Generalized Conflict-Clause Strengthening for Satisfiability Solvers

Allen Van Gelder
Computer Science Dept.
Univ. of California
Santa Cruz, CA, USA

`http://www.cse.ucsc.edu/~avg/`

`http://www.cse.ucsc.edu/~avg/Papers/`
These slides are `generalized-trans.pdf`

`http://www.cse.ucsc.edu/~avg/ProofChecker/`
Software directory, contains `MiniSat2ccmin.tar`
as well as individual files.

The Problem:

In a Satisfiability solver using the CDCL strategy (Grasp, Chaff, MiniSat, etc.), derive more useful conflict clauses.

More useful how?

- Stronger constraint: proper subset of a standard conflict clause
- Enable further backtracking

Main Idea in This Paper: Use "*volunteer*" clauses in addtion to antecedent clauses.

Related Work:

A. Biere, "Picosat Essentials", *JSAT* 2008.

G. Audemard, L. Bordeaux, Y. Hamadi, S. Jabbour, and L. Sais, A Generalized Framework for Conflict Analysis, *SAT* 2008.

N. Sörensson and A. Biere, Minimizing Learned Clauses, *SAT* 2009.

A. Van Gelder, Improved Conflict-Clause Minimization Leads to Improved Propositional Proof Traces, *SAT* 2009.

Volunteers and Implication Sequences

Implication Sequence:
A bookkeeping notation for Conflict Driven Clause Learning (CDCL) solvers.

Three Clause Types:

- Assumption “clause”: $[p, \top]$ records that p was assumed.
- Antecedent clause $C(q)$: *Earliest* to imply literal q:
    - All literals in C are assigned;
    - Literal q is assigned *true* (q may be $\bot$);
    - All literals in C are other than q are assigned *false*.
- Volunteer clause $C(q)$: *Re-implies* literal q, which was already implied which was already implied or assumed:
    - All literals in C are assigned;
    - Literal q is assigned *true* (q may be $\bot$);
    - All literals in C are other than q are assigned *false*.

The sequence of clauses records the order in which the solver took actions leading to a conflict.

Propagation Sequence: Subsequence beginning with an assumption and ending just before the next assumption, or at the end of the implication sequence.

Previously Known Properties of Implication Sequences

- If volunteers are discarded and a UIP conflict clause γ is derived from antecedents, then there is a unique minimum subset of γ that is logically implied by the antecedents, and it may be found in polynomial time. [Sörensson–Biere, 2009].
- The above minimum conflict clause may be derived in polynomial time with *Linear Input Regular* (LIR) resolution [Van Gelder, 2009].
  - LIR does not resolve on any variable more than once.
- If a UIP conflict clause γ is derived from antecedents only, then including certain volunteers may permit derivation of another conflict clause with the same asserting literal, but which allows further backtracking than the above minimum conflict clause. [Audemard *et al.*, 2008].
  - Slightly positive empirical results were shown.
  - Unpublished variant used in `precosat` (`http://fmv.jku.at`).

Properties of Implication Sequences Shown in This Paper

- Every implication sequence is *Renamable Horn*.
- Every clause that is logically implied by an implication sequence has a LIR resolution derivation.
- Given an implication sequence and a UIP conflict clause γ that is derived using antecedents only, and given an integer K, the question of whether some subset of γ with K literals can be derived is NP-complete.
  - Reduction from Hitting Set.
  - Similar theorem in [Kleine Büning–Lettmann, 1999].

Take-Home Message:

Conflict-clause minimization with antecedents only is a *solved problem*, but conflict-clause minimization including all volunteers is *intractable*.

Example, Implication Sequence and Traditional CDCL

Clauses (part of a formula):

$C_1 = [\overline{v}, \overline{x}, \overline{y}, \overline{z}]$ $C_2 = [y, \overline{u}, \overline{w}, \overline{x}]$ $C_3 = [z, \overline{x}, \overline{v}]$ $C_5 = [u, \overline{v}, \overline{w}]$ $C_6 = [x, \overline{t}, \overline{u}]$ $C_7 = [w, \overline{v}, \overline{z}]$

Implication sequence, with one level per line:

$$
\begin{array}{lllllll}
1 & [v, \top] & & & & & \\
2 & [w, \top], & C_5(u) & & & & \\
3 & [t, \top], & C_6(x) & C_2(y) & C_3(z) & C_7(w) & C_1(\bot)
\end{array}
$$

C_5 becomes a unit clause at level 2 with u as the implied literal, then becomes the antecedent as u is assigned *true*.

