Conditional Lower Bounds for Failed Literals and Related Techniques*

Matti Järvisalo and Janne H. Korhonen

HIIT & Department of Computer Science, University of Helsinki, Finland

Abstract. We prove time-complexity lower bounds for various practically relevant probing-based CNF simplification techniques, namely failed literal detection and related techniques. Specifically, we show that improved algorithms for these simplification techniques would give a $2^{\delta n}$ time algorithm for CNF-SAT for some $\delta < 1$, violating the Strong Exponential Time Hypothesis.

1 Introduction

Automated formula simplification at the conjunctive normal form (CNF) level is today an integral part of the SAT solving workflow, often notably speeding up Boolean satisfiability (SAT) solving of real-world application instances. Indeed, various polynomial-time techniques have been proposed for simplifying CNF formulas before (i.e., in preprocessing) and during (i.e., in inprocessing [26]) search for satisfiability; see e.g. [1–3, 6, 11, 14, 15, 20, 21, 25, 29, 30, 33]. However, formula simplification tends to come with a price. While stronger simplification might be achieved by using more computational effort, in practice time used for simplification should not outweight the benefits of the simplifications in terms of the overall solving time (i.e., the combined time used for simplification and search). While SAT solver developers keep on searching for more efficient ways of implementing simplification techniques, our formal understanding of the time complexity of different simplification techniques is rather limited at present. This paper take steps towards a more in-depth understanding of the price of simplification: we prove lower bounds for different *probing-based* CNF simplification techniques.

Unit propagation is a common basis for many different simplification techniques [3, 12, 13, 17, 18, 20, 21, 31, 32, 35, 38]. A key example is *failed literal elimination* [13, 31, 34], which aims at deducing unit clauses via checking whether assuming a truth value for a single variable results in a conflict by unit propagation. Failed literals is a key technique used during search within lookahead DPLL solvers [23] for both search tree pruning as well as a basis of branching heuristics [22, 27, 28, 31, 34]. Furthermore, in combination with conflict-driven

* This work was supported by Academy of Finland Finnish Centre of Excellence in Computational Inference Research COIN (grant #251170; M.J.) and Helsinki Doctoral Programme in Computer Science – Advanced Computing and Intelligent Systems (J.K.).

clause learning (CDCL) solvers, failed literals can be detected during preprocessing as well as during search, e.g., by inprocessing CDCL SAT solvers such as Lingeling [4]. Various clause elimination and clause strengthening techniques are essentially generalisations of failed literals, probing for either conflicts or specific literal dependencies using unit propagation by assuming one or more literals at a time.

1.1 Contributions

Our main result is a conditional lower bound for the *failed literal existence* problem, i.e., that of deciding whether a given CNF formula contains a failed literal. Since the fixpoint of unit propagation can be computed in time $O(n+m)$ on CNF formulas with n clauses and m variables, failed literal existence has a simple algorithm with running time $O\big(n(n+m)\big)$: for each literal $\ell \in F$, run unit propagation on $F \wedge (\ell)$ and see if a conflict is derived. An iterative application of this simple algorithm gives a $O\big(n^2(n+m)\big)$ algorithm for applying *failed literal elimination* until fixpoint.

In practice, the quadratic running time of the simple algorithm for failed literal existence can be too time consuming. However, as our main result, formalized as Theorem 1, we show that non-negligible improvements over the simple algorithm would give an improved algorithm for CNF-SAT.

Theorem 1. *Let $\varepsilon > 0$. If there is a $O\big((N+M)^{2-\varepsilon}\big)$ algorithm for failed literal existence on Horn-3-CNF formulas with N variables and M clauses, then there is a $2^{(1-\varepsilon/2)n}\operatorname{poly}(n,m)$ time algorithm for CNF-SAT on formulas with n variables and m clauses*

In other words, any such improvement, even in the restricted setting of Horn-3-CNF formulas, would give us a exponential speedup over brute force for CNF-SAT, improving upon the state of the art. Indeed, this would violate the *strong exponential time hypothesis (SETH)* [8, 24] stating that

$$\lim_{k\to\infty} \inf\{\delta \colon k\text{-CNF can be solved in time } O(2^{\delta n})\} = 1\,.$$

In particular, SETH would imply that CNF-SAT with unrestricted clause length cannot be solved in time $2^{(1-\varepsilon)n}\operatorname{poly}(n,m)$ for any $\varepsilon > 0$. Thus Theorem 1 gives a conditional lower bound against faster algorithms for failed literal existence.

