Formula Preprocessing in MUS Extraction*

Anton Belov[1], Matti Järvisalo[2], and Joao Marques-Silva[1,3]

[1] Complex and Adaptive Systems Laboratory, University College Dublin, Ireland
[2] HIIT & Department of Computer Science, University of Helsinki, Finland
[3] IST/INESC-ID, Lisbon, Portugal

Abstract. Efficient algorithms for extracting minimally unsatisfiable subformulas (MUSes) of Boolean formulas find a wide range of applications in the analysis of systems, e.g., hardware and software bounded model checking. In this paper we study the applicability of preprocessing techniques for Boolean satisfiability (SAT) in the context of MUS extraction. Preprocessing has proven to be extremely important in enabling more efficient SAT solving. Hence the study of the applicability and the effectiveness of preprocessing in MUS extraction is highly relevant. Considering the extraction of both standard and group MUSes, we focus on a number of SAT preprocessing techniques, and formally prove to what extent the techniques can be directly applied in the context of MUS extraction. Furthermore, we develop a generic theoretical framework that captures MUS extraction problems, and enables formalizing conditions for correctness-preserving applications of preprocessing techniques that are not applicable directly. We experimentally evaluate the effect of preprocessing in the context of group MUS extraction.

1 Introduction

Efficient algorithms for extracting minimally unsatisfiable subformulas (MUSes) of Boolean formulas find a wide range of applications in the analysis of systems, e.g., hardware and software bounded model checking. A variety of different approaches to MUS extraction has been proposed, see [5, 9, 22, 21, 23, 19] for recent examples and [20] for a survey. Typically the state-of-the-art MUS extraction algorithms use Boolean satisfiability (SAT) solvers as NP-oracles for checking the satisfiability of subformulas in an iterative manner.

In recent years, formula preprocessing has emerged as an extremely important technique in enabling efficient SAT solving (see e.g. [6, 8, 10, 7, 11, 1, 12, 15]). Thus, in this paper, we study of the applicability and the effectiveness of preprocessing in the context of MUS extraction.

The result of MUS extraction on a preprocessed input formula F' is an MUS M' of F'. However, since preprocessing changes the formula structure by, e.g., removing clauses and removing or adding literals to clauses, M' is, in general, *not* an MUS of F.

* The first and third authors are financially supported by SFI PI grant BEACON (09/IN.1/I2618), and by FCT grants ATTEST (CMU-PT/ELE/0009/2009) and POLARIS (PTDC/EIA-CCO/123051/2010). The second author is financially supported by Academy of Finland (grants 132812 and 251170).

Hence we are faced with the problem of *reconstructing* an MUS of F from M'. Considering the whole MUS extraction process, in order to benefit from preprocessing, this reconstruction must be performed efficiently. However, even guaranteeing correctness (i.e., ensuring that the reconstructed subformulas are actually MUSes) when applying preprocessing becomes non-trivial. This is especially true for the recently introduced problem of *group* (or *high-level*) [18, 22] MUS extraction, which is practically a very relevant generalization of the "plain" MUS extraction problem.

Considering the extraction of both standard and group MUSes, we focus on a number of important SAT preprocessing techniques, including *clause elimination procedures* [7, 12] such as *subsumption* and *blocked clause elimination* [14], and resolution-based preprocessing techniques (SatElite-style *variable elimination* [6], *self-subsuming resolution*, and *Boolean constraint propagation*). We show formally to what extent the techniques can be directly applied in the context of MUS extraction. It turns out that, especially in the case of group MUS extraction, maintaining correctness under preprocessing needs extra attention. This is further corroborated by the fact that incorrect results produced by some group MUS extractors that applied preprocessing in the special track of the 2011 SAT Competition on group MUS extraction were likely due to incorrect applications of standard SAT preprocessing techniques (see `http://www.satcompetition.org/2011/` for details). We develop a generic theoretical framework based on *labelled CNFs*, which provides a unifying view to variants of MUS extraction problems, and enables formalizing conditions for correctness-preserving applications of preprocessing techniques that are not applicable directly. Additionally, we experimentally evaluate the effect of preprocessing in the context of group MUS extraction.

2 Preliminaries

For a Boolean variable x, there are two *literals*, the positive literal, denoted by x, and the negative literal, denoted by $\bar{x}$. A *clause* is a disjunction of literals and a CNF formula a conjunction of clauses. A clause can be seen as a finite set of literals and a CNF formula as a finite set of clauses. A *unit clause* contains exactly one literal. A clause is a *tautology* if it contains both x and $\bar{x}$ for some variable x. A clause C is *subsumed* by a clause $C' \subset C$ (viewed as sets of literals). A truth assignment for a CNF formula F is a function τ that maps variables in F to $\{0, 1\}$. If $\tau(x) = v$, then $\tau(\bar{x}) = \bar{v}$, where $\bar{1} = 0$ and $\bar{0} = 1$. A clause C is satisfied by τ if $\tau(l) = 1$ for some $l \in C$. An assignment τ satisfies F if it satisfies every clause in F. A CNF formula is *satisfiable* if there is an assignment that satisfies it, and *unsatisfiable* otherwise. We denote the set of all unsatisfiable and satisfiable CNF formulas, resp., by UNSAT and SAT, resp. Two CNF formulas F and F' are *equisatisfiable* if we have that $F \in \mathsf{SAT}$ iff $F' \in \mathsf{SAT}$.

Minimal Unsatisfiability A CNF formula F is *minimally unsatisfiable* if (i) $F \in \mathsf{UNSAT}$, and (ii) for any clause $C \in F$, $F \setminus \{C\} \in \mathsf{SAT}$. We denote the set of minimally unsatisfiable CNF formulas by MU. A CNF formula F' is a *minimally unsatisfiable subformula (MUS)* of a formula F if $F' \subseteq F$ and $F' \in \mathsf{MU}$. The set of MUSes of a CNF formula F is denoted by $\mathsf{MUS}(F)$. In general, a given unsatisfiable formula F may contain more than one MUS.

Motivated by several industrially relevant applications, minimal unsatisfiability and related concepts have been extended to CNF formulas where clauses are partitioned into disjoint sets called *groups* [18, 22].

Definition 1. *Given an explicitly partitioned unsatisfiable CNF formula* $G = G_0 \cup G_1 \cup \cdots \cup G_k$ *(a* group MUS instance *or* group CNF formula*), where the* G_i*'s are pair-wise disjoint sets of clauses called* groups*, a group MUS of* G *is a subset* $\mathcal{G}$ *of* $\{G_1, \ldots, G_k\}$ *such that (i)* $G_0 \cup \bigcup \mathcal{G}$ *(seen as a monolithic CNF formula) is unsatisfiable, and (ii) for any group* $G \in \mathcal{G}$*,* $G_0 \cup \bigcup \left(\mathcal{G} \setminus \{G\}\right)$ *is satisfiable. The set of group MUSes of a group MUS instance* G *is denoted by* $\mathsf{GMUS}(G)$*.*

Clause Elimination Procedures Given a CNF formula F, a *clause elimination procedure* E is an algorithm that on input F returns a CNF formula $E(F) \subseteq F$ that is equisatisfiable with F. A specific clause elimination procedure E removes clauses satisfying a *specific (typically polynomial-time computable) redundancy property* P_E from F until fixpoint. In other words, E on input F modifies F by repeating the following: if there is a clause $C \in F$ satisfying P_E, let $F := F \setminus \{C\}$.

