Clause Elimination Procedures for CNF Formulas*

Marijn Heule[1], Matti Järvisalo[2], and Armin Biere[3]

[1] Department of Software Technology, Delft University of Technology, The Netherlands
[2] Department of Computer Science, University of Helsinki, Finland
[3] Institute for Formal Models and Verification, Johannes Kepler University Linz, Austria

Abstract. We develop and analyze clause elimination procedures, a specific family of simplification techniques for conjunctive normal form (CNF) formulas. Extending known procedures such as tautology, subsumption, and blocked clause elimination, we introduce novel elimination procedures based on *hidden* and *asymmetric* variants of these techniques. We analyze the resulting nine (including five new) clause elimination procedures from various perspectives: *size reduction*, *BCP-preservance*, *confluence*, and *logical equivalence*. For the variants not preserving logical equivalence, we show how to reconstruct solutions to original CNFs from satisfying assignments to simplified CNFs. We also identify a clause elimination procedure that does a transitive reduction of the binary implication graph underlying any CNF formula purely on the CNF level.

1 Introduction

Simplification techniques applied both before (i.e., in preprocessing) and during search have proven integral in enabling efficient conjunctive normal form (CNF) level Boolean satisfiability (SAT) solving for real-world application domains. Indeed, there is a large body of work on preprocessing CNF formulas (see [1–11] for examples), based on e.g. variable elimination and equivalence reasoning. Further, while many SAT solvers rely mainly on Boolean constraint propagation (i.e., unit propagation) during search, it is possible to improve solving efficiency by applying additional simplification techniques also during search, as witnessed e.g. by PrecoSAT (`http://fmv.jku.at/precosat`)—one of the most successful SAT solvers in the 2009 SAT Competition. Noticeably, when scheduling *combinations* of simplification techniques during search, even quite simply ideas, such as removal of subsumed clauses, can bring additional gains by enabling further simplifications by other techniques.

This work is motivated on one hand by the possibilities of lifting SAT solving efficiency further by integrating additional simplification techniques to the solving process before and/or during search, and on the other by understanding the relationships between different simplification techniques. In this paper, we concentrate on developing and analyzing clause elimination procedures, a specific family of simplification techniques for CNF formulas. Prior examples of such procedures are (explicit) tautology elimination (removing all tautologies from a CNF), subsumption elimination [7] (removing all subsumed clauses), and blocked clause elimination [11] (removing *blocked*

* The first author is supported by Dutch Organization for Scientific Research under grant 617.023.611, and the second author by Academy of Finland under grant #132812.

clauses [12]). As extensions of these procedures we introduce novel elimination procedures based on *hidden* and *asymmetric* variants of the techniques.

We analyze the resulting nine clause elimination procedures from various perspectives. One property is *effectiveness* (or *size reduction*), i.e., the ability to remove clauses and thus reduce the size of the CNF formula. Another orthogonal and practically relevant property is *BCP-preservance*, i.e, the ability to preserve all possible Boolean constraint propagations (i.e., unit propagations) that can also be done on the original CNF. The third property, *confluence*, implies that a procedure has a unique fixpoint. The fourth is *logical equivalence* w.r.t. the original CNF, i.e. preserving the set of satisfying assignments. For the variants that do not preserve logical equivalence, we show how to efficiently reconstruct solutions to original CNFs from satisfying assignments to simplified CNFs; this is important since in many application scenarios one needs to extract a satisfying assignment (witness) to the original SAT instances. Furthermore, we develop an extension of hidden tautology elimination that does a *transitive reduction* [13] (a structural property) of the binary implication graph underlying any CNF formula purely on the CNF level. We also evaluate the practical effectiveness of selected procedures, investigating both the CNF size reduction and resulting solving times.

This paper is organized as follows. After preliminaries (Sect. 2), we present an overview of the results on the properties of clause elimination procedures (Sect. 3). Then detailed analysis is presented (Sect. 4–6), followed by a section on solution reconstruction (Sect. 7). Then, before concluding, experimental results are presented (Sect. 8).

2 Preliminaries

CNF. 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 x. A truth assignment for a CNF formula F is a function τ that maps variables in F to $\{\mathbf{t}, \mathbf{f}\}$. If $\tau(x) = v$, then $\tau(\bar{x}) = \neg v$, where $\neg\mathbf{t} = \mathbf{f}$ and $\neg\mathbf{f} = \mathbf{t}$. A clause C is satisfied by τ if $\tau(l) = \mathbf{t}$ for some $l \in C$. An assignment τ satisfies F if it satisfies every clause in F. The set of literals occurring in a CNF formula F is denoted by $\mathsf{lits}(F)$. Formulas are *logically equivalent* if they have the same set of satisfying assignments over the common variables.

BCP and Failed Literals. For a CNF formula F, *Boolean constraint propagation* (BCP) (or *unit propagation*) propagates all unit clauses, i.e. repeats the following until fixpoint: if there is a unit clause $(l) \in F$, remove from $F \setminus \{(l)\}$ all clauses that contain the literal l, and remove the literal $\bar{l}$ from all clauses in F. The resulting formula is referred to as $\mathrm{BCP}(F)$. If $(l) \in \mathrm{BCP}(F)$ for some unit clause $(l) \notin F$, we say that BCP assigns the literal l to **t** (and the literal $\bar{l}$ to **f**). If $(l), (\bar{l}) \in \mathrm{BCP}(F)$ for some literal $l \notin F$ (or, equivalently, $\emptyset \in \mathrm{BCP}(F)$), we say that BCP derives a conflict.

For a partial assignment τ over the variables in F, let $\mathrm{BCP}(F, \tau) := \mathrm{BCP}(F \cup T_\tau \cup F_\tau)$, where $T_\tau = \{(x) \mid \tau(x) = \mathbf{t}\}$ and $F_\tau = \{(\bar{x}) \mid \tau(x) = \mathbf{f}\}$. It is easy to see that BCP has a unique fixpoint for any CNF formula, i.e., BCP is *confluent*.

A literal l is a *failed literal* if $\mathrm{BCP}(F \cup \{(l)\})$ contains the empty clause $\emptyset$, implying that F is logically equivalent to $\mathrm{BCP}(F \cup \{(\bar{l})\})$. For a formula F, *failed literal elimination* [1–3] (FLE) repeats the following until fixpoint: if there is a failed literal l in F, let $F := \mathrm{BCP}(F \cup \{(\bar{l})\})$. We denote the formula resulting from applying failed literal elimination on F by $\mathrm{FLE}(F)$. Since BCP is confluent, so is FLE, too.

Binary Implication Graphs and Equivalent Literal Substitution. Given a CNF formula F, we denote in the following by F_2 the set of binary clauses contained in F. For any F, one can associate with F_2 a unique directed *binary implication graph* (or simply $\mathsf{BIG}(F)$) with the node set $\mathsf{lits}(F_2)$ and edge relation $\{\langle \bar{l}, l' \rangle, \langle \bar{l}', l \rangle \mid (l \vee l') \in F_2\}$. In other words, for each binary clause $(l \vee l')$ in F, the two implications $\bar{l} \rightarrow l'$ and $\bar{l}' \rightarrow l$, represented by the binary clause, occur as edges in $\mathsf{BIG}(F)$. The strongly connected components (SCCs) of $\mathsf{BIG}(F)$ describe equivalent classes of literals (or simply equivalent literals) in F_2. *Equivalent literal substitution* (ELS) refers to substituting in F, for each SCC G of $\mathsf{BIG}(F)$, all occurrences of the literals occurring in G with the representative literal of G. Similar definitions occur in [8]. Notice that ELS is confluent modulo variable renaming.