Corollary 1. *Failed literal existence cannot be solved on Horn-3-CNF formulas with N variables and M clauses in time $O\big((N+M)^{2-\varepsilon}\big)$ for any $\varepsilon > 0$ unless SETH fails.*

This result falls in line with other recent work investigating lower bounds based on SETH [7, 9, 37]. While SETH itself is an extremely strong complexity assumption, our result can be interpreted as showing that any attempt to improve upon the simple $O\big(n(n+m)\big)$ algorithm for finding a failed literal faces barriers equivalent to improving the worst-case performance of CNF-SAT algorithms.

A detailed proof of Theorem 1 is presented in Section 3. As outlined in Section 4, minor variations of the proof also give the same quadratic lower bound for several related problems: checking the existence of *asymmetric tautologies* and *asymmetric literals*, as well as for checking whether a binary CSP restricted to domain-size 3 is singleton arc consistent.

2 Preliminaries

We assume that the reader is familiar with standard definitions on propositional satisfiability. When convenient, a clause is seen as a set of literals and a CNF formula as a set of clauses. Recall that the subclass k-CNF consists of CNF formulas consisting of clauses of length $\leq k$; Horn consists of CNF formulas in which each clause has at most one positive literal; and that Horn-k-CNF is the intersection of k-CNF and Horn.

Given a CNF formula F with clauses $(\neg l_1), \ldots, (\neg l_k)$, and $(l_1 \vee \cdots l_k \vee l_{k+1})$, the *unit resolution rule* allows to extend F by letting $F := F \wedge (l_{k+1})$, i.e., allows the derivation of the unit clause (l_{k+1}) from F in one step. *Unit propagation* on F applies the unit resolution rule until fixpoint, and we write $F \vdash_{\mathrm{up}} (l)$ if unit propagation on F derives the unit clause (l).

A literal $l \in F$ is a *failed literal* in F if unit propagation derives a conflict on $F \wedge (l)$, that is, we have $F \vdash_{\mathrm{up}} (\ell'), (\neg \ell')$ for some literal ℓ'. In particular, this implies that F is logically equivalent to $F \wedge (\neg l)$, which can be used to simplify F by letting $F := F \wedge (\neg l)$ if $l \in F$ fails in F; this is called the *failed literal rule. Failed literal elimination* refers to applying the failed literal rule until fixpoint.

3 Proof of the Failed Literal Existence Lower Bound

On a high level, our strategy for proving Theorem 1 follows that of Pătraşcu and Williams [37]. In particular, assume that the following conditions hold.

(i) For some $\varepsilon > 0$, there is a $O\big((N+M)^{2-\varepsilon}\big)$ algorithm for failed literal existence on Horn-3-CNF formulas with N variables and M clauses.
(ii) There is a reduction that maps a CNF formula F with n variables and m clauses to a Horn-3-CNF F_{fl} with N variables and M clauses such that
   ii.a F_{fl} has a failed literal if and only if F is satisfiable,
   ii.b $N, M \leq 2^{n/2} \operatorname{poly}(n,m)$, and
   ii.c F_{fl} can be constructed in time $2^{n/2} \operatorname{poly}(n,m)$.

Lemma 1. *Conditions (i) and (ii) imply that CNF-SAT has an algorithm with running time* $2^{(1-\varepsilon/2)n} \operatorname{poly}(n,m)$.

