Reconstructing Solutions after Blocked Clause Elimination*

Matti Järvisalo[1] and Armin Biere[2]

[1] Department of Computer Science, University of Helsinki, Finland
[2] Institute for Formal Models and Verification, Johannes Kepler University Linz, Austria

Abstract. Preprocessing has proven important in enabling efficient Boolean satisfiability (SAT) solving. For many real application scenarios of SAT it is important to be able to extract a full satisfying assignment for original SAT instances from a satisfying assignment for the instances after preprocessing. We show how such full solutions can be efficiently reconstructed from solutions to the conjunctive normal form (CNF) formulas resulting from applying a combination of various CNF preprocessing techniques implemented in the PrecoSAT solver—especially, blocked clause elimination combined with SatElite-style variable elimination and equivalence reasoning.

1 Introduction

CNF-level preprocessing has proven important in enabling efficient SAT solving. This is highlighted for instance by PrecoSAT [3]—one of the most successful SAT solvers in the 2009 SAT Competition—that applies a combination of different preprocessing techniques both before and during search. On the other hand, for many real applications scenarios it is important to be able to extract a full satisfying assignment for the original instances from satisfying assignments for preprocessed instances. However, CNF-level preprocessing/simplification techniques, such as SatElite-style variable elimination [1] and blocked clause elimination [2], often preserve only satisfiability, not the set of satisfying assignments. Especially, reconstruction of an original solution becomes non-straightforward when applying combinations of preprocessing techniques.

In this paper we show how such full satisfying assignments can be efficiently reconstructed from solutions to the CNFs resulting from applying combinations of various preprocessing techniques. Especially, we concentrate on the non-trivial case of combining blocked clause elimination [2] (BCE)—which has proven surprisingly powerful, being able to achieve the same level of simplification as the Plaisted-Greenbaum polarity-based CNF encoding and a combination of specific circuit-level simplification techniques— with SatElite-style variable elimination and equivalence reasoning [3–5]. We explain how solution reconstruction is done in practice in PrecoSAT, and formally justify the correctness of this process. The presented reconstruction techniques are both time and space wise linear, and hence have no real overhead w.r.t. solving.

* First author financially supported by Academy of Finland (grant 132812).

[3] See `http://fmv.jku.at/precosat`

2 Preliminaries

CNF. For a Boolean variable x, there are two *literals*, the positive literal, denoted by x, and the negative literal, denoted by $\bar{x}$, the *negation of* x. A *clause* is a disjunction of distinct literals and a CNF formula is a conjunction of clauses. When convenient, a clause is seen as a finite set of literals and a CNF formula as a finite set of clauses. A clause is a *tautology* if it contains both x and $\bar{x}$ for some variable 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 is satisfied by τ if it contains at least one literal l such that $\tau(l) = \mathbf{t}$. An assignment τ *satisfies* F if it satisfies every clause in F. Finally, given an assignment τ, let τ_x (resp., $\tau_{\bar{x}}$) denote the assignment for which $\tau_x(x) = \mathbf{t}$ (resp., $\tau_{\bar{x}}(x) = \mathbf{f}$) and which otherwise is identical to τ.

Resolution. The resolution rule states that, given two clauses $C_1 = \{x, a_1, \ldots, a_n\}$ and $C_2 = \{\bar{x}, b_2, \ldots, b_m\}$, 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 variable x. We write $C = C_1 \otimes_x C_2$. A sequence of clauses $(C_0, C_1, \ldots, C_n)$ is a resolution derivation of the clause C from a CNF formula F if (i) $C_n = C$, and (ii) each C_i, where $0 \leq i < n$, is either a clause in F (in this case C_i is called an *input clause*), or C_i is the resolvent of two clauses C_j and C_k, where $j, k < i$. We denote by $F \vdash C$ the fact that there is a resolution derivation of the clause C from the CNF formula F. A well-known refinement of resolution is tree-like resolution, where derivations have to be representable as trees.