3 Overview of Contributions

Before more detailed analysis, we now give an overview of the main results of this paper. We focus on nine different clause elimination procedures that are based on three variants (*explicit*, *hidden*, and *asymmetric*) of clause elimination techniques that remove *tautological*, *subsumed*, and *blocked* clauses. For (explicit) *tautology elimination* (TE), we have the variants *hidden tautology elimination* (HTE) and *asymmetric tautology elimination* (ATE). For (explicit) *subsumption elimination* (SE), we introduce the *hidden* and *asymmetric* variant HSE and ASE, respectively, and for (explicit) *blocked clause elimination* (BCE), the *hidden* and *asymmetric* variants HBCE and ABCE, resp.

A relevant aspect of simplification techniques is the question of how much a specific technique reduces the size of CNF formulas. In this paper we analyze the *relative effectiveness* of the considered clause elimination procedures based on the clauses removed by the procedures. For this we apply the following natural definition of effectiveness.

Definition 1. *Assume two clause elimination procedures S_1 and S_2 that take as input an arbitrary CNF formula F and each outputs a CNF formula that consists of a subset of F that is satisfiability-equivalent to F. Procedure S_1 is* at least as effective as *S_2 if, for any F and any output $S_1(F)$ and $S_2(F)$ of S_1 and S_2 on input F, respectively, we have that $S_1(F) \subseteq S_2(F)$; S_2 is* not as effective as *S_1 if there is an F for which there are outputs $S_1(F)$ and $S_2(F)$ of S_1 and S_2, respectively, such that $S_1(F) \subset S_2(F)$; and S_1 is* more effective than *S_2 if (i) S_1 is at least as effective as S_2, and (ii) S_2 is not as effective as S_1.*

Our definition of relative effectiveness takes into account *non-confluent* elimination procedures, i.e., procedures that do not generally have a unique fixpoint and that may thus have more than one possible output for a given input. The result of a non-confluent simplification procedure can be very unpredictable due to the non-uniqueness of results.

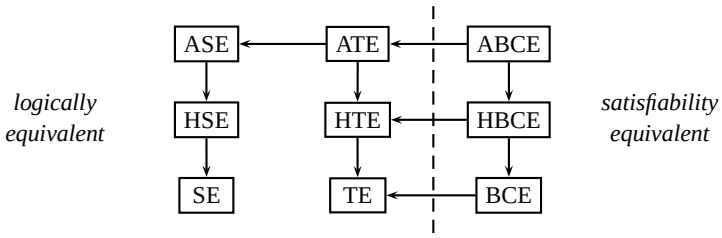

Fig. 1. Relative effectiveness hierarchy of clause elimination procedures. An edge from X to Y means that X is more effective than Y. A missing edge from X to Y means that X is not as effective as Y. However, notice that transitive edges are missing from the figure for clarity.

Our analysis on relative effectiveness results in an effectiveness hierarchy (Fig. 1) for the considered elimination procedures. For example, we show that for each of the known *explicit* techniques, the *hidden* and *asymmetric* variants are more effective, the latter of which being the most effective one of the three. In this sense, the novel variants are proper generalizations of the known explicit techniques. It also turns out that the most effective technique is the asymmetric variant of blocked clause elimination.

The further analysis presented in this paper considers the properties listed in Table 1. While each of the techniques preserves satisfiability (and are thus sound), it turns out that the variants of blocked clause elimination do not preserve logical equivalence; this is the motivation for demonstrating in Sect. 7 how one can efficiently reconstruct original solutions based on satisfying assignments for CNFs simplified using these variants. A further property of simplification techniques is BCP-preservance, which implies that relevant unit propagation (restricted to the remaining variables in the simplified CNF formula) possible in the original CNF is also possible in the simplified CNF under any partial assignment. This property is solver-related and very much practically relevant, since BCP is an integral part of a vast majority of SAT solvers today.

Definition 2. *For a formula F, a preprocessing procedure S* preserves BCP on F *if under any partial assignment τ over the variables in F and for any formula $S(F)$ resulting from applying S on F, we have that (i) for any literal l occurring in $S(F)$, $(l) \in \mathrm{BCP}(F,\tau)$ implies $(l) \in \mathrm{BCP}(S(F),\tau)$, and (ii) $\emptyset \in \mathrm{BCP}(F,\tau)$ implies $\emptyset \in \mathrm{BCP}(S(F),\tau)$ (the empty clause is obtained, i.e.,* BCP *derives a conflict). S is* BCP-preserving *if S preserves* BCP *on every CNF formula.*

Notice that our definition is similar to *deductive power* as defined in [10]. Also notice that BCP-preservance implies that logical equivalence is also preserved.

Interestingly, in turns out that BCP-preservance is quite a strict property, as only the basic SE and TE have it. However, by naturally combining HTE with a restricted version of FLE and ELS, we identify *extended hidden tautology elimination* (eHTE) which is *both* BCP-preserving *and* confluent (denoted in Table 1 with $*$), using conditions under which HTE does a *transitive reduction* [13] on the binary implication graphs underlying CNF formulas.

We proceed by giving detailed analysis of each of the variants of tautology, subsumption, and blocked clause based elimination procedures.

Table 1. Properties of clause elimination procedures

	SE	HSE	ASE	TE	HTE	ATE	BCE	HBCE	ABCE
satisfiability-equivalent	yes	yes	yes	yes	yes	yes	yes	yes	yes
logically equivalent	yes	yes	yes	yes	yes	yes	no	no	no
BCP-preserving	yes	no	no	yes	no / yes*	no	no	no	no
confluent	yes	no	no	yes	no / yes*	no	yes	no	no

4 Tautology-Based Clause Elimination Procedures

We begin by considering tautology elimination, introducing its hidden and asymmetric variants, and analyzing these procedures in more detail. For a given formula F, *tautology elimination* (TE) repeats the following until fixpoint: if there is a tautological clause $C \in F$, let $F := F \setminus \{C\}$. We refer to the reduced formula after applying tautology elimination on F as $\mathrm{TE}(F)$. It is easy to see that TE is confluent and BCP-preserving, and also that for any CNF formula F, $\mathrm{TE}(F)$ is logically equivalent to F.