Proof. On input CNF formula F, we (1) construct F_{fl}, and (2) use the algorithm for failed literal existence to decide whether F_{fl} has a failed literal, and thus whether F is satisfiable. The first step takes $2^{n/2} \operatorname{poly}(n,m)$ time and the second step takes $O\big((N+M)^{2-\varepsilon}\big)$ time, that is, $2^{(1-\varepsilon/2)n} \operatorname{poly}(n,m)$ time. $\square$

Thus in order to prove Theorem 1, it suffices to construct a reduction satisfying Condition (ii) above. In what follows, we first present a *reduction template* (in Section 3.1) which can be used as a base construction for obtaining reductions from CNF-SAT to failed literal existence as well as other related existence problems. We then instantiate the reduction for failed literal existence and show how to obtain Horn-3-CNF formulas from the reduction (in Section 3.2). Instantiations for related problems, namely, the existence problem for asymmetric tautologies and literals, and checking singleton arc consistency of binary CSPs of domain size three, are presented in Section 4.

3.1 A Reduction Template

Let $F = C_1 \wedge C_2 \wedge \cdots \wedge C_m$ be a CNF formula over variables $x_1, x_2, \ldots, x_n$. Without loss of generality, we assume that n is even and $n \geq 4$. We split the variable set into *high* variables $x_1, x_2 \ldots, x_{n/2}$ and the *low* variables $x_{n/2+1}, x_{n/2+2} \ldots, x_n$. Let $P = \{p_1, p_2, \ldots, p_{2^{n/2}}\}$ be the set of all truth assignments into the high variables, and similarly let $Q = \{q_1, q_2, \ldots, q_{2^{n/2}}\}$ be the set of all truth assignments into the low variables. For $p \in P$ and $q \in Q$, denote by pq the assignment into variables $x_1, x_2, \ldots, x_n$ obtained by combining p and q.

We now construct a new formula F' from F as follows. The variable set of F' is

$$\{y_r \colon r \in P \cup Q\} \cup \{c_i \colon i = 1, 2, \ldots, m\} \cup \{w\}\,,$$

and the clauses of F' are given by the following three rules.

(1) For each partial assignment $p \in P$ and clause $C_i \in F$, if p satisfies C_i we include the clause $(\neg y_p \vee c_i)$, or equivalently, $(y_p \to c_i)$.
(2) For each partial assignment $q \in Q$, we include the clause

$$\Big(y_q \vee \bigvee_{i \colon q(C_i) \neq 1} \neg c_i\Big)\,, \qquad \text{or equivalently,} \qquad \Big(\big(\bigwedge_{i \colon q(C_i) \neq 1} c_i\big) \to y_q\Big)\,.$$

In words, this clause states that having $c_i = 1$ for all clauses $C_i \in F$ that are not satisfied by q implies $y_q = 1$. Without loss of generality, we will assume that the original formula F contains the tautological clauses $(x_1 \vee \neg x_1)$ and $(x_2 \vee \neg x_2)$. This ensures that clauses generated by this rule have length at least 3; in particular, they are not units.
(3) For each partial assignment $q \in Q$, we include the clause $(\neg y_q \vee w)$, or equivalently, $(y_q \to w)$.

Intuitively, the important feature of F' is how unit propagation behaves on the formula. The variables of F' can be seen to be arranged in layers, as illustrated in Figure 1. These layers are (1) the high variables $\{y_p \colon p \in P\}$, (2) the clause variables $\{c_1, c_2, \ldots, c_m\}$, (3) the low variables $\{y_q \colon q \in Q\}$, and (4) the terminal variable $\{w\}$. Clauses are implications from variables on layer i to a single variable on layer $i + 1$. As all the clauses are Horn, positive units