An example of a clause elimination procedure is *blocked clause elimination* (BCE), which removes so-called *blocked clauses* [16] from CNF formulas until fixpoint. A literal l in a clause C of a CNF formula F blocks C (with respect to F) if for every clause $C' \in F$ with $\bar{l} \in C'$, the resolvent $(C \setminus \{l\}) \cup (C' \setminus \{\bar{l}\})$ obtained from resolving C and C' on l is a tautology. A clause is blocked (with respect to a fixed CNF formula) if it has a literal that blocks it. Note that clauses that contain pure literals are blocked [14]. Additional well-known clause elimination procedures include *tautology elimination* (removing tautological clauses) and *subsumption elimination* (removing subsumed clauses). These and other more involved clause elimination procedures are analyzed in the context of CNF satisfiability in [12, 13].

Resolution-Based Preprocessing The resolution rule states that, given two clauses $C_1 = (l \vee A)$ and $C_2 = (\bar{l} \vee B)$, the implied clause $C = (A \vee B)$, called the *resolvent* of C_1 and C_2, can be inferred by *resolving* on the literal l. We write $C = C_1 \otimes_l C_2$. This is lifted to two sets S_l and $S_{\bar{l}}$ of clauses that all contain the literal l and $\bar{l}$, resp., by $S_l \otimes_l S_{\bar{l}} = \{C_1 \otimes_l C_2 \mid C_1 \in S_l, C_2 \in S_{\bar{l}},$ and $C_1 \otimes_l C_2$ is not a tautology$\}$.

Variable Elimination (VE) [6] is defined following the Davis-Putnam procedure (DP). The elimination of a variable x in the whole CNF can be computed by pair-wise resolving each clause in S_x with every clause in $S_{\bar{x}}$. Replacing the original clauses in $S_x \cup S_{\bar{x}}$ with the set of *non-tautological* resolvents $S = S_x \otimes_x S_{\bar{x}}$ gives the CNF $(F \setminus (S_x \cup S_{\bar{x}})) \cup S$ which is equisatisfiable with F. In order to avoid exponential worst-case space complexity, VE is bounded typically as follows: a variable x is allowed to be eliminated only if $|S| \leq |S_x \cup S_{\bar{x}}| + \Delta$, i.e., the resulting CNF formula $(F \setminus (S_x \cup S_{\bar{x}})) \cup S$ will not contain more than a constant Δ more clauses than the original formula F (typically $\Delta = 0$ [6]). VE is currently one of the most important SAT preprocessing techniques, as witnessed by e.g. the SatElite preprocessor [6].

In the following, we will consider individual steps of variable elimination. Given a CNF formula F, the result of eliminating the variable x from F is $\mathsf{VE}(F, x) = (F \setminus (S_x \cup S_{\bar{x}})) \cup (S_x \otimes_x S_{\bar{x}})$. Note that in the case x appears in one polarity only (i.e., x is a pure literal), this operation simply removes all clauses that contain x.

Unit propagation Given a CNF formula F, unit propagation on F refers to applying the following steps on F until fixpoint: if there is a unit clause (l) in F, remove all clauses with the literal l from F, and remove the literal $\bar{l}$ from all clauses in F. We will consider individual steps of unit propagation on a CNF formula F, where a single literal l is propagated: $\mathsf{BCP}(F, l) = \{C \in F \mid C \cap \{l, \bar{l}\} = \emptyset\} \cup \{C \setminus \{\bar{l}\} \mid C \in F,\ \bar{l} \in C\}$.

Self-Subsuming Resolution (SSR) Given a CNF formula F, the self-subsuming resolution rule states that, given two clauses $C, D \in F$ such that $l \in C$ and $\bar{l} \in D$ for some literal l, and D is subsumed by $C \otimes_l D$, D can be *replaced* with $C \otimes_l D$ in F (or, informally, $\bar{l}$ can be removed from D). Hence a step of self-subsuming resolution, resolving C and D on l, results in the formula $\mathsf{SSR}(F, C, D, l) = (F \setminus D) \cup \{C \otimes_l D\}$. Regarding the practical importance of SSR, as noted in [6], applying SSR in combination with VE and subsumption elimination can give notable improvements w.r.t. applying VE alone.

3 Direct Preprocessing in MUS Extraction

In this section we address the question of the *direct* applicability of CNF preprocessing techniques described in Sect. 2 in the context of MUS extraction. That is, whether we can simply apply a technique to a formula F (keeping track of the changes), extract an MUS of the preprocessed formula, and reconstruct an MUS of F from it in an efficient and natural way.

3.1 Clause Elimination Procedures

Plain MUS Extraction It is rather straightforward to observe that clause elimination procedures can be directly applied in the context of plain MUS extraction: for any MUS M of a CNF formula F', such that F' is the result of applying any combination of clause elimination procedures on an input CNF F, we have that M is an MUS of F.

Proposition 1. *If F' is a result of applications of clause elimination procedures to an unsatisfiable CNF formula F, then* $\mathsf{MUS}(F') \subseteq \mathsf{MUS}(F)$.

Proof. Since $F' \subseteq F$, we have $M \subseteq F$ for any $M \in \mathsf{MUS}(F')$. Furthermore, since $M \in \mathsf{MU}$, we have $M \in \mathsf{MUS}(F)$. □

Note the inclusion instead of equality in Proposition 1: consider $F = F_1 \cup F_2$ such that $F_1, F_2 \in \mathsf{MU}$, $F_1 \cap F_2 = \emptyset$. Since both F_1 and F_2 are unsatisfiable on their own, there is a clause elimination procedure that removes all clauses either in F_1 or F_2 from F.

Group MUS Extraction We say that a clause elimination procedure S is applied on a group MUS instance $G = \{G_0, G_1, \ldots, G_n\}$ when referring to applying S on G seen as the monolithic CNF formula $\bigcup_{i=0}^{n} G_i$. The resulting group MUS instance is $S(G) = \{G'_0, G'_1, \ldots, G'_n\}$, where for each $i = 0..n$ we have $G'_i = G_i \cap S(\bigcup_{i=0}^{n} G_i)$. A natural idea for reconstructing a GMUS M of G from a GMUS M' of $S(G)$ would be to consider $M = \{G_i \in G \mid G'_i \in M'\}$. However, we will show that this natural idea does not generally work: whether M is always guaranteed to be a GMUS of G depends critically on the choice of the clause elimination procedure S. Surprisingly, even subsumption elimination is problematic, as witnessed by the following example.

Example 1. Consider the group MUS instance $G = \{G_0, G_1, G_2\}$, where $G_0 = \{(\bar{r})\}$, $G_1 = \{(p \vee q), (\bar{q} \vee r), (\bar{p} \vee r)\}$, and $G_2 = \{(p)\}$. Here $\{G_1\}$ is the only group MUS. Here $(p) \in G_2$ subsumes $(p \vee q) \in G_1$. However, $\{G'_1\}$ with $G'_1 = G_1 \setminus \{(p \vee q)\}$ is not a group MUS of $\{G'_0 = G_0, G'_1, G'_2 = G_2\}$ since $G_0 \cup G'_1 \in \mathsf{SAT}$; $\{G'_1, G'_2\}$ is. ■

A similar proposition to that of Proposition 1 in the context of group MUS extraction can be shown for the restricted case of what we call "monotonic" clause elimination procedures: a clause elimination procedure S is monotonic if, for any two CNFs F and F' s.t. $F' \subseteq F$, we have that $S(F') \subseteq S(F)$.