4.1 Hidden Tautology Elimination

For a given clause C and a CNF formula F, we denote by (*hidden literal addition*) $\mathrm{HLA}(F, C)$ the *unique* clause resulting from repeating the following clause extension steps until fixpoint: if there is a literal $l_0 \in C$ such that there is a clause $(l_0 \vee l) \in F_2 \setminus \{C\}$ for some literal l, let $C := C \cup \{\bar{l}\}$. Notice that $\mathrm{HLA}(F, C) = \mathrm{HLA}(F_2, C)$. Furthermore, notice that for any $l \in \mathrm{HLA}(F, C) \setminus C$, there is for some $l_0 \in C$ a chain of binary clauses $(l_0 \vee \bar{l}_1), (l_1 \vee \bar{l}_2), \ldots, (l_{k-1} \vee \bar{l}_k)$ with $l = l_k$, equivalent to the implication chains $\bar{l}_0 \to \bar{l}_1, \bar{l}_1 \to \bar{l}_2, \ldots, \bar{l}_{k-1} \to \bar{l}_k$ and $l_k \to l_{k-1}, l_{k-1} \to l_{k-2}, \ldots, l_1 \to l_0$, in F_2 (equivalently, paths in $\mathsf{BIG}(F)$).

Lemma 1. *For any CNF formula F and clause $C \in F$, $(F \setminus \{C\}) \cup \{\mathrm{HLA}(F, C)\}$ is logically equivalent to F.*

Proof. For any literal $l \in \mathrm{HLA}(F, C) \setminus C$, by the definition of $\mathrm{HLA}(F, C)$, there is a $i \geq 0$ such that $l \to l_i, \ldots, l_1 \to l_0$ with $l_0 \in C$. Hence $(l_0) \in \mathrm{BCP}((F \setminus \{C\}) \cup \{(l)\})$, which implies that for any satisfying assignment τ for $(F \setminus \{C\})$ and $\mathrm{HLA}(F, C)$, if $\tau(l) = \mathbf{t}$ then $\tau(l_0) = \mathbf{t}$. Thus τ satisfies C and therefore also F. □

Alternatively, observe that each extension step in computing HLA is an application of self-subsuming resolution [7] in reverse order.

For a given CNF formula F, a clause $C \in F$ is a *hidden tautology* if and only if $\mathrm{HLA}(F, C)$ is a tautology. *Hidden tautology elimination* repeats the following until fixpoint: if there is a clause C such that $\mathrm{HLA}(F, C)$ is a tautology, let $F := F \setminus \{C\}$. A formula resulting from this procedure is denoted by $\mathrm{HTE}(F)$.

Lemma 2. HTE *is more effective than* TE.

Proof. HTE is at least as effective as TE due to $C \subseteq \mathrm{HLA}(F, C)$: if C is a tautology, so is $\mathrm{HLA}(F, C)$. Moreover, let $F = (a \vee b) \wedge (\bar{b} \vee c) \wedge (a \vee c)$. Since $\mathrm{HLA}(F, (a \vee c)) = (a \vee \bar{a} \vee b \vee \bar{b} \vee c \vee \bar{c})$, HTE can remove $(a \vee c)$ from F, in contrast to TE. □

Proposition 1. HTE *is not confluent.*

Proof. Consider the formula $F = (\bar{a} \vee b) \wedge (\bar{a} \vee c) \wedge (a \vee \bar{c}) \wedge (\bar{b} \vee c) \wedge (b \vee \bar{c})$. Now, $\mathrm{HLA}(F, (\bar{a} \vee b)) = \mathrm{HLA}(F, (\bar{a} \vee c)) = \mathrm{HLA}(F, (b \vee \bar{c})) = (a \vee \bar{a} \vee b \vee \bar{b} \vee c \vee \bar{c})$. HTE can remove either $(\bar{a} \vee b)$ or both $(\bar{a} \vee c), (b \vee \bar{c})$. □

Proposition 2. *For any CNF formula F, any* $\mathrm{HTE}(F)$ *is logically equivalent to F.*

Proof. Follows from the fact that TE preserves logical equivalence and Lemma 1. □

Proposition 3. HTE *is not* BCP*-preserving.*

Proof. Consider the formula $F = (a \vee b) \wedge (b \vee c) \wedge (b \vee \bar{c})$. HTE can remove clause $(a \vee b)$. Consider the assignment τ which assigns $\tau(a) = \mathbf{f}$. We have $(b) \in \mathrm{BCP}(F, \tau)$. However, $(b) \notin \mathrm{BCP}(F \setminus \{(a \vee b)\}, \tau)$. □

Although HTE is not confluent and does not preserve BCP in general, we identify eHTE, a natural variant of HTE which is both BCP-preserving and confluent.

For some intuition, consider again the formula $F = (a \vee b) \wedge (b \vee c) \wedge (b \vee \bar{c})$. Notice that $\bar{b} \in \mathrm{HLA}(F, (b)) = (\bar{a} \vee b \vee \bar{b} \vee c \vee \bar{c})$. Recall that HTE can only remove $(a \vee b)$ from F. However, since $\bar{b} \in \mathrm{HLA}(F, (b))$, $\bar{b}$ is a failed literal. Consequently, we can remove (*all*) clauses containing the literal b from F and add a unit clause (b). In general, we have the following.

Lemma 3. *Given a CNF formula F, for any literal l it holds that l is a failed literal in F_2 if and only if* $\bar{l} \in \mathrm{HLA}(F_2, (l))$.

Proof. There is a path from l to $\bar{l}$ in $\mathsf{BIG}(F)$ if and only if $\bar{l} \in \mathrm{HLA}(F_2, (l))$. □

Based on this observation, given a CNF formula F, binary-clause restricted failed literal elimination FLE_2 repeats the following until fixpoint: if there is a literal $l \in \mathsf{lits}(F_2)$ with $\bar{l} \in \mathrm{HLA}(F_2, (l))$, let $F := \mathrm{BCP}(F \cup \{(l)\})$. Since FLE is confluent, so is FLE_2. Refer to [8] for algorithmic aspects in computing FLE_2.

It turns out that for any CNF formula it holds that after applying FLE_2, HTE does the equivalent of a *transitive reduction*[4] of the binary implication graph $\mathsf{BIG}(\mathrm{FLE}_2(F))$.

Lemma 4. *Given a CNF formula F, let $F' := \mathrm{FLE}_2(F)$. Let F'_{HTE} stand for any formula resulting from applying* HTE *on F'. It then holds that $\mathsf{BIG}(F'_{\mathrm{HTE}})$ is a transitive reduction of $\mathsf{BIG}(F')$.*

Proof. Since $\mathsf{BIG}(F')$ is only influenced by F'_2, we focus on binary clauses removed from F' by HTE. For such a binary clause $C = (l \vee l')$, there are the edges $\bar{l} \to l'$ and $\bar{l}' \to l$ in $\mathsf{BIG}(F')$. Since neither l nor l' is a failed literal in F', there are also two paths $\bar{l} \to \ldots \to c$ and $\bar{l}' \to \ldots \to \bar{c}$ in $\mathsf{BIG}(F' \setminus C)$ such that $c, \bar{c} \in \mathrm{HLA}(F', C)$. Hence there are also the paths $\bar{l} \to \ldots \to c \to \ldots \to l'$ and $\bar{l}' \to \ldots \to \bar{c} \to \ldots \to l$, and hence both $\bar{l} \to l'$ and $\bar{l}' \to l$ are transitive edges in $\mathsf{BIG}(F')$. This shows that HTE only removes transitive edges of $\mathsf{BIG}(F')$. Applying HTE until fixpoint, all such transitive edges are removed from $\mathsf{BIG}(F')$, since any such $C = (l \vee l')$, such that there are the paths $\bar{l} \to \ldots \to c \to \ldots \to l'$ and $\bar{l}' \to \ldots \to \bar{c} \to \ldots \to l$, is a hidden tautology. □

[4] A directed graph G' is a transitive reduction [13] of the directed graph G provided that (i) G' has a directed path from node u to node v if and only if G has a directed path from node u to node v, and (ii) there is no graph with fewer edges than G' satisfying condition (i).