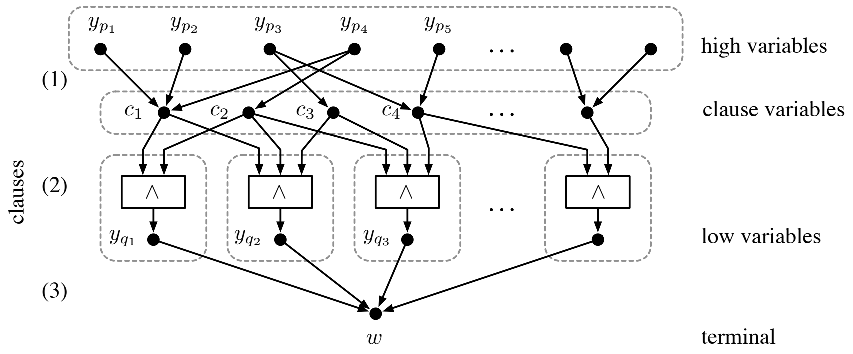

Fig. 1. Illustration of the reduction

will propagate only downwards towards the terminal w, and negative units will propagate only upwards toward the variables y_p.

Furthermore, the "satisfiability gadgets" consisting of the clauses (2) control the flow of unit propagation between variables y_p and w. That is, the only way for unit propagation to pass through these gadgets is that we start from a literal y_p for an assignment p that can be extended to a satisfying assignment for F.

More formally, we make the following observations about F'.

Lemma 2. *Given a CNF formula F with n variables and m clauses, F' is a CNF formula with N clauses and M variables such that*

(a) *F' is a Horn-CNF formula,*
(b) *F' is satisfied by assigning all variables to 0,*
(c) *$N = 2^{n/2} + m + 1$,*
(d) *$M = 2^{n/2}\operatorname{poly}(m)$, and*
(e) *F' can be constructed in time $2^{n/2}\operatorname{poly}(n, m)$.*

Lemma 3. *Let $p \in P$ be fixed.*

(a) *If there is a $q \in Q$ such that pq satisfies F, then $F' \wedge (y_p) \vdash_{\text{up}} (w)$.*
(b) *if there is no $q \in Q$ such that pq satisfies F, then unit propagation on $F' \wedge (y_p)$ derives exactly the units (c_i) for i such that $p(C_i) = 1$.*
(c) *Unit propagation on $F' \wedge (\neg w)$ derives exactly the units $(\neg y_q)$ for $q \in Q$.*

Proof. Fix $p \in P$ and $q \in Q$. We start by making the following simple observation: the assignment pq satisfies F if and only if $\{i \colon p(C_i) \neq 1\} \cap \{i \colon q(C_i) \neq 1\} = \emptyset$, which in turn is equivalent to $\{i \colon q(C_i) \neq 1\} \subseteq \{i \colon p(C_i) = 1\}$.

For (a), assume that there is $q \in Q$ such that pq satisfies F. For any i such that $p(C_i) = 1$, we have $(y_p \to c_i)$, and thus $F' \wedge (y_p) \vdash_{\text{up}} (c_i)$. Since pq satisfies F, we have by the earlier observation that $F' \wedge (y_p) \vdash_{\text{up}} (c_i)$ for all i such that $q(C_i) \neq 1$. Thus unit propagation derives (y_q) using the clause $\big(\bigwedge_{q(C_i)\neq 1} c_i\big) \to y_q$, and, further, (w) using the clause $(y_q \to w)$.

For (b), note that unit resolution on $F' \wedge (y_p)$ immediately derives (c_i) for i such that $p(C_i) = 1$, and no other units are immediately derived or already present in F'. Since pq does not satisfy F for any $q \in Q$, we have $\{i\colon p(C_i) \neq 1\} \cap \{i\colon q(C_i) \neq 1\} \neq \emptyset$. Thus, unit propagation does not derive (y_q) from the clause $\big(\bigwedge_{q(C_i)\neq 1} c_i\big) \rightarrow y_q$, nor does it derive (w).