Proposition 2. *Let $G = \{G_0, G_1, \ldots, G_k\}$ be a group MUS instance, and S any monotonic clause elimination procedure. For any GMUS $M' \subseteq S(G)$ of the group MUS instance $S(G) = \{G'_0, G'_1, \ldots, G'_k\}$ obtained from applying S on G, $M = \{G_i \in G \mid G'_i \in M'\}$ is a GMUS of G.*

Proof. Assume that M is not a group MUS of G. Take any group MUS $M'' \subset M$ of M. The monolithic CNF formula $S(M'')$ is unsatisfiable. Since S is monotonic, for each group, say G''_i, in $S(M'')$, M' contains a group G'_i that is a superset of G''_i. Furthermore, since $M'' \subset M$, there is a group in M' that is not a superset of any group in $S(M'')$. It follows that M' is not a group MUS of G', which is a contradiction. □

In other words, any monotonic clause elimination procedure can be safely used for preprocessing in the context of plain MUS and group MUS extraction. In addition to tautology elimination, this includes, e.g., blocked clause elimination.

Proposition 3. *Let $G = \{G_0, G_1, \ldots, G_k\}$ be a group MUS instance. For any GMUS $M' \subseteq \mathsf{BCE}(G)$ of the group MUS instance $\mathsf{BCE}(G) = \{G'_0, G'_1, \ldots, G'_k\}$ obtained from applying BCE on G, $M = \{G_i \in G \mid G'_i \in M'\}$ is a GMUS of G.*

Proof. By Proposition 2, it is enough to show that BCE is monotonic. Recall that a literal l in a clause C of a CNF formula F blocks C (with respect to F) if for *every* clause $C' \in F$ with $\bar{l} \in C'$, the resolvent $(C \setminus \{l\}) \cup (C' \setminus \{\bar{l}\})$ is a tautology. Note that, in particular, l blocks C if $\bar{l}$ does not appear in any clause of F (i.e. l is pure). Hence, if l blocks C wrt F, then l blocks C wrt any $F' \subseteq F$, and thus $\mathsf{BCE}(F') \subseteq \mathsf{BCE}(F)$. □

Furthermore, *pure literal elimination* (PLE) is also covered. The CNF formula $\mathsf{PLE}(F)$ resulting from applying pure literal elimination on F is the formula at the fixpoint of the following: while there is a pure literal l in F, let $F := F \setminus \{C \mid l \in C\}$.

Proposition 4. *Let $G = \{G_0, G_1, \ldots, G_k\}$ be a group MUS instance. For any GMUS $M' \subseteq \mathsf{PLE}(G)$ of the group MUS instance $\mathsf{PLE}(G) = \{G'_0, G'_1, \ldots, G'_k\}$ obtained from applying PLE on G, $M = \{G_i \in G \mid G'_i \in M'\}$ is a GMUS of G.*

Proof. By Proposition 2, since PLE is clearly monotonic. □

Notice that any monotonic clause elimination procedure is also confluent (i.e., has a unique fixpoint). However, the opposite does not hold: a counterexample is subsumption elimination, which is confluent but not monotonic (recall Example 1).

3.2 Resolution-Based Preprocessing

Unit propagation

Plain MUS Extraction For the following, given a CNF formula F, let $F' = \mathsf{BCP}(F, l)$ where $(l) \in F$. For a clause C in F', the BCP *support* $\mathsf{support}_{\mathsf{BCP}}(C, F)$ of C in F is $\{C\}$ if $C \in F$, and $\{(l), (\bar{l} \vee C)\}$ (the premises that produced C) otherwise. A natural idea for reconstructing an MUS M of F from an MUS M' of F' would be to let $M = \bigcup_{C \in M'} \mathsf{support}_{\mathsf{BCP}}(C, F)$. Indeed, in the context of plain MUS extraction, this natural idea works, i.e., M is always guaranteed to be an MUS of F.

Proposition 5. *Let M' be an MUS of $F' = \mathsf{BCP}(F, l)$, where $(l) \in F$. Let $M = \bigcup_{C \in M'} \mathsf{support}_{\mathsf{BCP}}(C, F)$. Then $M \in \mathsf{MUS}(F)$.*

Proof. Assume w.l.o.g. that we have $F = \{(l), (l \vee C'_1), \ldots, (l \vee C'_n), (\bar{l} \vee C_1), \ldots, (\bar{l} \vee C_m)\} \cup R$, where R is the set of clauses in F which do not contain the variable of l. Hence the formula $F' = \mathsf{BCP}(F, l) = \{C_1, \ldots, C_m\} \cup R$. We have $M' \in \mathsf{MU}$, and want to show $M \in \mathsf{MU}$. Note that if $M' \subseteq R$, then $M = M'$, and we are done.

Otherwise, let $M' = \{C_{i_1}, \ldots, C_{i_k}\} \cup R'$, where $C_{i_j} \in \{C_1, \ldots, C_m\}$, $C_{i_j} \notin R'$, and $R' \subseteq R$. Then, we have $M = \{(l), (\bar{l} \vee C_{i_1}), \ldots, (\bar{l} \vee C_{i_k})\} \cup R'$. Clearly $\mathsf{BCP}(M, l) = M'$, and since $M' \in \mathsf{UNSAT}$, M must also be UNSAT (by the soundness of BCP). Let now C' be any clause in M, and let $\hat{M} = M \setminus \{C'\}$. If $C' \neq (l)$, then $\mathsf{BCP}(\hat{M}, l) \subset M'$, and since $M' \in \mathsf{MU}$, we have $\mathsf{BCP}(\hat{M}, l) \in \mathsf{SAT}$, and so, by the soundness of BCP, $\hat{M} \in \mathsf{SAT}$. If $C' = (l)$, then $\hat{M} = \{(\bar{l} \vee C_{i_1}), \ldots, (\bar{l} \vee C_{i_k})\} \cup R'$. But, since $M' \in \mathsf{MU}$, we have $R' \in \mathsf{SAT}$. Furthermore, the variable of l does not appear in R', and so setting l to 0 will satisfy the rest of the clauses in $\hat{M}$, and so $\hat{M} \in \mathsf{SAT}$.

We conclude that $M \in \mathsf{UNSAT}$, and, for any $C' \in M$, $M \setminus \{C'\} \in \mathsf{SAT}$. Hence, $M \in \mathsf{MU}$, and since $M \subset F$, we have $M \in \mathsf{MUS}(F)$. □

In other words, if a formula F_n is the result of an application of a sequence of BCP steps $F_2 = \mathsf{BCP}(F_1, l_1)$, …, $F_n = \mathsf{BCP}(F_{n-1}, l_{n-1})$, to a formula F_1, then given an MUS M_n of F_n, we can reconstruct an MUS M_1 of F_1 by taking the *transitive support* of the clauses in M_n, i.e., $M_{n-1} = \bigcup_{C \in M_n} \mathsf{support}_{\mathsf{BCP}}(C, F_{n-1})$, …, $M_1 = \bigcup_{C \in M_2} \mathsf{support}_{\mathsf{BCP}}(C, F_2)$. In particular, if $\emptyset \in F_n$, i.e. if the sequence of BCP steps results in a conflict, then the clauses of F_1 that were used to derive the conflict constitute an MUS of F_1. Thus Proposition 5 is a generalization of [17, Proposition 1] that states that inconsistent subformulas detected by unit propagation are minimally unsatisfiable.