Variable Elimination as SatElite-style Preprocessing. Following the Davis-Putnam procedure [6] (DP), a preprocessing technique VE, referred to as *variable elimination by clause distribution* in [1], can be defined. For a CNF formula F, let $S_x \subseteq F$ and $S_{\bar{x}} \subseteq F$ consist of all the clauses in F that contain the literal x and $\bar{x}$, respectively. 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 $S_{\bar{x}}$. Formally, the resolution operator $\otimes$ can be lifted to sets of clauses:

$$S_x \otimes_x S_{\bar{x}} = \{C_1 \otimes_x C_2 \mid C_1 \in S_x, C_2 \in S_{\bar{x}}, \text{ and } C_1 \otimes_x C_2 \text{ is not a tautology}\}.$$

Now, replacing the original clauses in $S_x \cup S_{\bar{x}}$ with the set $S = S_x \otimes_x S_{\bar{x}}$ of non-tautological resolvents gives the CNF $(F \setminus (S_x \cup S_{\bar{x}})) \cup S$ which is satisfiability-equivalent to F. Since DP is a complete proof procedure for CNFs, with exponential worst-case space complexity, for practical applications as a preprocessing technique, variable elimination needs to be bounded; e.g., SatElite eliminates a considered variable only when the resulting CNF formula $(F \setminus (S_x \cup S_{\bar{x}})) \cup S$ will not contain more clauses as the original formula F. For the following, let $\mathrm{VE}(F, x)$ denote the result of applying variable elimination to F w.r.t x.

Blocked Clause Elimination (BCE) is a satisfiability-preserving CNF preprocessing techniques which removes so called *blocked clauses* [7] from CNF formulas.

Definition 1. *A literal l in a clause C of a CNF F* blocks *C w.r.t. 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.*

With respect to a fixed CNF and its clauses we have:

Definition 2. *A clause is* blocked *if it has a literal that blocks it.*

Example 1. Consider the formula $F_{\text{blocked}} = (a \vee b) \wedge (a \vee \bar{b} \vee \bar{c}) \wedge (\bar{a} \vee c)$. Only the first clause of F_{blocked} is not blocked. The second clause contains two blocked literals: a and $\bar{c}$. Also literal c in the last clause is blocked. Notice that after removing either $(a \vee \bar{b} \vee \bar{c})$ or $(\bar{a} \vee c)$, the clause $(a \vee b)$ becomes blocked. This is actually an extreme case in which BCE can remove all clauses of a formula, resulting in a trivially satisfiable formula. □

In the example, notice that although BCE alone can show that the original formula is satisfiable, a solution to the original CNF is not directly available.

Recent work [2] shows that, although a simple technique, BCE is surprisingly powerful. For example, without any circuit-level information, on the standard Tseitin CNF encoding BCE can achieve at least the same level of simplification as the Plaisted-Greenbaum polarity-based CNF encoding and a combination of specific circuit-level simplification techniques. Moreover, as shown in [2], BCE and SatElite-style variable elimination are to some extend orthogonal preprocessing techniques, which justifies combining these techniques for even more effective preprocessing. Notice also that, in contrast to variable elimination, BCE has a unique fixpoint for any CNF formula, i.e., BCE is confluent. This is due to the following.

Proposition 1 ([7]). *Given a CNF formula F, let clause $C \in F$ be blocked with respect to F. Any clause $C' \in F$, where $C' \neq C$, that is blocked with respect to F is also blocked with respect to $F \setminus \{C\}$.*

Exploiting Equivalent Literals. For two literals l_1 and l_2, let $l_1 \equiv l_2$ denote the CNF formula $\{\{l_1, \bar{l}_2\}, \{\bar{l}_1, l_2\}\}$. For a given CNF formula F, if $F \vdash l_1 \equiv l_2$, the equivalent literals l_1 and l_2 can be exploited by the equivalence reduction in which all occurrences of l_2 are substituted by l_1 (or vice versa), eliminating the variable of l_2 (or l_1). For example, *hyper binary resolution*, in which the clause $\{l, l'\}$ can be derived in one step from the clauses $\{l, l_1, \ldots, l_n\}$ and $\{\bar{l}_i, l'\}$ where $1 \leq i \leq n$, can be used to derive new binary clauses [3–5].