For (c), note that $F' \wedge (\neg w) \vdash_{\mathrm{up}} (\neg y_q)$ for all $q \in Q$, using the clause $(y_q \rightarrow w)$. Since each clause $\big(\bigwedge_{q(C_i)\neq 1} c_i\big) \rightarrow y_q$ has length at least 3 and variables w and y_q do not appear in any other clauses, no further units are derived. $\square$

3.2 The Lower Bound

To complete the reduction to failed literal existence, we construct the formula F_{fl} by adding the clause $(\neg w \vee \neg y_p)$ or equivalently, $(w \rightarrow \neg y_p)$ to F' for each $p \in P$. Lemma 2 also holds for F_{fl}. Furthermore, we have the following.

Lemma 4. *Let ℓ be a literal in F_{fl}. We have that*

(a) *ℓ is a failed literal in F_{fl} if $\ell = y_p$ for some $p \in P$ and there is a $q \in Q$ such that pq satisfies F, and*
(b) *ℓ is not a failed literal otherwise.*

Proof. First, we note that (a) follows from Lemma 3(a), as we have $F_{\mathrm{fl}} \wedge (y_p) \vdash_{\mathrm{up}} (w)$ and $(w \rightarrow \neg y_p)$ if p satisfies the conditions of (a). Thus, it suffices to show that there are no other failed literals.

Now let ℓ be literal that does not satisfy the requirements of (a). We show that $F_{\mathrm{fl}} \wedge (\ell)$ is satisfiable, so ℓ cannot be a failed literal. There are three cases to consider.

1. If ℓ is a negative literal, assigning all variables to 0 satisfies $F_{\mathrm{fl}} \wedge (\ell)$, as all clauses are non-unit Horn clauses.
2. If ℓ is a positive literal and ℓ is not of form y_p, then assigning variables in the set $\{\ell, w\} \cup \{y_q\colon q \in Q\}$ to 1 and other variables to 0 satisfies $F_{\mathrm{fl}} \wedge (\ell)$.
3. If $\ell = y_p$ for some $p \in P$ and pq does not satisfy F for any $q \in Q$, then by Lemma 3(b) assigning y_p and all variables c_i such that $p(C_i) = 1$ to 1 and other variables to 0 satisfies $F_{\mathrm{fl}} \wedge (y_p)$.

$\square$

The formula F_{fl} is still not necessarily 3-CNF. However, standard rewriting of a long clause as clauses of length 3 preserves the Horn property: rewriting a Horn clause $(x_1 \wedge \cdots \wedge x_{k-1}) \rightarrow l)$, where l is either x_k or $\neg x_k$, using fresh variables $a_3, \ldots, a_{k-1}$ gives the Horn clauses

$$\big((x_1 \wedge x_2) \rightarrow a_3\big) \wedge \big((a_3 \wedge x_3) \rightarrow a_4\big) \wedge \cdots \wedge \big((a_{k-1} \wedge x_{k-1}) \rightarrow l\big)\,.$$

This rewriting preserves unit propagations over the original clause, and hence does not affect the existence of failed literals.

Theorem 1 follows now from Lemmas 1–4 and the rewriting from Horn-CNF to Horn-3-CNF.

4 Extensions

The reduction template described in Section 3.1 can be used to prove similar lower bounds for existence problems over other notions related to failed literals. Here we consider the existence problem for asymmetric tautologies and literals, and checking singleton arc consistency of binary CSPs of domain size three.

For a CNF formula F, a clause $C = (l_1 \vee \cdots \vee l_k) \in F$ is an *asymmetric tautology* [19] if unit propagation derives a conflict on $(F \setminus C) \wedge (\neg l_1) \wedge \cdots \wedge (\neg l_k)$. If C is an asymmetric tautology, then F can be simplified by letting $F := F \setminus C$. A clause $C \in F$, a literal ℓ is an *asymmetric literal in* C if unit propagation on $F \vee (\ell)$ derives (ℓ') for some $\ell' \in C \setminus \{\ell\}$. If ℓ is an asymmetric tautology in C, then F can be simplified by letting $F := (F \setminus C) \wedge (C \setminus l)$. The notion of asymmetric literals is a generalization of hidden literals [20]. Both asymmetric tautologies and asymmetric literals are detected e.g. by vivification and during inprocessing within the Lingeling SAT solver [4].