Group MUS Extraction In the context of group MUS extraction, however, unit propagation *cannot* be safely applied over different groups by simply applying BCP on the monolithic CNF formula $F_G = G_0 \cup \cdots \cup G_n$, where $G = \{G_0, \ldots, G_n\}$ is the input group MUS instance. An intrinsic problem arises from the fact that $\mathsf{BCP}(F, l)$ can be seen as the combination of elimination of all clauses that are subsumed by (l) in F, and $\mathsf{VE}(F', l)$, where F' is the CNF formula resulting from the subsumption elimination step w.r.t. (l). More concretely, recall Example 1 which applies naturally to BCP as well. Another intrinsic problem in applying BCP steps using clauses from different groups is that the resolvents would inherit multiple group identities. Additionally, the sets of inherited group identities is dependent on the BCP variable ordering, as shown next.

Example 2. Consider the group MUS instance $G = \{G_0, \ldots, G_3\}$ with $G_0 = \{(x), (y)\}$, $G_1 = \{(z \vee p \vee q)\}$, $G_2 = \{(\bar{x} \vee \bar{z})\}$, and $G_3 = \{(\bar{y} \vee \bar{z}), (p \vee \bar{q}), (\bar{p} \vee q), (\bar{p} \vee \bar{q})\}$. Here $\{G_1, G_3\}$ is a group MUS of G. Assume now that a sequence of BCP steps is applied to G, viewed as a monolithic CNF formula $F_G = G_0 \cup \cdots \cup G_3$. There are two possible BCP sequences: both sequences produce an unsatisfiable CNF formula that contains all four binary clauses over p and q.

Now consider the possible supports of the clause $C = (p \vee q)$. If the first step is $\mathsf{BCP}(F_G, x)$, then the transitive support of C in F_G is $\{(x), (\bar{x} \vee \bar{z}), (z \vee p \vee q)\}$. In this case the derivation of C involves clauses from G_0, G_1, and G_2. If the first step is $\mathsf{BCP}(F_G, y)$, then the transitive support of C in F_g is $\{(y), (\bar{y} \vee \bar{z}), (z \vee p \vee q)\}$, involving clauses from G_0, G_1, G_3. Now, if we would associate C with all groups in its support, in the former case starting with $\mathsf{BCP}(F_G, x)$ (i.e., under a variable ordering preferring x to y) we end up with $\{G_1, G_2, G_3\} \supset \{G_1, G_3\}$. ■

A partial way of safely applying BCP on a group MUS instance $G = \{G_0, G_1, \ldots, G_k\}$ is to apply BCP fully on the special group G_0. In case unit propagation on G_0 alone leads to a conflict, then G has a single group MUS, namely the empty set. Otherwise, the derived unit clause can be propagated *individually* inside each group G_i, $1 \leq i \leq k$. The intuitive justification for this solution is that in the instance preprocessed with BCP, the transitive support of any clause $C \in G_i$ consists only of clauses of G_0 and a single G_i. By definition, the clauses in G_0 are always included in the unsatisfiability check for any selection of groups G_i, where $i > 0$, and furthermore, this way the group identities will not mix between the other groups.

Self-Subsuming Resolution

While the support for $\mathsf{BCP}(F, l)$ allows to reconstruct a plain MUS of F from an MUS of $\mathsf{BCP}(F, l)$, this technique fails for SSR under the following natural definition of support: given a CNF formula F, let $F' = \mathsf{SSR}(F, C, D, l)$. For a clause E in F', the SSR *support* $\mathsf{support}_{\mathsf{SSR}}(E, F)$ of E in F is $\{E\}$ if $E \in F$, and $\{C, D\}$ otherwise. As with the case of BCP support, this definition allows to recover the resolution step involved in the procedure. Consider the following example.

Example 3. Consider the CNF formula $F = \{(\bar{x} \vee p), (x \vee p \vee q), (\bar{p}), (x \vee \bar{q}), (\bar{x})\}$. After the application of self-subsuming resolution to the first two clauses of F we obtain the formula $F' = \{(\bar{x} \vee p), (p \vee q), (\bar{p}), (x \vee \bar{q}), (\bar{x})\}$. The only MUS of F' is $M' = \{(p \vee q), (\bar{p}), (x \vee \bar{q}), (\bar{x})\}$. Since $\mathsf{support}_{\mathsf{SSR}}((p \vee q), F) = \{(\bar{x} \vee p), (x \vee p \vee q)\}$, the union of the supports of all clauses in M' is precisely the formula F, which can easily be seen to *not* be in MU. ■

Variable Elimination

Since unit propagation is a special case of variable elimination, the problems discussed above with direct applications of BCP on the group MUS level apply to VE as well. However, similarly as for SSR, VE is problematic even in the context of *plain* MUS extraction. Intuitively, part of the problem is that the resolvents produced by a step $\mathsf{VE}(F, x)$ of variable elimination can have multiple pairs of supports, i.e., are produced via more than one distinct pair of premises (note that this is not the case for BCP). The problems caused by this behaviour are highlighted by the following example.

Example 4. Consider the CNF formula $F = A \cup R$, where
$A = \{(x \vee p \vee q \vee r), (x \vee \bar{q} \vee r), (\bar{x} \vee p \vee s), (\bar{s}), (\bar{x} \vee q)\}$ and
$R = \{(p \vee q \vee \bar{r}), (p \vee \bar{q} \vee \bar{r}), (\bar{p} \vee q \vee r), (\bar{p} \vee q \vee \bar{r}), (\bar{p} \vee \bar{q} \vee r), (\bar{p} \vee \bar{q} \vee \bar{r})\}$.
Notice that R is the set of all possible clauses on p, q, r, except $(p \vee q \vee r)$ and $(p \vee \bar{q} \vee r)$. Then, $F' = \mathsf{VE}(F, x) = A' \cup R$, where $A' = \{(p \vee q \vee r \vee s), (p \vee q \vee r), (p \vee \bar{q} \vee r \vee s), (\bar{s})\}$. F' has two MUSes: $M_1 = \{(p \vee q \vee r \vee s), (p \vee \bar{q} \vee r \vee s), (\bar{s})\} \cup R$ and $M_2 = \{(p \vee q \vee r), (p \vee \bar{q} \vee r \vee s), (\bar{s})\} \cup R$. Consider the idea of computing a *minimal support* of M_1 and M_2, i.e., the minimal set of premises $P \subseteq F$ such that $M_1 \subseteq \mathsf{VE}(P, x)$ and $M_2 \subseteq \mathsf{VE}(P, x)$, respectively, with the idea that such a minimal support would be an MUS of F. The minimal supports of M_1 and M_2 are $\{(x \vee p \vee q \vee r), (x \vee \bar{q} \vee r), (\bar{x} \vee p \vee s), (\bar{s})\} \cup R$ and $\{(x \vee p \vee q \vee r), (x \vee \bar{q} \vee r), (\bar{x} \vee p \vee s), (\bar{s}), (\bar{x} \vee q)\} \cup R = A \cup R$, respectively. The former is an MUS of F. However, the latter is not; in other words, even taking such a restricted, and "tightened-up", version of support for reconstructing an MUS is not generally correct. ■

For enabling direct applications of VE on group MUS instances, VE needs to be restricted. As in the case of BCP, VE can be applied solely on G_0, seen as a CNF formula, replacing the original G_0 with the resulting formula in the original instance. Furthermore, correctness is preserved if VE is applied *inside each group* G_i, $1 \leq i \leq k$, meaning that "internal" variables that occur only in clauses of a single group can be eliminated.

However, compared to such "ad hoc" technique-specific restrictions for applying preprocessing techniques in the context of group MUS extraction, a more generic framework for guaranteed correctness-preserving applications for different preprocessing techniques is called for. In the next two sections, we develop such a framework based on the concept of so-called *labelled CNF formulas* [2]. We then formally prove correctness of labelled variants of clause elimination and resolution-based preprocessing techniques for MUS extraction problems expressed in terms of labelled CNF formulas.

