Bounded clause elimination

Jules Jacobs

May 7, 2021

Bounded variable elimination and blocked clause elimination are two effective SAT preprocessing techniques. In this note I define *bounded clause elimination*, which generalizes both.

Given a CNF formula F and a clause $c \in F$ and a literal $l \in c$, define $\mathsf{elim}(F, c, l)$ to be the CNF formula F with clause c *replaced* by all resolvents of c along l.

The formula F consists of clause c, clauses that contain l, clauses that contain $\neg l$, and clauses that contain neither l nor $\neg l$:

$$F = (l \vee \vec{c}) \wedge (\bigwedge_i l \vee \vec{a}_i) \wedge (\bigwedge_j \neg l \vee \vec{b}_j) \wedge (\bigwedge_k \vec{d}_k)$$

Now $\mathsf{elim}(F, c, l)$ is:

$$\mathsf{elim}(F, c, l) = (\bigwedge_j \vec{c} \vee \vec{b}_j) \wedge (\bigwedge_i l \vee \vec{a}_i) \wedge (\bigwedge_j \neg l \vee \vec{b}_j) \wedge (\bigwedge_k \vec{d}_k)$$

It is clear that $F \implies \mathsf{elim}(F, c, l)$ because we've only added resolvents, but the reverse implication does not hold because we've deleted the clause $l \vee \vec{c}$. Take $F = l$, for example; then eliminating the only clause l gives us the empty CNF, which is satisfied for any variable assignment, whereas F is only satisfied for $l = 1$. However, the two formulas are equisatisfiable.

Lemma 1. *F and $\mathsf{elim}(F, c, l)$ are equisatisfiable.*

Proof. Since $F \implies \mathsf{elim}(F, c, l)$, it suffices to show that any assignment for $\mathsf{elim}(F, c, l)$ can be turned into an assignment for F. If the clause $l \vee \vec{c}$ is satisfied by the assignment for $\mathsf{elim}(F, c, l)$, then we can use the same assignment to satisfy F, because the remaining clauses in F are also in $\mathsf{elim}(F, c, l)$. So suppose $l = 0$ and $\vec{c} = 0$ in the assignment that satisfies $\mathsf{elim}(F, c, l)$. Then $\mathsf{elim}(F, c, l)$ simplifies to:

$$\mathsf{elim}(F, c, l) = (\bigwedge_j \vec{b}_j) \wedge (\bigwedge_i \vec{a}_i) \wedge (\bigwedge_k \vec{d}_k)$$

Given this assignment for all variables except l, the formula F simplifies to:

$$F = (l \vee \vec{c})$$

Hence the same assignment but with $l = 1$ instead of $l = 0$ satisfies F. $\square$

The proof of this lemma gives us a method to reconstruct solutions for F from solutions for $\mathsf{elim}(F, c, l)$: if the clause we eliminated is already satisfied, do nothing, and otherwise flip the value of l.

We can do *bounded clause elimination* by heuristically picking clauses to eliminate. We can simulate both blocked clause elimination and bounded variable elimination using elim:

- **Blocked clause elimination** deletes a clause c if there is a literal $l \in c$ such that all resolvents of c along l are tautologies. This is equivalent to *replacing* c by the resolvents.
- **Bounded variable elimination** chooses a literal l are replaces all clauses involving l by all their resolvents. This is the same as running clause elimination multiple times, once for each clause that contains l.

Clause deletion

A slightly different perspective is clause deletion: when is it safe to delete a clause? Deleting a clause may increase the number of satisfying assignments, but that is fine as long as (a) it doesn't turn an UNSAT problem into a SAT problem and (b) we have a method to reconstruct a satisfying assignment for the original problem from a satisfying assignment for the new problem.

The argument above shows that it is safe to delete a clause c when all its resolvents along l are implied by the remaining clauses. The solution reconstruction method is the same: if c is not satisfied, flip l.

We can still simulate bounded variable elimination: first add all resolvents, and now we can delete the original clauses because all their resolvents are (trivially) implied.

Implementation in a solver

- Keep track of a stack of deleted clauses, and which literal l was used to delete it.
- We can delete a clause at any time if its resolvents along some l are implied by permanent clauses.
- Whenever the user adds a new clause containing $\neg l$, restore all clauses that were deleted using l. (Adding the assumption $l = 0$ can be treated as adding the unit clause $\neg l$.)
- To reconstruct the original solution, pop all deleted clauses from the stack, flipping l if necessary to make the clause satisfied.