Notice that for every formula F such that $\mathsf{BIG}(F)$ is acyclic, it holds that $\mathsf{BIG}(F)$ has a unique transitive reduction, since the transitive reduction of any directed acyclic graph is unique [13]. In this case, there are no non-trivial SCCs in $\mathsf{BIG}(F)$. Furthermore, even for directed graph with cycles, the transitive reduction is unique modulo node equivalence classes [13]. This implies that applying the *combination* of $\mathrm{FLE}_2(F)$ and ELS before HTE, i.e., additionally substituting equivalent literals with the representatives of the literal equivalence classes (non-trivial strongly connected components) in $\mathsf{BIG}(\mathrm{FLE}_2(F))$, a unique transitive reduction (module variable renaming) is obtained.

With this intuition, for a formula F, extended hidden tautology elimination (eHTE) does the following two steps:

1. Repeat until fixpoint: (1a) Let $F := \mathrm{FLE}_2(F)$. (1b) Let $F := \mathrm{ELS}(F)$.
2. Apply HTE on F.

By the discussion above, eHTE is confluent.

Theorem 1. *eHTE is confluent.*

Furthermore, it turns out that by applying HTE on $\mathrm{FLE}_2(F)$, BCP is preserved in general; that is, even without applying equivalent literal substitution (Step 1b), we have a BCP-preserving variant of HTE.

Lemma 5. *For any CNF formula F,* HTE *preserves* BCP *on* $\mathrm{FLE}_2(F)$ *w.r.t.* F.

Proof. Consider an arbitrary CNF formula F, and let $F := \mathrm{FLE}_2(F)$. Assume that HTE removes a clause $C = (l_1 \vee \cdots \vee l_k) \in F$ from F; hence C is a hidden tautology in F, i.e., $\mathrm{HLA}(F, C)$ is a tautology.

Due to first applying FLE_2, C can not be a unit clause (l_1): otherwise, (l_1) would be a failed literal in F. The only way for BCP on all clauses of F to use C is that we have an assignment τ with $\tau(l_1) = \cdots = \tau(l_{k-1}) = \mathbf{f}$, in which case BCP on F can derive the unit clause (l_k), i.e., assign l_k to **t**; hence the case that C is a tautology is trivial. If C is a binary clause $(l_1 \vee l_2)$, then by Lemma 4 the implications representing C are transitive edges in $\mathsf{BIG}(F \setminus \{C\})$, and hence there are alternative implication chains between l_1^{i+1} and l_2^{i+1} in F which preserve BCP over C.

Now assume that C contains at least three literals and $\mathrm{HLA}(F, C)$ contains the opposite literals l and $\bar{l}$. Due to FLE_2, by assigning only a single l_i for some $i \in \{1, \ldots, k-1\}$ to **f**, BCP on binary clauses F_2 only, can not derive a conflict, and hence can not derive the unit clauses (l) and $(\bar{l})$. Otherwise $\bar{l}_i$ would be a failed literal. Therefore there are two distinct literals $l', l'' \in C$, based on which $\bar{l}$ and l are included in $\mathrm{HLA}(F, C)$, and $\mathsf{BIG}(F)$ contains two implication chains $\bar{l}' \to \bar{l}'_1, \bar{l}'_1 \to \bar{l}'_2, \ldots, \bar{l}'_{k'} \to l$ and $\bar{l}'' \to \bar{l}''_1, \bar{l}''_1 \to \bar{l}''_2, \ldots, \bar{l}''_{k''} \to \bar{l}$. Now there are two cases:

1. $l', l'' \in C \setminus \{l_k\}$. Since $\tau(l') = \tau(l'') = \mathbf{f}$, it follows that $(l), (\bar{l}) \in \mathrm{BCP}(F \setminus \{C\}, \tau)$, i.e., BCP derives a conflict without using C.
2. $l' \in C \setminus \{l_k\}$ and $l'' = l_k$. Then $\tau(l') = \mathbf{f}$, and it follows that $(l) \in \mathrm{BCP}(F \setminus \{C\}, \tau)$. Hence l is assigned to **t** by BCP under τ. Furthermore, since $l'' = l_k$ and the implication chain $\bar{l}_k \to \bar{l}''_1, \bar{l}''_1 \to \bar{l}''_2, \ldots, \bar{l}''_{k''} \to \bar{l}$ can be seen in the reversed order as $l \to l''_{k''}, l''_{k''} \to l''_{k''-1}, \ldots, l''_1 \to l_k$, after assigning l to **t** it follows that $(l_k) \in \mathrm{BCP}(F \setminus \{C\}, \tau)$. Hence BCP assigns l_k to **t** without using C. □

Furthermore, since ELS only does variable renaming by substituting equivalent literals, it can not interfere with BCP, and we have the following.

Theorem 2. *eHTE is* BCP*-preserving.*

Moreover, the following lemma follows the intuition on failed literals in HLA.

Lemma 6. *eHTE is more effective than* HTE*.*

In fact, here Step 1b of eHTE can again be omitted without affecting this result.

4.2 Asymmetric Tautology Elimination

For a clause C and a CNF formula F, (*asymmetric literal addition*) $\mathrm{ALA}(F, C)$ denotes the *unique* clause resulting from repeating the following until fixpoint: if $l_1, \ldots, l_k \in C$ and there is a clause $(l_1 \vee \ldots \vee l_k \vee l) \in F \setminus \{C\}$ for some literal l, let $C := C \cup \{\bar{l}\}$. A clause C is called an *asymmetric tautology* if and only if $\mathrm{ALA}(F, C)$ is a tautology.

Given a formula F, *asymmetric tautology elimination* (ATE) repeats the following until fixpoint: if there is an asymmetric tautological clause $C \in F$, let $F := F \setminus \{C\}$.

Lemma 7. *$\mathrm{ALA}(F, C)$ is a tautology if and only if* BCP *on $(F \setminus \{C\}) \cup \bigcup_{l \in C} \{(\bar{l})\}$ derives a conflict.*

As can be seen from Lemma 7, ATE performs what could be called *asymmetric branching* on clauses, which is used, e.g., in the technique of *clause distillation* [9].

The example in the proof of Proposition 1 implies the following.