For detecting and exploiting equivalent literals in preprocessing/simplification, PrecoSAT implements a lazy version of hyper binary resolution. It also finds equivalent literals during failed literal probing. Equivalences are represented with a union find data structure. During garbage collection, not only top-level satisfied clauses are removed but all their literals are mapped to their representatives of the union find data structure. This essentially removes equivalent literals from the CNF; afterwards, only the representatives remain.

3 Solution Reconstruction for Individual Techniques

In this section we describe how to reconstruct solutions for each of the considered preprocessing techniques separately. We start with variable elimination for which reconstruction can be seen as part of the completeness proof of DP.

Proposition 2. *Let τ be a satisfying assignment for $\mathrm{VE}(F, x)$. Either τ_x or $\tau_{\bar{x}}$ satisfies $S_x \cup S_{\bar{x}}$, and, the one that does, also satisfies $F = \mathrm{VE}(F, x) \cup (S_x \cup S_{\bar{x}})$.*

To reconstruct a solution after VE has been applied repeatedly for the variables $x_1, \ldots, x_m$, it is enough to save (remember) the clauses $(S_{x_1} \cup S_{\bar{x}_1}), \ldots, (S_{x_m} \cup S_{\bar{x}_m})$. Assume that τ satisfies $\mathrm{VE}(\cdots \mathrm{VE}(\mathrm{VE}(F, x_1), x_2) \cdots, x_m)$. Let $\tau^{m+1} = \tau$, and, iteratively from $i = m$ to 1, define τ^i as the one of $\tau^{i+1}_{x_i}$ and $\tau^{i+1}_{\bar{x}_i}$ which satisfies $(S_{x_m} \cup S_{\bar{x}_m})$. Proposition 2 guarantees that τ^1 is a satisfying assignment for the original formula F.

If the application only requires to reconstruct one solution, then in practice[4] it is enough to only save either S_{x_i} or $S_{\bar{x}_i}$. W.l.o.g. assume S_{x_i} is saved. Then, if $\tau^{i+1}_{\bar{x}_i}$ satisfies the saved S_{x_i}, we pick $\tau^i = \tau^{i+1}_{\bar{x}_i}$, since this truth assignment obviously satisfies $S_{\bar{x}_i}$ as well. Otherwise x_i is forced to be **t** and we must set $\tau^i = \tau^{i+1}_{x_i}$. This case occurs if and only if there is a clause in S_{x_i} for which τ^{i+1} assigns all literals except x_i to **f**.

In an actual implementation only the smaller of the two sets is saved. Thus this technique is also efficient in the case where VE is used for pure literal elimination as discussed in [2]. In addition to plain VE, it also works for functional substitution [1] as in the SatElite preprocessor. The only difference between VE and functional substitution is that the latter removes some redundant clauses from $S_x \otimes_x S_{\bar{x}}$ while maintaining the set of satisfying assignments.

Equivalent literals are substituted by their representatives during preprocessing. Clauses used to derive equivalent literals become trivial and are removed during garbage collection. However, the relation between original literals and their representatives is maintained. If a satisfying assignment for the remaining clauses is found, the truth value of a substituted literal is defined to be the value of its representative. This extends the satisfying assignment for the remaining clauses to a satisfying assignment for the original formula.

Finally, consider BCE. In analogy to the case of VE, the proof [7] which shows that removal of a blocked clause does not turn an unsatisfiable formula into a satisfiable formula, gives us grounds to reconstruct solutions for BCE.

Proposition 3. *Assume that literal l blocks C w.r.t. F. Let τ be a satisfying assignment for $F \setminus \{C\}$. If τ does not satisfy C, then τ_l satisfies both $F \setminus \{C\}$ and C and thus F.*

In practice it is enough to save all removed blocked clauses $C_1, \ldots, C_m$ together with their blocking literals $l_1, \ldots, l_m$.[5] Let τ^m be a satisfying assignment for F_m, where $F_i = F \setminus \cup_{j=1}^{i} \{C_j\}$ for $i = 1 \ldots m$ and $F_0 = F$. If τ^i satisfies C_i, we pick $\tau^{i-1} = \tau^i$, and otherwise $\tau^{i-1} = \tau^i_{l_i}$. Using Proposition 3, one can show by induction that τ^i satisfies F_i, and thus τ^0 is a satisfying assignment for F.