Furthermore, we consider the problem of *checking singleton arc consistency* [10, 36] of constraint satisfaction problems (CSPs) with constraints over pairs of variable (i.e., binary CSPs) and with variable domains restricted to cardinality three. On CNF formulas, checking singleton arc consistency is equivalent to checking for the existence of failed literals, and extends to naturally to CSPs[1]. Here we consider the *restricted setting* of $(3, 2)$-CSPs, i.e., binary CSPs with variable domains restricted to cardinality three.

Theorem 2. *Let $\varepsilon > 0$. There is a $2^{(1-\varepsilon/2)n} \operatorname{poly}(n, m)$ time algorithm for CNF-SAT on formulas with n variables and m clauses if one of the following holds.*

(a) *There is a $O\big((N + M)^{2-\varepsilon}\big)$ algorithm for asymmetric tautology existence over Horn-3-CNF formulas with N variables and M clauses.*
(b) *There is a $O\big((N + M)^{2-\varepsilon}\big)$ algorithm for asymmetric literal existence over Horn-3-CNF formulas with N variables and M clauses.*
(c) *There is a $O\big((N + M)^{2-\varepsilon}\big)$ algorithm for checking singleton arc consistency over $(3, 2)$-CSPs with N variables and M constraints.*

Proof sketch. For (a) and (b), we start from the reduction template given in Section 3.1 and extend it in a similar manner as in Section 3.2 with following differences. For (a), we construct formula F_{at} by adding the clause $(y_p \to w)$ to F' for each $p \in P$. This new clause is an asymmetric tautology if and only if p can be extended to a satisfying assignment for F, and there are no other asymmetric tautologies in F_{at} (compare with Lemma 4).

[1] Enforcing arc consistency refers to reducing the variable domains of a CSP until fixpoint using the following rule: if there is a variable x and a value v in the domain $D(x)$ of x that is not supported by a constraint in the CSP, then let $D(x) := D(x) \setminus \{v\}$. A CSP is singleton arc consistent if for all x and $v \in D(x)$, enforcing singleton arc consistency on the CSP after assigning $D(x) := \{v\}$ does not result in an empty domain for some variable.

For (b), we construct the formula F_{al} by adding the clause $(w \vee y_p)$ to F' for each $p \in P$; again, only possible asymmetric literals arise from these clauses and correspond to satisfying assignments of F as before. Replacing w with $\neg w$ in all clauses gives a Horn formula.

For (c), we transform the Horn-3-CNF formula F_{fl} obtained from the failed literal reduction into $(3,2)$-CSP instance, by emulating each length 3 Horn clause with two binary constraints over domain of size 3. That is, we replace each clause $\big((a \wedge b) \rightarrow c\big)$ by two constraints C and C' as follows. For C, we set $\mathrm{var}(C) = \{a, c\}$ and allow all assignments with $a = 0$ or $a = 2$, and assignments $a = 1, c = 0$ and $a = 1, c = 2$. For C', we set $\mathrm{var}(C') = \{b, c\}$ and allow all assignments with $b = 0$ or $b = 2$, and assignments $b = 1, c = 0$ and $b = 1, c = 1$. The resulting CSP instance is singleton arc consistent if and only if F_{fl} has no failed literals. $\square$

5 Concluding Remarks

We established a connection between the strong exponential time hypothesis and the existence of subquadratic algorithms for checking whether a given CNF formula contains at least one failed literal, as well as several other related simplification techniques. Any improvement over the obvious algorithm for failed literal existence, even on Horn-3-CNF formulas, would require a major algorithmic breakthrough.