Proposition 4. ATE *is not confluent.*

Proposition 5. *For any CNF formula F, $\mathrm{ATE}(F)$ is logically equivalent to F.*

Proof. For any clause C removed by ATE, $(F \setminus \{C\}) \cup \bigcup_{l \in C} \{(\bar{l})\}$ is unsatisfiable. This implies that $F \setminus \{C\} \models C$, i.e., $F \setminus \{C\}$ logically entails C. □

Proposition 6. ATE *is not* BCP*-preserving.*

Proof. Consider the following translation of x = If-Then-Else(c, t, e) into CNF:

$$(\bar{x} \vee \bar{c} \vee t) \wedge (x \vee \bar{c} \vee \bar{t}) \wedge (\bar{x} \vee c \vee e) \wedge (x \vee c \vee \bar{e}) \wedge (x \vee \bar{e} \vee \bar{t}) \wedge (\bar{x} \vee e \vee t)$$

Notice that ATE can remove $(x \vee \bar{e} \vee \bar{t})$ and $(\bar{x} \vee e \vee t)$. However, after removal, for truth assignment $\tau(e) = \tau(t) = \mathbf{f}$, BCP will no longer assign x to **t**. Also, for truth assignment $\tau(e) = \tau(t) = \mathbf{t}$, BCP will no longer assign x to **f**. □

The fact that $\mathrm{HLA}(F, C) = \mathrm{ALA}(F_2, C)$ implies the following.

Lemma 8. *For any CNF formula F and clause $C \in F$, $\mathrm{HLA}(F, C) \subseteq \mathrm{ALA}(F, C)$.*

Lemma 9. ATE *is more effective than* HTE*.*

Proof. ATE is at least as effective as HTE due to $\mathrm{HLA}(F, C) \subseteq \mathrm{ALA}(F, C)$: if $\mathrm{HLA}(F, C)$ is a tautology, then $\mathrm{ALA}(F, C)$ is a tautology. Moreover, consider the formula $F = (a \vee b \vee c) \wedge (a \vee b \vee d) \wedge (a \vee c \vee \bar{d})$. ATE will remove $(a \vee b \vee c)$ from F, while HTE removes none of the clauses. □

5 Subsumption-Based Clause Elimination Procedures

We now turn to the explicit, hidden, and asymmetric variants of the procedures that eliminate subsumed clauses. Given a CNF formula F, a clause $C_1 \in F$ *subsumes* (another) clause $C_2 \in F$ in F if and only if $C_1 \subset C_2$, and then C_2 is *subsumed by* C_1. Any assignment that satisfies C_1 will also satisfy C_2. For a given formula F, *subsumption elimination* (SE) repeats the following until fixpoint: if there is a subsumed clause $C \in F$, let $F := F \setminus \{C\}$. We refer to the reduced formula after applying subsumption elimination on F as $\mathrm{SE}(F)$. It is easy to see that SE is confluent and BCP-preserving, and that for any CNF formula F, $\mathrm{SE}(F)$ is logically equivalent to F.

5.1 Hidden Subsumption Elimination

For a given formula F, *hidden subsumption elimination* (HSE) repeats the following until fixpoint: if there is a clause $C \in F$ for which $\mathrm{HLA}(F, C)$ is subsumed in F, let $F := F \setminus \{C\}$.

By replacing HTE with HSE in the proof of Proposition 1 we have the following.

Proposition 7. HSE *is not confluent.*

Lemma 10. *For any CNF formula* F, $\mathrm{HSE}(F)$ *is logically equivalent to* F.

Proof. Follows from Lemma 1 and the fact that SE preserves logical equivalence. □

Proposition 8. HSE *is not* BCP*-preserving.*

Proof. Let $F = (a \vee b \vee c) \wedge (a \vee b \vee d) \wedge (b \vee \bar{c})$. HSE can remove $(a \vee b \vee d)$, because $\mathrm{HLA}(F, (a \vee b \vee d)) = (a \vee b \vee c \vee d)$ is subsumed by $(a \vee b \vee c)$. Consider the assignment τ which assigns $\tau(a) = \tau(d) = \mathbf{f}$. We have $(b) \in \mathrm{BCP}(F, \tau)$. However, $(b) \notin \mathrm{BCP}(F \setminus \{(a \vee b \vee d)\}, \tau)$. □

Notice that the above proof also holds after F is simplified by FLE.

Lemma 11. HSE *is more effective than* SE.

Proof. HSE is at least as effective as SE since for any CNF formula F, (i) for every clause $C \in F$, $C \subseteq \mathrm{HLA}(F, C)$, and (ii) if C is subsumed then any clause $C' \supseteq C$ is subsumed. Moreover, let $F = (a \vee b \vee c) \wedge (a \vee b \vee d) \wedge (b \vee \bar{c}) \wedge (\bar{a} \vee d \vee \bar{d})$. HSE can remove $(a \vee b \vee d)$ because $\mathrm{HLA}(F, (a \vee b \vee d)) = (a \vee b \vee c \vee d)$, in contrast to SE. □

Also notice that, given two identical clauses C_1 and C_2 (i.e., $C_1 \subseteq C_2$ and $C_2 \subseteq C_1$), HSE can remove either C_1 or C_2, while SE cannot.

Lemma 12. *It holds that (i)* HSE *is not as effective as* HTE*, and that (ii)* HTE *is not as effective as* HSE.

Proof. Consider the formula F_{HSE}. HTE can remove the tautology $(\bar{a} \vee d \vee \bar{d})$, but no other clauses. HSE can remove $(a \vee b \vee d)$, but no other clauses. □

5.2 Asymmetric Subsumption Elimination

For a given formula F, *asymmetric subsumption elimination* (ASE) repeats the following until fixpoint: if there is a clause $C \in F$ for which $\mathrm{ALA}(F, C)$ is subsumed in F, let $F := F \setminus \{C\}$.

By replacing ATE with ASE in the proof of Lemma 6 we have the following.

Proposition 9. ASE *is not* BCP*-preserving.*

Lemma 13. ASE *is more effective than* HSE*.*

Proof. ASE is at least as effective as HSE since (i) for every clause $C \in F$ we have $\mathrm{HLA}(F, C) \subseteq \mathrm{ALA}(F, C)$ (Lemma 8), and (ii) if C is subsumed then any clause $C' \supseteq C$ is subsumed. Moreover, consider the formula $F = (a \vee b \vee c) \wedge (a \vee b \vee d) \wedge (a \vee c \vee \bar{d})$. ASE will remove $(a \vee b \vee c)$ from F, while HSE removes no clauses from F. □

Lemma 14. ATE *is more effective than* ASE*.*

Proof. To see that ATE is at least as effective as ASE, consider the following. If there is a clause $C \in F$ for which $\mathrm{ALA}(F, C)$ is subsumed by $C' \in F \setminus \{C\}$, then $\mathrm{ALA}(F, C)$ is a tautology: say $\mathrm{ALA}(F, C)$ is subsumed by $C' = (l_1 \vee \ldots \vee l_k)$. Due to the update rule of ALA, $\bar{l}_1, \ldots, \bar{l}_k \in \mathrm{ALA}(F, C)$. Moreover, consider the formula $F = (a \vee \bar{a})$. ASE will not remove this tautology, in contrast to ATE. □

6 Clause Elimination Procedures based on Blocked Clauses

As the final family of clause elimination procedures considered in this paper, we now introduce and analyze procedures that eliminate blocked clauses [12].