C_7 is a volunteer because it re-implies w.

$$
\begin{aligned}
D_1 &= \mathbf{res}_y(C_2, C_1) = [\bot, \overline{v}, \overline{x}, \overline{u}, \overline{w}, \overline{z}] \\
D_2 &= \mathbf{res}_z(C_3, D_1) = [\bot, \overline{v}, \overline{x}, \overline{u}, \overline{w}]
\end{aligned}
$$

D_2 is the 1-UIP clause. Minimize it by resolving with antecedent C_5:

$$
D_3 = \mathbf{res}_u(C_5, D_2) = [\bot, \overline{v}, \overline{x}, \overline{w}]
$$

Backtrack to level 2, so w remains assigned.

Example, Implication Sequence and Volunteers

Clauses (part of a formula) and implication sequence, repeated:

$C_1 = [\overline{v}, \overline{x}, \overline{y}, \overline{z}]$ $C_2 = [y, \overline{u}, \overline{w}, \overline{x}]$ $C_3 = [z, \overline{x}, \overline{v}]$ $C_5 = [u, \overline{v}, \overline{w}]$ $C_6 = [x, \overline{t}, \overline{u}]$ $C_7 = [w, \overline{v}, \overline{z}]$

$$\begin{array}{ll} 1 & [v, \top] \\ 2 & [w, \top], \quad C_5(u) \\ 3 & [t, \top], \quad C_6(x) \quad C_2(y) \quad C_3(z) \quad C_7(w) \quad C_1(\bot) \end{array}$$

C_7 is a volunteer (inverse arc per Audemard *et al.*) because it re-implies w.

$$\begin{aligned} D_1 &= \mathbf{res}_y(C_2, C_1) = [\bot, \overline{v}, \overline{x}, \overline{u}, \overline{w}, \overline{z}] \\ D_2 &= \mathbf{res}_z(C_3, D_1) = [\bot, \overline{v}, \overline{x}, \overline{u}, \overline{w}] \\ D_3 &= \mathbf{res}_u(C_5, D_2) = [\bot, \overline{v}, \overline{x}, \overline{w}] \end{aligned}$$

Because $\overline{w}$ prevents further backtracking, resolve it away with the volunteer, C_7:

$$\begin{aligned} D_6 &= \mathbf{res}_w(C_7, D_3) = [\bot, \overline{v}, \overline{x}, \overline{z}] \\ D_7 &= \mathbf{res}_z(C_3, D_6) = [\bot, \overline{v}, \overline{x}] \end{aligned}$$

D_6, is non-asserting, due to $\overline{z}$ at level 3 being re-introduced.
So C_3 must be used to resolve out the extra level-3 literal, producing D_7.

Backtrack to level 1, so v remains assigned.

Example, Implication Sequence, Volunteers, and LIR Proof

Clauses (part of a formula) and implication sequence, repeated:

$C_1 = [\overline{v}, \overline{x}, \overline{y}, \overline{z}]$ $C_2 = [y, \overline{u}, \overline{w}, \overline{x}]$ $C_3 = [z, \overline{x}, \overline{v}]$ $C_5 = [u, \overline{v}, \overline{w}]$ $C_6 = [x, \overline{t}, \overline{u}]$ $C_7 = [w, \overline{v}, \overline{z}]$

$$
\begin{array}{ll}
1 & [v, \top] \\
2 & [w, \top], \quad C_5(u) \\
3 & [t, \top], \quad C_6(x) \quad C_2(y) \quad C_3(z) \quad C_7(w) \quad C_1(\bot)
\end{array}
$$

C_7 is a volunteer because it re-implies w.

Here is a linear input regular (LIR) derivation from C_1, the falsified clause:

$$
\begin{aligned}
D_1 &= \mathbf{res}_y(C_2, C_1) = [\bot, \overline{v}, \overline{x}, \overline{u}, \overline{w}, \overline{z}] \\
D_8 &= \mathbf{res}_u(C_5, D_1) = [\bot, \overline{v}, \overline{x}, \overline{w}, \overline{z}] \\
D_9 &= \mathbf{res}_w(C_7, D_8) = [\bot, \overline{v}, \overline{x}, \overline{z}] \\
D_{10} &= \mathbf{res}_z(C_3, D_9) = [\bot, \overline{v}, \overline{x}]
\end{aligned}
$$

D_{10} is the same as D_7.

Conclusion and Future Work

Short proofs (LIR) always exist using volunteers.

Finding the width-optimum conflict clause with volunteers is intractable.

Future work should study using volunteers more effectively.