4 Combined Solution Reconstruction

First, BCE and VE can be combined by saving clauses for reconstructing solutions after BCE resp. VE on the same reconstruction stack. Reconstruction works in reverse order

[4] By private communication with Niklas Sörensson.

[5] A space efficient way to save this information is to maintain l_i as the first literal in the saved clause C_i. This also allows to keep track of eliminated variables in VE.

in which these clauses have been saved. This also works nicely if BCE is applied on-the-fly during VE: while counting the non-trivial resolvents of $S_x \otimes_x S_{\bar{x}}$ to determine whether VE is applied to x, it may occur that a clause $C \in (S_x \cup S_{\bar{x}})$ has only trivial resolvents w.r.t. x, even though the overall number of non-trivial resolvents exceeds $|S_x \cup S_{\bar{x}}|$, which prevents x from being eliminated. Yet C can be removed as a blocked clause and is saved on the reconstruction stack.

In order to combine VE and equivalent reasoning it is enough to make sure that VE is only attempted after all equivalent literals have been first substituted. Enforcing this order of using equivalent literal reasoning and VE makes sure that variables eliminated with VE are always representatives and the only remaining variables of their equivalence class. Eliminating a representative through VE will eliminate its whole equivalence class, and after this it is not possible that further equivalent literals could be added to the equivalence class of an eliminated variable.

When combining BCE with equivalent literal reasoning, however, the situation is different: at some point after removing a blocked clause C, a literal l which blocked C may become equivalent to another literal and may even become a representative of its equivalence class. On the other hand, one may be forced to flip the value of l during solution reconstruction since BCE removed C (recall Sect. 3). Hence the values of all the literals in the equivalence class should be flipped, which appears not to be sound since this could make some other clause unsatisfied. However, as we show in the next section, the value of l will never have to be flipped in such a situation.

5 Equivalent Literals and Blocked Clause Satisfiability

Equivalent literals detected and applied in simplifying a CNF after removing blocked clauses cannot make the removed blocked clauses to not to be satisfied under a satisfying assignment for the rest of the formula.

Theorem 1. *Assume a CNF formula F, a clause $C \in F$ which is blocked for $l \in C$ w.r.t. F, and a literal l'. If $F \setminus \{C\} \vdash l \equiv l'$, then $(F \setminus \{C\}) \cup (l \equiv l') \models C$.*

In other words, any satisfying assignment for $(F \setminus \{C\}) \cup (l \equiv l')$ also satisfies the blocked clause C. This means that binary equivalences detected during preprocessing can be exploited when applying BCE, at the same time guaranteeing all the blocked clauses removed by BCE will be satisfied by any satisfying assignment for the resulting preprocessed CNF formula. Notice that this lemma is independent of the techniques used for deriving the clauses in $l \equiv l'$.

Proof (of Theorem 1). Assume a CNF formula F, a clause $C = \{l, l_1, \ldots, l_k\} \in F$ which is blocked for $l \in C$ w.r.t. F. Denote by $B \subset F$ the set of clauses which contain the literal $\bar{l}$. Hence each clause in B contains at least one of the literals $\bar{l}_1, \ldots, \bar{l}_k$. Assume that $F \setminus \{C\} \vdash l \equiv l'$ for some literal l', and hence there is a resolution derivation of $\{l, \bar{l}'\}$ and $\{\bar{l}, l'\}$ from $F \setminus \{C\}$.

If F is unsatisfiable, $F \setminus \{C\}$ is also unsatisfiable since C is blocked, and hence trivially $(F \setminus \{C\}) \cup (l \equiv l') \models C$. Now consider the case that F and (thus) also $F \setminus \{C\}$

and $(F \setminus \{C\}) \cup (l \equiv l')$ are satisfiable. Take an arbitrary satisfying assignment τ for $(F \setminus \{C\}) \cup (l \equiv l')$. We will show that any such τ also satisfies C.