The resolution rule states that, given two clauses $C_1 = \{l, a_1, \ldots, a_n\}$ and $C_2 = \{\bar{l}, b_2, \ldots, b_m\}$, the implied clause $C = \{a_1, \ldots, a_n, b_1, \ldots, b_m\}$, called the *resolvent* of C_1 and C_2, can be inferred by *resolving* on the literal l, and write $C = C_1 \otimes_l C_2$.

Given a CNF formula F, a clause C and a literal $l \in C$, the literal l *blocks* C w.r.t. F if (i) for each clause $C' \in F$ with $\bar{l} \in C'$, $C \otimes_l C'$ is a tautology, or (ii) $\bar{l} \in C$, i.e., C is itself a tautology [5]. Given a CNF formula F, a clause C is *blocked* w.r.t. F if there is a literal that blocks C w.r.t. F. Removal of blocked clauses preserves satisfiability [12].

For a CNF formula F, *blocked clause elimination* (BCE) repeats the following until fixpoint: if there is a blocked clause $C \in F$ w.r.t. F, let $F := F \setminus \{C\}$. The CNF formula resulting from applying BCE on F is denoted by $\mathrm{BCE}(F)$.

Proposition 10. *For some CNF formula* F, $\mathrm{BCE}(F)$ *is not logically equivalent to* F*.*

Proof. Consider the following CNF formula, having a structure that is often observed in CNF encodings of graph coloring problems.

$$F_{\mathrm{BCE}} = (a \vee b \vee c) \wedge (d \vee e \vee f) \wedge (\bar{a} \vee \bar{d}) \wedge (\bar{b} \vee \bar{e}) \wedge (\bar{c} \vee \bar{f}) \wedge (\bar{a} \vee \bar{b}) \wedge (\bar{a} \vee \bar{c}) \wedge (\bar{b} \vee \bar{c}) \wedge (\bar{d} \vee \bar{e}) \wedge (\bar{d} \vee \bar{f}) \wedge (\bar{e} \vee \bar{f}).$$

[5] Here $\bar{l} \in C$ is included in order to handle the special case that for any tautological *binary* clause $(l \vee \bar{l})$, both l and $\bar{l}$ block the clause. Notice that, even without this addition, every *non-binary* tautological clause contains at least one literal that blocks the clause.

BCE can remove the last six binary clauses (the second row) in F_{BCE}. Consider the truth assignment τ with $\tau(a) = \tau(b) = \tau(f) = \mathbf{t}$ and $\tau(c) = \tau(d) = \tau(e) = \mathbf{f}$. Although τ satisfies $\mathrm{BCE}(F_{\mathrm{BCE}})$, the clause $(\bar{a} \vee \bar{b})$ in F_{BCE} is falsified by τ. □

6.1 Hidden Blocked Clause Elimination

For a given CNF formula F, a clause $C \in F$ is called *hidden blocked* if $\mathrm{HLA}(F, C)$ is blocked w.r.t. F. *Hidden blocked clause elimination* (HBCE) repeats the following until fixpoint: if there is a hidden blocked clause $C \in F$, remove C from F.

Lemma 15. *Removal of an arbitrary hidden blocked clause preserves satisfiability.*

Proof. Follows from the facts that F is logically equivalent to $(F \setminus \{C\}) \cup \{\mathrm{HLA}(F, C)\}$ and that BCE preserves satisfiability. □

Proposition 11. HBCE *is not confluent.*

Proof. Let $F = (\bar{a} \vee b) \wedge (\bar{a} \vee c) \wedge (a \vee \bar{d}) \wedge (\bar{b} \vee d) \wedge (\bar{c} \vee d)$. F contains four hidden blocked clauses: $\mathrm{HLA}(F, (\bar{a} \vee b)) = (\bar{a} \vee b \vee \bar{c} \vee \bar{d})$ with blocking literal b, $\mathrm{HLA}(F, (\bar{a} \vee c)) = (\bar{a} \vee \bar{b} \vee c \vee \bar{d})$ with blocking literal c, $\mathrm{HLA}(F, (\bar{b} \vee d)) = (a \vee \bar{b} \vee c \vee d)$ with blocking literal $\bar{b}$, and $\mathrm{HLA}(F, (\bar{c} \vee d)) = (a \vee b \vee \bar{c} \vee d)$ with blocking literal $\bar{c}$. HBCE removes either $(\bar{a} \vee b)$ and $(\bar{b} \vee d)$, or $(\bar{a} \vee c)$ and $(\bar{c} \vee d)$. □

Replacing BCE with HBCE in the proof of Proposition 10, we have the following.

Proposition 12. *For some CNF formula F,* $\mathrm{HBCE}(F)$ *is not logically equivalent to F.*

Lemma 16. HBCE *is more effective than* BCE *and* HTE*.*

Proof. HBCE is at least as effective as BCE due to $C \subseteq \mathrm{HLA}(F, C)$ and that each blocking literal $l \in C$ is also blocking $\mathrm{HLA}(F, C)$. HBCE is at least as effective as HTE since tautologies are blocked clauses. Moreover, let $F = (a \vee c) \wedge (\bar{a} \vee d) \wedge (\bar{b} \vee c) \wedge (b \vee d) \wedge (\bar{c} \vee \bar{d})$. Now $\mathrm{HLA}(F, (a \vee c)) = (a \vee b \vee c \vee \bar{d})$ with blocking literal a, and $\mathrm{HLA}(F, (\bar{a} \vee d)) = (\bar{a} \vee \bar{b} \vee \bar{c} \vee d)$ with blocking literal $\bar{a}$. Hence HBCE removes both $(a \vee c)$ and $(\bar{a} \vee d)$, while neither BCE nor HTE can remove any clause of F. □

6.2 Asymmetric Blocked Clause Elimination

For a given CNF formula F, a clause $C \in F$ is called asymmetric blocked if $\mathrm{ALA}(F, C)$ is blocked w.r.t. F. *Asymmetric blocked clause elimination* (ABCE) repeats the following until fixpoint: if there is an asymmetric blocked clause $C \in F$, let $F := F \setminus \{C\}$.

Lemma 17. *Removal of an asymmetric blocked clause preserves satisfiability.*

Proof. Follows from the facts that F is logically equivalent to $(F \setminus \{C\}) \cup \{\mathrm{ALA}(F, C)\}$ and that BCE preserves satisfiability. □

Proposition 13. ABCE *is not confluent.*

Proof. Let $F = (\bar{a} \vee b) \wedge (\bar{a} \vee c) \wedge (a \vee \bar{d}) \wedge (\bar{b} \vee d) \wedge (\bar{c} \vee d)$. F contains four asymmetric blocked clauses: $\mathrm{ALA}(F, (\bar{a} \vee b)) = (\bar{a} \vee b \vee \bar{c} \vee \bar{d})$ with blocking literal b, $\mathrm{ALA}(F, (\bar{a} \vee c)) = (\bar{a} \vee \bar{b} \vee c \vee \bar{d})$ with blocking literal c, $\mathrm{ALA}(F, (\bar{b} \vee d)) = (a \vee \bar{b} \vee c \vee d)$ with blocking literal $\bar{b}$, and $\mathrm{ALA}(F, (\bar{c} \vee d)) = (a \vee b \vee \bar{c} \vee d)$ with blocking literal $\bar{c}$. ABCE removes either $(\bar{a} \vee b)$ and $(\bar{b} \vee d)$, or $(\bar{a} \vee c)$ or $(\bar{c} \vee d)$ from F. □

Replacing BCE with ABCE in the proof of Proposition 10, we have the following.