4 Labelled CNF Formulas

Assume a countable set of labels $Lbls$. A *labelled clause* (L-clause) is a tuple $\langle C, L \rangle$, where C is a clause, and L is a finite (possibly empty) subset of $Lbls$. We denote the label-sets by superscripts, i.e. C^L is the labelled clause $\langle C, L \rangle$. A *labelled CNF (LCNF)* formula is a finite set of labelled clauses. For an LCNF formula Φ, let $Cls(\Phi) = \bigcup_{C^L \in \Phi} \{C\}$ be the *clause-set* of Φ, and $Lbls(\Phi) = \bigcup_{C^L \in \Phi} L$ be the *label-set* of Φ. LCNF satisfiability is defined in terms of the satisfiability of the clause-sets of an LCNF formula: Φ is satisfiable if and only if $Cls(\Phi)$ is satisfiable. We will re-use the notation SAT (resp. UNSAT) for the set of satisfiable (resp. unsatisfiable) LCNF formulas[4]. However, the semantics of minimal unsatisfiability and MUSes of labelled CNFs are defined in terms of their label-sets via the concept of the *induced subformula*.

Definition 2 (Induced subformula). *Let Φ be an LCNF formula, and let $M \subseteq Lbls(\Phi)$. The subformula of Φ* induced by *M is the LCNF formula $\Phi|_M = \{C^L \in \Phi \mid L \subseteq M\}$.*

[4] To avoid overly optimistic complexity results, we will tacitly assume that the sizes of label-sets of the clauses in LCNFs are polynomial in the number of the clauses

In other words, $\Phi|_M$ consists of those labelled clauses of Φ whose label-sets are included in M, and so $Lbls(\Phi|_M) \subseteq M$, and $Cls(\Phi|_M) \subseteq Cls(\Phi)$. Alternatively, any clause that has at least one label outside of M is removed from Φ. Thus, it is convenient to talk about the *removal* of a label from Φ. Let $l \in Lbls(\Phi)$ be any label. The LCNF formula $\Phi|_{M\setminus\{l\}}$ is said to be obtained by the *removal of label l from Φ*.

Definition 3 (Minimally Unsatifiable LCNF). *An LCNF formula Φ is* minimally unsatisfiable *(denoted $\Phi \in \mathsf{LMU}$) if $\Phi \in \mathsf{UNSAT}$, and for all $M \subset Lbls(\Phi)$, $\Phi|_M \in \mathsf{SAT}$.*

Definition 4 (Labelled MUS). *Let Φ be an LCNF formula. A set of labels $M \subseteq Lbls(\Phi)$ is a* labelled MUS (LMUS) *of Φ ($M \in \mathsf{LMUS}(\Phi)$), if $\Phi|_M \in \mathsf{LMU}$.*

Note that LMUSes are sets of *labels*, rather than sets of clauses; this is motivated by the following example. Also, note that the empty set can be an LMUS of Φ (this is the case when the subset of clauses of $Cls(\Phi)$ labelled with $\emptyset$ is unsatisfiable), and for any LMUS M of Φ, $Cls(\Phi|_M)$ includes *all* clauses of Φ labelled with $\emptyset$. Finally, if $M \in \mathsf{LMUS}(\Phi)$, then $Lbls(\Phi|_M) = M$ (note the equality). We now illustrate how some of the notions of minimal unsatisfiability get represented in the framework of LCNFs.

Example 5. (a) Let $F = \{C_1, \ldots, C_n\}$ be a CNF formula, and let $\{i\}$ be the label-set of clause C_i. For any LMUS M of $\Phi = \{C_i^{\{i\}} \mid C_i \in F\}$, the CNF formula $\{C_i \mid i \in M\}$ is an MUS of F (and vice versa). (Notice that this is a reduction from MU to LMU.)
(b) Let $F = G_0 \cup G_1 \cup \ldots G_k$ be a group CNF formula. For each $C \in F$, take the label-set of C to be $\emptyset$ if $C \in G_0$, and $\{i\}$ if $C \in G_i$ for $i \geq 1$. For any LMUS M of the resulting LCNF Φ, $\{G_i \mid i \in M\}$ is a group MUS of F (and vice versa).
(c) For a CNF formula F and $C \in F$, let the set of variables of C be the label-set of C. Any LMUS M of the resulting LCNF is a *variable-MUS* [4] of F (and vice versa). ■

In the following, we refer to the LCNF formula constructed from a CNF formula F as in Example 5(a) as the LCNF *associated* with F; similarly, the LCNF formula constructed from the group CNF formula F as in Example 5(b) is referred to as the LCNF associated with the group CNF F. Notice that in Example 5(c) the label-sets of clauses are not necessarily disjoint. This allows to capture the semantics of "intersecting" groups, or, to put it differently, the multiple group identity of clauses (recall the discussion of BCP in the context of group MUS extraction in Section 3).

Computing LMUSes

It is not difficult to see that the LMUS extraction problem can be reduced to the group MUS extraction problem: given an LCNF formula Φ. For each label $l \in Lbls(\Phi)$, introduce a fresh variable p_l. For each L-clause $C^L \in \Phi$, create the clause $C \vee \bigvee_{l \in L} p_l$, and put the resulting clauses into the group G_0. Finally, for each $l \in Lbls(\Phi)$ create a singleton group $G_l = \{(\bar{p}_l)\}$. The resulting group-CNF formula $F_\Phi = \{G_0\} \cup \{G_l \mid l \in Lbls(\Phi)\}$ is equisatisfiable with Φ. Furthermore, $\{G_{l_1}, \ldots, G_{l_k}\}$ is a group-MUS of F_Φ if and only if $\{l_1, \ldots, l_k\}$ is an LMUS of Φ. We omit the proof, but the argument relies on the fact that a removal of a group G_l from F_Φ leaves the literal p_l pure in the clauses of G_0, thus satisfying all clauses with p_l. This in turn is equivalent to the removal of

all clauses $C^L \in \Phi$ with $l \in L$, i.e., the removal of the label l from Φ. Note that this reduction together with Example 5(a) can be used to show that the LMU decision problem is DP-complete.

Although the reduction from LMUS extraction to group MUS extraction enables the use of any group MUS extractor for the computation of LMUSes, we observe that in fact there is a simpler and likely more efficient way to compute LMUSes: one can simply load the clauses of the group G_0 of the formula F_Φ into an incremental SAT solver (such as Minisat), and use the variables p_l as *assumption variables*. Notice that the state-of-the-art assumption-based MUS extractors, such as MUSer2 [3] which is used in the experiments of this work, already do *exactly* this when computing MUSes and group-MUSes.

With this practical motivation, we will next provide liftings of the "problematic" preprocessing techniques (recall Section 3) to the labelled MUS setting. The liftings resolve the problems discussed in Section 3, and are safe to implement and apply using assumption variables.

5 Preprocessing in LMUS Extraction

We proceed by lifting clause elimination and resolution-based preprocessing techniques to the labelled case, resulting in correctness-preserving preprocessing techniques for labelled CNFs that are applicable in the general setting of group MUS extraction. It should be noted that labelled CNFs can be used to generalize all concepts related to minimal unsatisfiability and irredundancy (e.g. MSSes, MESes, MaxSAT, etc.) in various settings (clauses, groups, variables, circuits, etc.) [2]. As a by-product, given the natural mapping between plain and group MUS instances described in Example 5, this opens a path for correctness-preserving preprocessing for these settings as well.