The case in which $\tau(l) = \mathbf{t}$ (that is, τ satisfies l) is trivial. Now assume $\tau(l) = \mathbf{f}$. Then $\tau(l') = \mathbf{f}$ since τ satisfies $l \equiv l'$. Consider an arbitrary resolution derivation $\pi = (C_1, \ldots, C_m)$ of $C_m = \{\bar{l}, l'\}$ from $F \setminus \{C\}$. Assume w.l.o.g. that π is tree-like. We claim that there is an input clause $C' = \{\bar{l}, l'_1, \ldots, l'_k\} \in B$ in π such that $\tau(l'_i) = \mathbf{f}$ for all i. Since $C' \in B$, it then follows that one of the l'_is is one of the literals $\bar{l}_1, \ldots, \bar{l}_k$, and hence τ satisfies C (recall that $C = \{l, l_1, \ldots, l_k\}$).

To prove the claim, we show that there is a path $P_1, \ldots, P_n$ of clauses in π (seen as a tree) from the root of the tree ($P_1 = C_m$) to a leaf (P_n is an input clause of π), such that each clause P_i on the path contains $\bar{l}$ and τ assigns all literals in P_i except $\bar{l}$ to $\mathbf{f}$.

First notice that for $P_1 = C_m$ we know that $\tau(\bar{l}) = \mathbf{t}$ and $\tau(l') = \mathbf{f}$. Now assume that $P_i = \{\bar{l}\} \cup D$, where D is a set of literals such that τ assigns every literal in D to $\mathbf{f}$, was directly derived from clauses C_a and C_b in π resolving on the variable x. Notice that at least one of C_a and C_b must contain $\bar{l}$. First consider the case that C_a contains $\bar{l}$ and C_b does not. Since τ assigns all literals in D to $\mathbf{f}$, τ must satisfy the literal for x in C_b. (Otherwise τ does not satisfy C_b which would imply that τ does not satisfy an input clause in π and hence τ cannot be a satisfying truth assignment for $(F \setminus \{C\}) \cup (l \equiv l')$, in contradiction to our assumption.) Hence τ assigns all literals in C_a apart from $\bar{l}$ to $\mathbf{f}$. In this case let $P_{i+1} = C_a$. The case that C_b contains $\bar{l}$ and C_a does not is identical.

Now consider the case that both C_a and C_b contain $\bar{l}$. Since τ assigns a unique truth value to x, τ assigns all literals in either C_a or C_b apart from $\bar{l}$ to $\mathbf{f}$. In this case let P_{i+1} be this particular clause. □

6 Conclusions

We showed how and why—in theory and in practice—full solutions to CNF formulas can be reconstructed from solutions to the CNF after applying both individual and combinations of preprocessing techniques, including blocked clause elimination, SatElite-style variable elimination and equivalence reasoning.

References

1. Eén, N., Biere, A.: Effective preprocessing in SAT through variable and clause elimination. In: Proc. SAT'05. Volume 3569 of LNCS., Springer (2005) 61–75
2. Järvisalo, M., Biere, A., Heule, M.: Blocked clause elimination. In: Proc. TACAS 2010. Volume 6015 of LNCS., Springer (2010) 129–144
3. Bacchus, F.: Enhancing Davis Putnam with extended binary clause reasoning. In: Proc. AAAI 2002. AAAI Press (2002) 613–619
4. Bacchus, F., Winter, J.: Effective preprocessing with hyper-resolution and equality reduction. In: Proc. SAT 2003. Volume 2919 of LNCS., Springer (2004) 341–355
5. Gershman, R., Strichman, O.: Cost-effective hyper-resolution for preprocessing CNF formulas. In: Proc. SAT 2005. Volume 3569 of LNCS., Springer (2005) 423–429
6. Davis, M., Putnam, H.: A computing procedure for quantification theory. Journal of the ACM **7**(3) (1960) 201–215
7. Kullmann, O.: On a generalization of extended resolution. Discrete Applied Mathematics **96–97** (1999) 149–176