Proposition 14. *For some CNF formula F, $\mathrm{ABCE}(F)$ is not logically equivalent to F.*

Lemma 18. ABCE *is more effective than* HBCE *and* ATE*.*

Proof. ABCE is at least as effective as HBCE due to $\mathrm{HLA}(F, C) \subseteq \mathrm{ALA}(F, C)$ (recall Lemma 8): if $\mathrm{HLA}(F, C)$ is a tautology, then $\mathrm{ALA}(F, C)$ is a tautology. ABCE is at least as effective as ATE since tautologies are blocked clauses. Moreover, consider the formula $F_{\mathrm{ABCE}} = (\bar{a} \vee b \vee c) \wedge (b \vee c \vee \bar{d}) \wedge (a \vee\ d) \wedge (\bar{b} \vee \bar{d}) \wedge (\bar{c} \vee \bar{d})$. Now $\mathrm{ALA}(F_{\mathrm{ABCE}}, (\bar{a} \vee b \vee c)) = (\bar{a} \vee b \vee c \vee d)$ in which b and c are blocking literals. Hence ABCE can remove $(\bar{a} \vee b \vee c)$ (and in fact all clauses in F_{ABCE}). Neither HBCE nor ATE can remove any clause from F_{ABCE}. □

7 Reconstructing Solutions after HBCE and ABCE

Since the elimination procedures based on blocked clauses do not preserve logical equivalence, a truth assignment τ satisfying $\mathrm{BCE}(F)$ may not satisfy F. However, a satisfying assignment for F can be constructed based on τ as follows [14]. Add the clauses $C \in F \setminus \mathrm{BCE}(F)$ back in the opposite order of their elimination. In case C is satisfied by τ, do nothing. Otherwise, assuming that $l \in C$ is blocking C, flip the truth value of l in τ to **t**. After all clauses have been added, the modified τ satisfies F.

We now show that this procedure can be used to reconstruct solutions for formulas simplified using HBCE or ABCE. The lemmas will focus on ALA, but because HLA is a restricted version of ALA, all lemmas also hold when ALA is replaced by HLA.

Lemma 19. *Given a clause $C \in F$, if $\mathrm{ALA}(F, C)$ is blocked and not a tautology, then there is a literal $l \in C$ blocking it.*

Proof. By construction, for each literal $l \in \mathrm{ALA}(F, C) \setminus C$, here is a clause $C' \in F$ that contains $\bar{l}$ and $C' \setminus \{\bar{l}\} \subseteq \mathrm{ALA}(F, C)$. Therefore, because $\mathrm{ALA}(F, C)$ is not a tautology, $C' \otimes_l \mathrm{ALA}(F, C) = \mathrm{ALA}(F, C) \setminus \{l\}$ is not a tautology either. Hence l is not blocking $\mathrm{ALA}(F, C)$. □

Lemma 20. *Given a CNF formula F and a truth assignment τ satisfying F, if $C \notin F$ is falsified by τ, then $\mathrm{ALA}(F, C)$ is falsified by τ.*

Proof. From Lemma 7 follows that $F \cup \{\mathrm{ALA}(F, C)\}$ is logically equivalent to $F \cup \{C\}$. Therefore, $\mathrm{ALA}(F, C)$ is satisfied by τ if and only if τ satisfies C. □

Lemma 21. *Given a CNF formula F and a truth assignment τ satisfying F, if $C \notin F$ is falsified by τ and $\mathrm{ALA}(F, C)$ is blocked w.r.t. F with blocking literal $l \in C$, then τ satisfies at least two literals in each clause $C' \in F$ with $\bar{l} \in C'$.*

Proof. First, such $C' \in F$ contain a literal $\bar{l}$ which is satisfied by τ. Second, because l is blocking, each clause C' must contain one more literal $l' \neq \bar{l}$ such that $\bar{l}' \in \mathrm{ALA}(F, C)$. Since all literals in $\mathrm{ALA}(F, C)$ are falsified by τ, l' must be satisfied by τ. □

Combining these three lemmas, we can reconstruct a solution for F if we have a satisfying assignment τ for any $\mathrm{ABCE}(F)$ (and also any $\mathrm{HBCE}(F)$). The clauses $C \in F \setminus \mathrm{ABCE}(F)$ are added back in reverse order of elimination to ensure that $\mathrm{ALA}(F, C)$ is blocked. If C is satisfied by F do nothing. Otherwise, we know that there is a literal $l \in C$ blocking $\mathrm{ALA}(F, C)$; recall Lemma 19. Furthermore, all literals in $\mathrm{ALA}(F, C)$ are falsified; recall Lemma 20. However, any $C' \in F$ containing $\bar{l}$ has two satisfied literals; (recall Lemma 21. Therefore, by flipping the truth assignment for l to **t**, C becomes satisfied, while no such C' becomes falsified.

Theorem 3. *The following holds for an arbitrary CNF formula F and truth assignment τ satisfying F. For any clause $C \notin F$ for which C, $\mathrm{HLA}(F, C)$, or $\mathrm{ALA}(F, C)$ is blocked w.r.t. F with blocking literal l, either (i) τ satisfies $F \cup \{C\}$, or (ii) τ', which is a copy of τ except for $\tau'(l) = \mathbf{t}$, satisfies $F \cup \{C\}$.*

The reconstruction proof provides several useful elements that can be used to implement HBCE and ABCE more efficiently. First, since only original literals $l \in C$ can be blocking $\mathrm{HLA}(F, C)$ or $\mathrm{ALA}(F, C)$, we can avoid a blocking literal check for all literals $l \in \mathrm{HLA}(F, C) \setminus C$ or $l \in \mathrm{ALA}(F, C) \setminus C$. Second, it is enough to save each removed *original* clause C. None of the additional literals in the *extended* clause $\mathrm{HLA}(F, C)$ (or $\mathrm{ALA}(F, C)$, resp.) not occurring in C have to be flipped.

8 Experimental Evaluation

We shortly present initial experiments results on the effectiveness of selected clause elimination procedures, focusing on the current implementations of HTE and HBCE. The benchmarks set used consists of the 2009 SAT Competition application instances (292 in total), with each instance processed beforehand with BCP. A comparison of the effectiveness of BCE, HTE, and HBCE (all until fixpoint) is shown on the left in Fig. 2, illustrating the percentage of clauses remaining after applying the individual techniques (with original number of clauses 100%). Here data for each plot is sorted according to the reduction percentage, with the percentages of clauses remaining on the y-axis. We include BCE due to recent encouraging results presented in [11]. In line with our analysis (recall Fig. 1), HBCE is clearly the most effective technique. There is not that clear a winner between BCE and HTE, although HTE does prevail in the end.

The hidden clause elimination procedures are probably the most interesting novel techniques in practice, because they can be implemented efficiently. In particular, eHTE is expected to be useful, since it also preserves BCP. Since we have no efficient implementation of FLE_2 and ELS at this time, the experiments on practical use focus on HTE instead.