5.1 Labelled Clause Elimination

While monotonic clause elimination procedures, including blocked clause elimination, can be directly applied in the group MUS context (recall Proposition 2), for other clause elimination procedures direct applicability appears to be limited. Especially, subsumption elimination cannot be directly applied (recall Example 1).

A correctness-preserving lifting of clause elimination procedures which preserve *logical equivalence* to the general setting of LMUS extraction is provided by the following proposition. Note that subsumption elimination is one of such procedures.

Proposition 6. *Let Φ be an LCNF formula such that for some clauses $C_1^{L_1}, \ldots, C_k^{L_k}$ and C^L in Φ, $\{C_1, \ldots, C_k\} \models C$ and $\bigcup_{1 \leq i \leq k} L_i \subseteq L$. Then, any LMUS of $\Phi \setminus \{C^L\}$ is an LMUS of Φ.*

Proof. Let $\Phi' = \Phi \setminus \{C^L\}$, and let M be an LMUS of Φ', i.e. $\Phi'|_M \in \mathsf{LMU}$. We need to show that $\Phi|_M \in \mathsf{LMU}$. Note that since $\Phi = \Phi' \cup \{C^L\}$ we have $\Phi|_M = \Phi'|_M \cup \{C^L\}|_M$. Thus, if $L \nsubseteq M$, then $\Phi|_M = \Phi'|_M$, and we are done since $\Phi'|_M \in \mathsf{LMU}$.

If $L \subseteq M$, then $C^L \in \Phi|_M$, and since $\bigcup_{1 \leq i \leq k} L_i \subseteq L$, *all* clauses $C_i^{L_i}$ are in $\Phi|_M$, and hence in $\Phi'|_M$. Consider any label $l \in M$, and let $M' = M \setminus \{l\}$. If $l \in L$,

then $C^L \notin \Phi|_{M'}$, and therefore $\Phi|_{M'} = \Phi'|_{M'} \in \mathsf{SAT}$, since $\Phi'|_M \in \mathsf{LMU}$. If $l \notin L$, then $\Phi|_{M'} = \Phi'|_{M'} \cup \{C^L\}$, and since $\bigcup L_i \subseteq L$, all clauses $C_i^{L_i}$ are in $\Phi'|'_M$. Since $\Phi'|_{M'} \in \mathsf{SAT}$, any of its satisfying assignments satisfies all clauses $C_i^{L_i}$, and also the clause C since $\{C_1, \ldots, C_k\} \models C$. Hence $\Phi|_{M'} \in \mathsf{SAT}$, and for any $l \in M$, $\Phi|_{M\setminus\{l\}} \in \mathsf{SAT}$, and so $\Phi|_M \in \mathsf{LMU}$. □

Applying Proposition 6 to subsumption elimination, we obtain that a clause $C_1^{L_1}$ subsumed by clause $C_2^{L_2}$ can be eliminated correctly (w.r.t. to LMUS computation) if $L_2 \subseteq L_1$. In particular, in the group-MUS setting, all clauses subsumed by the clauses from the same group, or by the clause from group G_0, can be eliminated safely.

5.2 Labelled Resolution-based Preprocessing

We now introduce liftings of the resolution-based preprocessing techniques to the context of LMUS extraction.

Definition 5 (L-resolvent). *The* L-resolvent *of two labelled clauses* $(x \vee A)^{L_1}$ *and* $(\bar{x} \vee B)^{L_2}$ *on variable* x *is the labelled clause* $(A \vee B)^{L_1 \cup L_2}$.

We will re-use the symbol $\otimes_x$ to denote the operation of L-resolution. As with the case of (plain) clauses, L-resolution rule is extended to sets of labelled clauses: for two such sets S_x and $S_{\bar{x}}$ of L-clauses which all contain the literal x and $\bar{x}$, resp., let $S_x \otimes_x S_{\bar{x}} = \{C_1^{L_1} \otimes_x C_2^{L_2} \mid C_1^{L_1} \in S_x, C_2^{L_2} \in S_{\bar{x}},$ and $C_1 \otimes_x C_2$ is not a tautology$\}$.

Labelled Variable Elimination Given an LCNF formula Φ, with subformulas $\Phi_x = \{C^L \in \Phi \mid x \in C\}$ and $\Phi_{\bar{x}} = \{C^L \in \Phi \mid \bar{x} \in C\}$, similarly to the case of (plain) CNF, we define the operation $\mathsf{LVE}(\Phi, x) = (\Phi \setminus (\Phi_x \cup \Phi_{\bar{x}})) \cup (\Phi_x \otimes_x \Phi_{\bar{x}})$. Notice that (as with VE) the definition implies that for any $C^L \in \mathsf{LVE}(\Phi, x)$, we have $x \notin C$, and either (i) $C^L \in \Phi$, or (ii) there exist $(x \vee C_1)^{L_1}$ and $(\bar{x} \vee C_2)^{L_2}$ in Φ such that $C^L = (C_1 \vee C_2)^{L_1 \cup L_2}$, or both (i) and (ii). It is not difficult to see that $Cls(\mathsf{LVE}(\Phi, x)) = \mathsf{VE}(Cls(\Phi), x)$, that is, the set of (plain) clauses underlying the LCNF Φ undergoes the same transformation as it would without labels, modulo the repeated clauses. Hence, as with the case of VE, LVE preserves satisfiability.

We will now show that the presence of labels attached to the clauses during the variable elimination allows to keep track of the relationship between the pre- and post-elimination formulas, and, as a result, allows to perform elimination correctly, that is, any LMUS of $\mathsf{LVE}(\Phi, x)$ is also an LMUS of Φ. As a first step, we show that the operations of LVE and $|_M$ commute.

Lemma 1. *For any LCNF* Φ*, variable* x*, and set of labels* M*,* $\mathsf{LVE}(\Phi, x)|_M = \mathsf{LVE}(\Phi|_M, x)$.

Proof. Take any $C^L \in \mathsf{LVE}(\Phi, x)|_M$. Note that $L \subseteq M$ and $x \notin C$. By the definition of LVE, we have that either (i) $C^L \in \Phi$, or (ii) for some $(x \vee C_1)^{L_1}$ and $(\bar{x} \vee C_2)^{L_2}$ in Φ, we have $C = C_1 \vee C_2$ and $L_1 \cup L_2 = L$, or both (i) and (ii). In the case (i), the clause C^L is in $\Phi|_M$, since $L \subseteq M$, and since $x \notin C$, $C^L \in \mathsf{LVE}(\Phi|_M, x)$. In the case (ii), both clauses $(x \vee C_1)^{L_1}$ and $(\bar{x} \vee C_2)^{L_2}$ are in $\Phi|_M$, since $L_1 \cup L_2 = L \subseteq M$, and by the definition of LVE, $C^L = (C_1 \vee C_2)^{L_1 \cup L_2} \in \mathsf{LVE}(\Phi|_M, x)$.