However, several questions related to our results remain open. So far, we were unable to establish a similar lower bound for failed literal existence for 2-CNF formulas, and as such cannot rule out the possibility of a subquadratic algorithm for the 2-CNF case. In contrast, our proofs require clauses of width 3, or domain size 3 in the case of binary CSPs. Similarly, it is open whether there exist $O\big((n+m)^{3-\epsilon}\big)$ time algorithms for computing the *fixpoint* of failed literal elimination. Proving a conditional lower bound for the fixpoint computation would be a stronger result than our lower bound for failed literal existence. Again, it is not clear whether such a lower bound can be established, especially in the limited setting of Horn-CNFs; note that for 2-CNFs, computing the fixpoint can be done in time $O\big(n(n+m)\big)$, as all failed literals in the fixpoint fail without eliminating other failed literals first. Another possible extension of our results would be to prove a more general conditional lower bound for *k-step lookahead*, i.e., the extension of failed literal existence / failed literal elimination (i.e., 1-step lookahead) to testing sets of $k > 1$ literals instead of a single literal at a time [16].

Finally, a converse result—that is, showing that faster CNF-SAT algorithms would imply faster algorithms for failed literal existence—would give an even tighter connection between the complexity of the two problems.

References

1. Bacchus, F.: Enhancing Davis Putnam with extended binary clause reasoning. In: Proc. AAAI. pp. 613–619. AAAI Press / The MIT Press (2002)

2. Bacchus, F., Winter, J.: Effective preprocessing with hyper-resolution and equality reduction. In: Proc. SAT 2003. LNCS, vol. 2919, pp. 341–355. Springer (2004)
3. Berre, D.L.: Exploiting the real power of unit propagation lookahead. Electronic Notes in Discrete Mathematics 9, 59–80 (2001)
4. Biere, A.: Lingeling, Plingeling and Treengeling entering the SAT Competition 2013. In: Proceedings of SAT Competition 2013. Department of Computer Science Series of Publications B, vol. B-2013-1, pp. 51–52. University of Helsinki (2013)
5. Biere, A., Heule, M., van Maaren, H., Walsh, T. (eds.): Handbook of Satisfiability, Frontiers in Artificial Intelligence and Applications, vol. 185. IOS Press (2009)
6. Brafman, R.I.: A simplifier for propositional formulas with many binary clauses. IEEE Transactions on Systems, Man, and Cybernetics, Part B 34(1), 52–59 (2004)
7. Calabro, C., Impagliazzo, R., Kabanets, V., Paturi, R.: The complexity of unique k-SAT: An isolation lemma for k-CNFs. Journal of Computer and System Sciences 74(3), 386–393 (2008)
8. Calabro, C., Impagliazzo, R., Paturi, R.: The complexity of satisfiability of small depth circuits. In: IWPEC 2009 Revised Selected Papers, LNCS, vol. 5917, pp. 75–85. Springer (2009)
9. Cygan, M., Dell, H., Lokshtanov, D., Marx, D., Nederlof, J., Okamoto, Y., Paturi, R., Saurabh, S., Wahlstrom, M.: On problems as hard as CNF-SAT. In: Proc. CCC. pp. 74–84. IEEE (2012)
10. Debruyne, R., Bessière, C.: Some practicable filtering techniques for the constraint satisfaction problem. In: Proc. IJCAI. pp. 412–417. Morgan Kaufmann (1997)
11. Eén, N., Biere, A.: Effective preprocessing in SAT through variable and clause elimination. In: Proc. SAT. LNCS, vol. 3569, pp. 61–75. Springer (2005)
12. Fourdrinoy, O., Grégoire, É., Mazure, B., Sais, L.: Reducing hard SAT instances to polynomial ones. In: Proc. IRI. pp. 18–23. IEEE Systems, Man, and Cybernetics Society (2007)
13. Freeman, J.: Improvements to propositional satisfiability search algorithms. PhD thesis, University of Pennsylvania, USA (1995)
14. Gelder, A.V.: Toward leaner binary-clause reasoning in a satisfiability solver. Ann. Math. Artif. Intell. 43(1), 239–253 (2005)
15. Gershman, R., Strichman, O.: Cost-effective hyper-resolution for preprocessing CNF formulas. In: Proc. SAT. LNCS, vol. 3569, pp. 423–429. Springer (2005)
16. Gwynne, M., Kullmann, O.: Generalising unit-refutation completeness and SLUR via nested input resolution. J. Autom. Reasoning 52(1), 31–65 (2014)
17. Han, H., Somenzi, F.: Alembic: An efficient algorithm for CNF preprocessing. In: Proc. DAC. pp. 582–587. IEEE (2007)
18. Heule, M., Dufour, M., van Zwieten, J., van Maaren, H.: March_eq: Implementing additional reasoning into an efficient look-ahead SAT solver. In: SAT 2004 Selected Papers. LNCS, vol. 3542, pp. 345–359. Springer (2005)
19. Heule, M., Järvisalo, M., Biere, A.: Clause elimination procedures for CNF formulas. In: Proc. LPAR-17. LNCS, vol. 6397, pp. 357–371. Springer (2010)
20. 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)
21. Heule, M., Järvisalo, M., Biere, A.: Revisiting hyper binary resolution. In: Proc. CPAIOR. LNCS, vol. 7874, pp. 77–93. Springer (2013)
22. Heule, M., van Maaren, H.: Aligning CNF- and equivalence-reasoning. In: SAT 2004 Selected Papers. LNCS, vol. 3542, pp. 145–156. Springer (2005)
23. Heule, M., van Maaren, H.: Look-ahead based SAT solvers. In: Biere et al. [5], pp. 155–184