As can be seen on the right in Fig. 2 (time as a function of number of instances solved), HTE gives gains w.r.t. solution times for MiniSAT 2.0. Here we used the version of MiniSAT without the built-in preprocessor to see the effect of HTE on its own.

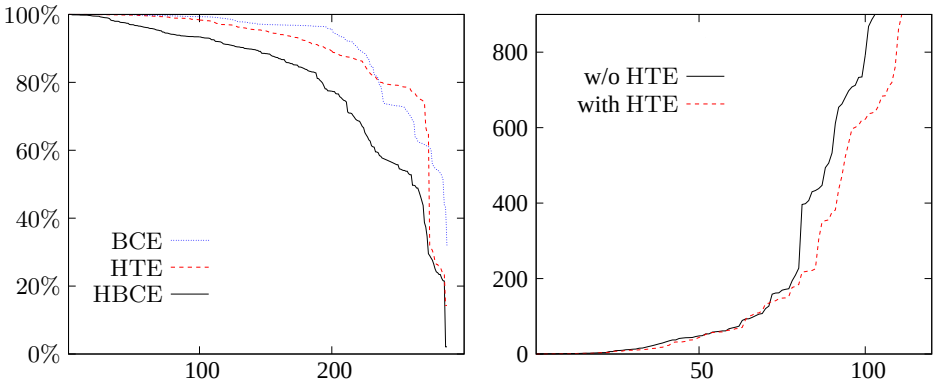

Fig. 2. Comparison of the effectiveness of various clause elimination procedures on the size of SAT 2009 benchmark instances (left). Also, the number of instances solved in less than t seconds by MiniSAT 2.0 without and with HTE as preprocessing step (right).

Notice that we also conducted an additional experiment in which we first preprocessed all instances using SatELite [7]; this resulted in similar performance gains.

For most benchmarks in the SAT 2009 application suit, the cost of applying HTE is less then a second. However, on instances in which $\mathsf{BIG}(F)$ contains large SCCs, the computational cost is on average 60 seconds. We expect that by combining HTE with ELS, as in eHTE, HTE will be quite efficient also for these instances.

Applying any of the asymmetric clause elimination procedures until fixpoint will hardly be useful in practice. The most important reason is that all these procedures are very costly. Also, because they do not preserve BCP, for several instances they can decrease performance even in case these costs are neglected. However, the asymmetric procedures will probably be of practical use when they are restricted. For instance, by only applying them on long clauses or for a short time (i.e., not until fixpoint).

Our implementation of HTE does not explicitly compute $\mathrm{HLA}(F, C)$ for each $C \in F$. Instead, for each literal $l \in \mathsf{lits}(F)$, we compute $\mathrm{HLA}(F, (l))$. Elimination of clauses is realized as follows: First, mark each literal l' for which $\bar{l}' \in \mathrm{HLA}(F, (l))$ with label l. Second, for all clauses C with $l \in C$ we check whether there is a literal $l'' \in C$ marked with label l. If there is, then C is a hidden tautology. In order to make this procedure sound, we need to add a unit clause (l) in case $\bar{l} \in \mathrm{HLA}(F, (l))$. Notice that this 'trick' cannot be used for HBCE. So, $\mathrm{HLA}(F, C)$ needs to be explicitly computed to check whether $\mathrm{HLA}(F, C)$ is a hidden blocked clause. This makes our current implementation of HBCE much more costly (compared to HTE). Also, while performing HBCE, some clauses can become hidden blocked clauses. Therefore, when run until fixpoint, multiple loops through the clauses are required (in contrast to HTE). As a result of this, our current implementation of HBCE is on average ten times as slow as the implementation of HTE, making HBCE at the moment impractical. However, as stated above, the cost of HTE and HBCE can be reduced by first applying ELS.

9 Conclusions

We introduced novel clause elimination procedures as hidden and asymmetric variants of the known techniques of tautology, subsumption, and blocked clause elimination. We analyzed all of the variants from various perspectives—relative effectiveness, BCP-preservance, confluence, logical equivalence—highlighting intricate differences between the procedures. This also resulted in a relative effectiveness hierarchy, in which the asymmetric variant of blocked clause elimination dominates all other procedures.

As one of the most interesting results, we developed $e\mathrm{HTE}$, a variant of hidden tautology elimination, that is both BCP-preserving and confluent, and at the same time more effective than the other procedures (tautology and subsumption elimination) that have both of these properties. In fact, $e\mathrm{HTE}$ does a transitive reduction (a structural property) of the binary implication graph underlying any CNF formula purely on the CNF level. Furthermore, we showed how to reconstruct solutions for the procedures, and presented experimental results on the practical effectiveness of selected procedures.

Efficient implementations of the introduced procedures and integration of the most practical ones with other simplification techniques remains as important further work.

References

1. Freeman, J.: Improvements to propositional satisfiability search algorithms. PhD thesis, University of Pennsylvania (1995)
2. Le Berre, D.: Exploiting the real power of unit propagation lookahead. Electronic Notes in Discrete Mathematics **9** (2001) 59–80
3. Lynce, I., Marques-Silva, J.: The interaction between simplification and search in propositional satisfiability. In: CP'01 Workshop on Modeling and Problem Formulation. (2001)
4. Bacchus, F.: Enhancing Davis Putnam with extended binary clause reasoning. In: Proc. AAAI, AAAI Press (2002) 613–619
5. Subbarayan, S., Pradhan, D.K.: NiVER: Non-increasing variable elimination resolution for preprocessing SAT instances. In: Proc. SAT. Volume 3542 of LNCS., Springer (2005) 276–291
6. Gershman, R., Strichman, O.: Cost-effective hyper-resolution for preprocessing CNF formulas. In: Proc. SAT. Volume 3569 of LNCS., Springer (2005) 423–429
7. Eén, N., Biere, A.: Effective preprocessing in SAT through variable and clause elimination. In: Proc. SAT. Volume 3569 of LNCS., Springer (2005) 61–75
8. Gelder, A.V.: Toward leaner binary-clause reasoning in a satisfiability solver. Annals of Mathematics and Artificial Intelligence **43**(1) (2005) 239–253
9. Jin, H., Somenzi, F.: An incremental algorithm to check satisfiability for bounded model checking. Electronic Notes in Theoretical Computer Science **119**(2) (2005) 51–65
10. Han, H., Somenzi, F.: Alembic: An efficient algorithm for CNF preprocessing. In: Proc. DAC, IEEE (2007) 582–587
11. Järvisalo, M., Biere, A., Heule, M.: Blocked clause elimination. In: Proc. TACAS. Volume 6015 of LNCS., Springer (2010) 129–144
12. Kullmann, O.: On a generalization of extended resolution. Discrete Applied Mathematics **96–97** (1999) 149–176
13. Aho, A., Garey, M., Ullman, J.: The transitive reduction of a directed graph. SIAM Journal on Computing **1**(2) (1972) 131–137
14. Järvisalo, M., Biere, A.: Reconstructing solutions after blocked clause elimination. In: Proc. SAT. Volume 6175 of LNCS., Springer (2010) 340–345