For the opposite direction, take $C^L \in \mathsf{LVE}(\Phi|_M, x)$. Note that $x \notin C$. By the definition of LVE, we have that either (i) $C^L \in \Phi|_M$, or (ii) for some $(x \vee C_1)^{L_1}$ and $(\bar{x} \vee C_2)^{L_2}$ in $\Phi|_M$, we have $C = C_1 \vee C_2$ and $L_1 \cup L_2 = L$, or both (i) and (ii). In the case (i), since $C^L \in \Phi$ and $x \notin C$, by the definition of LVE, we have $C^L \in \mathsf{LVE}(\Phi, x)$, and since $C^L \in \Phi|_M$ we have $L \subseteq M$, and so $C^L \in \mathsf{LVE}(\Phi, x)|_M$. In the case (ii), since $(x \vee C_1)^{L_1}$ and $(\bar{x} \vee C_2)^{L_2}$ are in Φ, by the definition of LVE we have $C^L = (C_1 \vee C_2)^{L_1 \cup L_2} \in \mathsf{LVE}(\Phi, x)$; since both clauses are in $\Phi|_M$, we have $L_1 \subseteq M$ and $L_2 \subseteq M$, Hence $L = L_1 \cup L_2 \subseteq M$, and $C^L \in \mathsf{LVE}(\Phi, x)$. □

Correctness of LVE with respect to LMUS extraction is established by applying Lemma 1.

Theorem 1. *For any LCNF formula Φ and variable x, any LMUS of $\mathsf{LVE}(\Phi, x)$ is an LMUS of Φ.*

Proof. Let M be an LMUS of $\mathsf{LVE}(\Phi, x)$, i.e. $\mathsf{LVE}(\Phi, x)|_M \in \mathsf{UNSAT}$, and for any $M' \subset M$, $\mathsf{LVE}(\Phi, x)|_{M'} \in \mathsf{SAT}$. By Lemma 1, we have $\mathsf{LVE}(\Phi, x)|_M = \mathsf{LVE}(\Phi|_M, x)$, and so $\mathsf{LVE}(\Phi|_M, x) \in \mathsf{UNSAT}$, and since LVE preserves satisfiability, $\Phi|_M \in \mathsf{UNSAT}$. Similarly, for the $M' \subset M$, by Lemma 1, we have $\mathsf{LVE}(\Phi, x)|_{M'} = \mathsf{LVE}(\Phi|_{M'}, x)$, and so $\mathsf{LVE}(\Phi|_{M'}, x) \in \mathsf{SAT}$, and so $\Phi|_{M'} \in \mathsf{SAT}$. Hence, $\Phi|_M \in \mathsf{UNSAT}$ and for any $M' \subset M$, $\Phi|_{M'} \in \mathsf{SAT}$, that is, M is an LMUS of Φ. □

Notice that the presence of labels addresses the problems with resolution-based preprocessing techniques in plain and group MUS settings outlined in Section 3. For example, labels provide a way to represent the multiple group identity of resolvents: a resolvent of two clauses from different groups simply inherits the identity of both groups. Furthermore, in the context of plain MUS extraction, if a clause C can be obtained by resolving two pairs of clauses C_1, C_2 and C_3, C_4, then in the LCNF setting, we will have *two* L-clauses C^{L_1} and C^{L_2} with $L_1 \neq L_2$. Although this might impede the effectiveness of VE, the correctness with respect to MUS computation is guaranteed. In fact, in Section 6, we demonstrate empirically that in the context of group MUS extraction, the technique is still effective.

Labelled Unit Propagation Notice that $\mathsf{BCP}(F, l)$ can be seen as the combination of (i) elimination of all clauses $(l \vee C_1), \ldots, (l \vee C_k)$ that are subsumed by (l) in F, and (ii) $\mathsf{VE}(F', l)$, where F' is the CNF formula resulting from the subsumption elimination step w.r.t. (l). Hence, combining Proposition 6 and Theorem 1, we can define *labelled unit propagation* $\mathsf{LBCP}(\Phi, (l)^L)$ for a given LCNF Φ and labelled unit clause $(l)^L \in \Phi$ as the combination of (1) labelled subsumption with the restriction that for each $(l \vee C_i)^L_i \in \Phi$ we have $L \subseteq L_i$ (following Proposition 6), and (2) $\mathsf{LVE}(\Phi', l)$, where Φ' is the LCNF resulting from step (1). Therefore, in the group-MUS setting, BCP can be applied *within* any of the groups, and by propagating any of the unit clauses derived from group G_0 to the groups G_i for $i > 0$.

Labelled Self-Subsuming Resolution Recall that a step of self-subsuming resolution $\mathsf{SSR}(F, C, D, l) = (F \backslash D) \cup \{C \otimes_l D\}$ can be seen as first adding the resolvent $(C \otimes_l D)$ to F, and then applying subsumption elimination to remove the the clause $D \supset C \otimes_l D$. Hence we define *labelled self-subsuming resolution* $\mathsf{LSSR}(\Phi, C^{L_C}, D^{L_D}, l)$ as the combination of (1) computing the L–resolvent of C^{L_C} and D^{L_D}, and (2) applying

labelled subsumption to remove D^{L_D} from the LCNF resulting from step (i), with the restriction that $L_C \cup L_D \subseteq L_D$, i.e., $L_C \subseteq L_D$. The correctness of LSSR is established in the following proposition.

Proposition 7. *Let Φ be any LCNF formula with two L-clauses C^{L_C} and D^{L_D} resolvable on the variable l and satisfying $L_C \subseteq L_D$. Then, any LMUS of the formula $\mathsf{LSSR}(\Phi, C^{L_C}, D^{L_D}, l)$ is an LMUS of Φ.*

Proof. Follows directly from the facts that the clauses $D^{L_D} \in \Phi$ and $C^{L_C} \otimes_l D^{L_D} \in \Phi'$ have the exact same set of labels L_D since $L_C \subseteq L_D$, and that $Cls(\Phi)$ and $Cls(\Phi')$ are logically equivalent. □

5.3 Applying the Labelled Preprocessing Techniques in Practice

The reduction from LMUS extraction to group MUS extraction and the subsequent discussion on the applicability of incremental SAT solvers to the LMUS computation problem (recall Section 4) suggest a simple way to implement most of the LCNF-based preprocessing techniques, namely LBCP, LVE, LSSR, and labelled subsumption elimination. As discussed before, given an LCNF formula Φ, add a fresh variable p_l for each $l \in Lbls(\Phi)$, and for every $C^L \in \Phi$ create a clause $(C \vee \bigvee_{l \in L} p_l)$. By F_Φ let us denote the resulting CNF formula (*not* the group-CNF discussed earlier). It is easy to see that the corresponding preprocessing techniques for plain CNF formulas can now be applied to F_Φ as long as VE is disallowed to eliminate the variables p_l. The resulting CNF formula F'_Φ is then mapped back into an LCNF formula Φ' by converting the variables p_l in the clauses into the label-sets of L-clauses to obtain the preprocessed version of Φ. The formula Φ' is then given to an LMUS computation algorithm. Based on the results presented in this section, the computed LMUS M of Φ' is an LMUS of Φ.

Connecting back to practical group MUS extraction, a simple way to apply the labelled preprocessing techniques within group MUS extraction is to exploit assumptions within an incremental SAT solver that incorporates the original non-labelled versions of the preprocessing techniques (recall the discussion on computing LMUSes in Section 4). We used this approach for the experiments described next.