24. Impagliazzo, R., Paturi, R., Zane, F.: Which problems have strongly exponential complexity? Journal of Computer and System Sciences 63(4), 512–530 (2001)
25. Järvisalo, M., Biere, A., Heule, M.: Simulating circuit-level simplifications on CNF. J. Autom. Reasoning 49(4), 583–619 (2012)
26. Järvisalo, M., Heule, M., Biere, A.: Inprocessing rules. In: Proc. IJCAR. LNCS, vol. 7364, pp. 355–370. Springer (2012)
27. Kullmann, O.: Investigating the behaviour of a SAT solver on random formulas (2002)
28. Kullmann, O.: Fundaments of branching heuristics. In: Biere et al. [5], pp. 205–244
29. Li, C.M.: Equivalency reasoning to solve a class of hard SAT problems. Inf. Process. Lett. 76(1-2), 75–81 (2000)
30. Li, C.M.: Integrating equivalency reasoning into Davis-Putnam procedure. In: Proc. AAAI. pp. 291–296. AAAI Press / The MIT Press (2000)
31. Li, C.M., Anbulagan: Heuristics based on unit propagation for satisfiability problems. In: Proc. IJCAI. pp. 366–371. Morgan Kaufmann (1997)
32. Lynce, I., Marques-Silva, J.P.: Probing-based preprocessing techniques for propositional satisfiability. In: Proc. ICTAI. pp. 105–. IEEE Computer Society (2003)
33. Manthey, N., Heule, M., Biere, A.: Automated reencoding of boolean formulas. In: Proc. HVC 2012. LNCS, vol. 7857, pp. 102–117. Springer (2013)
34. Niemelä, I., Simons, P.: Smodels - an implementation of the stable model and well-founded semantics for normal LP. In: Proc. LPNMR. LNCS, vol. 1265, pp. 421–430. Springer (1997)
35. Piette, C., Hamadi, Y., Sais, L.: Vivifying propositional clausal formulae. In: Proc. ECAI. FAIA, vol. 178, pp. 525–529. IOS Press (2008)
36. Prosser, P., Stergiou, K., Walsh, T.: Singleton consistencies. In: Proc. CP. LNCS, vol. 1894, pp. 353–368. Springer (2000)
37. Pătraşcu, M., Williams, R.: On the possibility of faster SAT algorithms. In: Proc. SODA. pp. 1065–1075 (2010)
38. Sheeran, M., Stålmarck, G.: A tutorial on Stålmarck's proof procedure for propositional logic. Formal Methods in System Design 16(1), 23–58 (2000)