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 \(V^1 = \phi\) all clauses are connected to the empty clause. Hence, it is already non-redundant.

In the first step, we remove clause \(C^1_1 = (b \lor x \lor c)\) as well as its cone clauses \(C^d_6 = (x \lor c), C^d_8 = (x \lor y \lor d), C^d_9 = (x \lor y),\) and \(C^d_{10} = \square\). Those clauses are highlighted in Figure 1. Now a QBF solver is invoked on all remaining clauses, i.e., on the QBF \(\Phi_2 = \Pi.\neg V^1 \setminus \{C^1_1\} \cup V^d \setminus \{C^d_6, C^d_8, C^d_9, C^d_{10}\}\).

Also \(\Phi_2\) is false. Therefore, we can conclude that the clause \(C^1_2 = (b \lor x \lor c)\) is not part of the minimal unsatisfiable core and can be removed permanently. The resolution proof \(P_2\) of \(\Phi_2\) (the highlighted part of the proof shown in Figure 2) is not a resolution proof of \(\Phi_1\) as it contains initial clauses that do not occur in \(\phi\). However, the proof of \(\Phi_1\) can be used to rewrite the proof of \(\Phi_2\) such that it becomes a proof of \(\Phi_1\) without using \(C^1_1\). 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 \(C^d_{10}\) needs to be added. In the new proof, clause \(C^1_2\) is not needed for proving \(\Phi_1\). 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.
Assume that the clause $C_1$ is tested for its MUS membership. The highlighted clauses are in the cone of $C_1$. 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
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 $\Phi$. The dashed edges and vertices from the matrix of $\Phi$ are added in order to obtain a proof for $\Phi$. 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 × 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. 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 a 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.

http://www.qbflib.org
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.

<table>
<thead>
<tr>
<th></th>
<th>solved instances</th>
<th>average core size</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>FALSE</td>
<td>TRUE</td>
</tr>
<tr>
<td></td>
<td>FALSE</td>
<td>TRUE</td>
</tr>
<tr>
<td>eval08</td>
<td>683 inst.</td>
<td>137 inst.</td>
</tr>
<tr>
<td>eval22</td>
<td>436</td>
<td>62</td>
</tr>
<tr>
<td>QMusExt</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>534</td>
<td>407</td>
</tr>
<tr>
<td>DepQBF</td>
<td>679</td>
<td>134</td>
</tr>
<tr>
<td></td>
<td>2438</td>
<td>1695</td>
</tr>
<tr>
<td>SMUSer</td>
<td>32</td>
<td></td>
</tr>
<tr>
<td></td>
<td>146</td>
<td></td>
</tr>
</tbody>
</table>

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. 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.

<table>
<thead>
<tr>
<th></th>
<th>avg. size</th>
<th>reductions</th>
<th>avg. run time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>formula</td>
<td>proof</td>
<td>core</td>
</tr>
<tr>
<td>FALSE</td>
<td>eval08</td>
<td>651</td>
<td>16904</td>
</tr>
<tr>
<td></td>
<td>eval22</td>
<td>501</td>
<td>12271</td>
</tr>
<tr>
<td>TRUE</td>
<td>eval08</td>
<td>840</td>
<td>17950</td>
</tr>
<tr>
<td></td>
<td>eval22</td>
<td>595</td>
<td>70213</td>
</tr>
</tbody>
</table>

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