6 Experimental results

The aim of the experimental study was to evaluate the potential effectiveness of various preprocessing techniques in the context of group MUS extraction. The focus on group MUSes is due to the high relevance of the problem to a number of formal verification applications (e.g. model checking and equivalence checking). To this end, we integrated some of the preprocessing techniques discussed in this paper into the group MUS extractor MUSer2 [3]. Specifically, we implemented BCE, which, as shown in Section 3, can be applied to group CNF instances safely prior to group MUS extraction by simply disregarding the group identities of the clauses. To implement additional preprocessing techniques, we took advantage of the fact that MUSer2 is an *assumption-based* MUS extractor, and followed the recipe outlined in the previous section: we configured it to work with the Minisat 2.2.0 (`http://minisat.se/`) SAT solver, and ran the SatElite

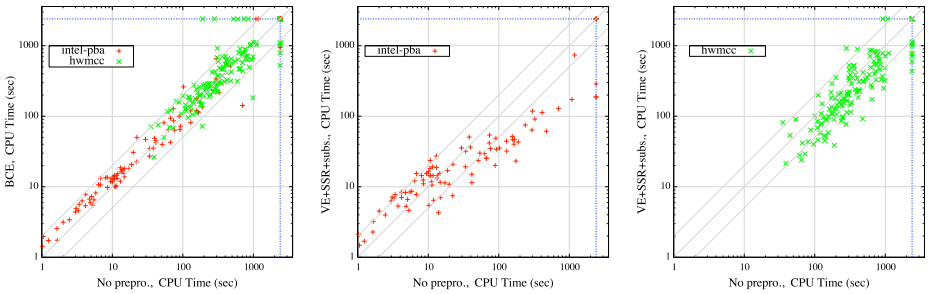

Fig. 6.1. Left: base vs. BCE. Center and right: base vs. VE+SSR+subsumption elimination (with SatElite), `intel-pba` center, `hwmcc` right. CPU time includes the time used for preprocessing.

preprocessor [6] of Minisat prior to group MUS extraction (note that SatElite allows to prohibit the elimination of particular variables). This corresponds to applying a combination of VE, SSR, and subsumption elimination prior to group MUS extraction

For the experiments, we used two sets of group MUS benchmarks. The first set, `intel-pba`, contains 99 instances submitted by Intel to the group MUS track of SAT Competition 2011. These instances originate from a proof-based abstraction framework. Their characteristic features are the size (reaching 4 million clauses), and the fact that over 90% of the clauses belong to group G_0. Each of the rest of the groups represents a gate (flop) over multiple timeframes in BMC unrolling. The second set, `hwmcc`, consists of 148 `belov` instances used in the same competition. These instances represent BMC unrolling of unsatisfiable instances from HWMCC 2010, whereby each AIG gate in each timeframe is represented a separate group (of 3 clauses). In these instances G_0 consists only of the unit clause that represent properly assertion. Note that hence the two sets differ drastically in structure, in a sense representing two extreme opposites in applications of group MUS extraction in proof-based abstraction.

The scatter plot on the left in Fig. 6.1, which demonstrates the effects of BCE on group MUS extraction time, suggests that BCE is not an effective technique for preprocessing group MUS instances. This is despite the fact that on most benchmarks BCE removes significant number of clauses (e.g. 2.5 million out of 3 on some of instances). On the other hand, as seen from the center and right plots in Fig. 6.1, the positive impact of resolution- and subsumption- based preprocessing on group MUS extraction time can be very significant, particularly on the difficult instances from the `intel-pba` set, where an order of magnitude speed-ups can be observed in some cases.

7 Conclusions

In this paper, we show that many CNF-level preprocessing techniques, routinely applied for speeding up SAT solving, are problematic in the context of plain MUS extraction, and, especially so, in the practically relevant context of group MUS extraction. To alleviate this problem, we developed sound liftings of the preprocessing techniques to the general context of labelled MUS extraction that captures group MUS extraction as well

as various other forms of MUS extraction problems. Our experimental results show that label-based preprocessing can improve the efficiency of group MUS extraction.

References

1. Bacchus, F.: Enhancing Davis Putnam with extended binary clause reasoning. In: Proc. AAAI. pp. 613–619. AAAI Press (2002)
2. Belov, A., Marques-Silva, J.: Generalizing redundancy in propositional logic: Foundations and hitting sets duality. Tech. rep., arXiv (2012), http://arxiv.org/abs/1207.1257
3. Belov, A., Marques-Silva, J.: MUSer2: An efficient MUS extractor. J. SAT 8, 123–128 (2012)
4. Belov, A., Ivrii, A., Matsliah, A., Marques-Silva, J.: On efficient computation of variable MUSes. In: Proc. SAT. LNCS, vol. 7317, pp. 298–311. Springer (2012)
5. Desrosiers, C., Galinier, P., Hertz, A., Paroz, S.: Using heuristics to find minimal unsatisfiable subformulas in satisfiability problems. J. Comb. Optim. 18(2), 124–150 (2009)
6. Eén, N., Biere, A.: Effective preprocessing in SAT through variable and clause elimination. In: Proc. SAT. LNCS, vol. 3569, pp. 61–75. Springer (2005)
7. Fourdrinoy, O., Grégoire, É., Mazure, B., Sais, L.: Eliminating redundant clauses in SAT instances. In: Proc. CPAIOR. LNCS, vol. 4510, pp. 71–83. Springer (2007)
8. Gershman, R., Strichman, O.: Cost-effective hyper-resolution for preprocessing CNF formulas. In: Proc. SAT. LNCS, vol. 3569, pp. 423–429. Springer (2005)
9. Grégoire, É., Mazure, B., Piette, C.: On approaches to explaining infeasibility of sets of Boolean clauses. In: Proc. ICTAI. pp. 74–83. IEEE (2008)
10. Han, H., Somenzi, F.: Alembic: An efficient algorithm for CNF preprocessing. In: Proc. DAC. pp. 582–587. IEEE (2007)
11. Heule, M., Järvisalo, M., Biere, A.: Efficient CNF simplification based on binary implication graphs. In: Proc. SAT. LNCS, vol. 6695, pp. 201–215. Springer (2011)
12. Heule, M., Järvisalo, M., Biere, A.: Clause elimination procedures for CNF formulas. In: Proc. LPAR-17. LNCS, vol. 6397, pp. 357–371. Springer (2010)
13. Heule, M., Järvisalo, M., Biere, A.: Covered clause elimination. In: LPAR short paper (2010)
14. Järvisalo, M., Biere, A., Heule, M.: Blocked clause elimination. In: Proc. TACAS. LNCS, vol. 6015, pp. 129–144. Springer (2010)
15. Järvisalo, M., Heule, M., Biere, A.: Inprocessing rules. In: Proc. IJCAR. LNCS, vol. 7364, pp. 355–370. Springer (2012)
16. Kullmann, O.: On a generalization of extended resolution. Discrete Applied Mathematics 96–97, 149–176 (1999)
17. Li, C., Manyà, F., Mohamedou, N., Planes, J.: Resolution-based lower bounds in MaxSAT. Constraints 15, 456–484 (2010)
18. Liffiton, M.H., Sakallah, K.A.: Algorithms for computing minimal unsatisfiable subsets of constraints. J. Autom. Reasoning 40(1), 1–33 (2008)
19. van Maaren, H., Wieringa, S.: Finding guaranteed MUSes fast. In: Proc. SAT. LNCS, vol. 4996, pp. 291–304. Springer (2008)
20. Marques-Silva, J.: Computing minimally unsatisfiable subformulas: State of the art and future directions. J. Mult-Valued Log. S. 19(1–3), 163–183 (2012)
21. Marques-Silva, J., Lynce, I.: On improving MUS extraction algorithms. In: Proc. SAT. LNCS, vol. 6695, pp. 159–173. Springer (2011)
22. Nadel, A.: Boosting minimal unsatisfiable core extraction. In: Proc. FMCAD. pp. 221–229. IEEE (2010)
23. Ryvchin, V., Strichman, O.: Faster extraction of high-level minimal unsatisfiable cores. In: Proc. SAT. LNCS, vol. 6695, pp. 174–187. Springer (2011)