On CDCL-Based Proof Systems with the Ordered Decision Strategy

Nathan Mull[1(✉)], Shuo Pang[2], and Alexander Razborov[1,2,3]

[1] Department of Computer Science, University of Chicago, Chicago, USA
{nmull,razborov}@cs.uchicago.edu
[2] Department of Mathematics, University of Chicago, Chicago, USA
{spang,razborov}@math.uchicago.edu
[3] Steklov Mathematical Institute, Moscow, Russia

Abstract. We prove that CDCL SAT-solvers with the ordered decision strategy and the DECISION learning scheme are equivalent to ordered resolution. We also prove that, by replacing this learning scheme with its opposite, which learns the first possible non-conflict clause, they become equivalent to general resolution. In both results, we allow nondeterminism in the solver's ability to perform unit propagation, conflict analysis, and restarts in a way that is similar to previous works in the literature. To aid the presentation of our results, and possibly future research, we define a model and language for CDCL-based proof systems – particularly those with nonstandard features – that allow for succinct and precise theorem statements.

1 Introduction

Since their conception, SAT-solvers have become significantly more efficient, but they have also become significantly more complex. Consequently, there has been increasing interest in understanding their theoretical limitations and strengths. Much of the recent literature has focused on the relationship between CDCL SAT-solvers[1] and the resolution proof system. Beame et al. [5] were the first to study this relationship and many followed suit (see [3,6,7,11,13–15,17–20] among others). In particular, Pipatsrisawat and Darwiche [18] show that, under a few assumptions, CDCL with the nondeterministic decision strategy (i.e., when the solver has to choose a variable to assign, it chooses both the variable and its assigned value nondeterministically) polynomially simulates resolution. An obvious question arises from this result: how much does the theoretical efficiency of CDCL depend on nondeterminism in the decision strategy? Along these lines, Atserias et al. [3] (concurrently with [18]) show that CDCL with the *random* decision strategy (i.e., both the variable and assigned value are chosen uniformly at random) simulates bounded-width resolution, under essentially

[1] In this paper, we focus solely on solvers which implement conflict-driven clause learning (CDCL). We refer to such solvers as CDCL SAT-solvers, CDCL solvers, or just CDCL for short.

L. Pulina and M. Seidl (Eds.): SAT 2020, LNCS 12178, pp. 149–165, 2020.
https://doi.org/10.1007/978-3-030-51825-7_12

the same assumptions as those in [18]. More recently, Vinyals [20] has shown that CDCL with the VSIDS decision strategy – among other common dynamic decision strategies – does not simulate general resolution. We attempt to make progress on this question by studying a simple decision strategy that we call the *ordered* decision strategy. This strategy is identical to the one studied by Beame et al. [4] in the context of DPLL without clause learning. It is defined naturally: when the solver has to choose a variable to assign, it chooses the smallest unassigned variable according to some fixed order and chooses its assigned value nondeterministically. If unit propagation is used, the solver may assign variables out of order; a unit clause does not necessarily correspond to the smallest unassigned variable. This possibility of "cutting the line" is precisely what makes the situation more subtle and nontrivial. Thus, our motivating question is the following:

> *Is there a family of contradictory CNFs* $\{\tau_n\}_{n=1}^{\infty}$ *that possess polynomial size resolution refutations but require superpolynomial time for CDCL using the ordered decision strategy?*

We also note in passing that this question may be motivated as a way of understanding the strength of *static* decision strategies such as MINCE [1] and FORCE [2].

Our Contributions. A proof system that captures any class of CDCL solvers should be no stronger than general resolution, and if it captures solvers with the ordered decision strategy, it should be reasonably expected to be at least as strong as ordered resolution with respect to the same order. Our main results show that, depending on the learning scheme employed, both of these extremes are attained. More specifically, we prove

1. CDCL with the ordered decision strategy and a learning scheme we call DECISION-L is equivalent to ordered resolution (Theorem 1). In particular, it does not simulate general resolution.
2. CDCL with the ordered decision strategy and a learning scheme we call FIRST-L is equivalent to general resolution (Theorem 2).

Remark 1. As the name suggests, DECISION-L is the same as the so-called DECISION learning scheme.[2] FIRST-L is a learning scheme designed to directly simulate particular resolution steps in the presence of certain forms of nondeterminism, and is similar to FirstNewCut [5]. In the full version of this paper [16], we also prove linear width lower bounds which, combined with the second result, create a sharp contrast with the size-width relationship for general resolution proved by Ben-Sasson and Wigderson [8].

In these two results, the CDCL solver may arbitrarily choose the conflict/unit clause if there are several, may elect not to do conflict analysis/unit propagations

[2] The name DECISION-L better fits the naming conventions of our model.

at all, and may restart at any time. This substantial amount of nondeterminism allows us to identify two proof systems that are, more or less straightforwardly, equivalent to the corresponding CDCL variant. Determining the exact power of these systems constitutes our main technical contribution.

There are a couple points of interpretation to emphasize here. First, the implicit separation between CDCL solvers and general resolution in the first result applies to *actual* SAT-solver implementations, albeit with heuristics that are not usually used in practice, and could, in principle, be demonstrated by experiment. In contrast, the second result does not say anything substantial about actual SAT-solver implementations. But we also note that this is not unprecedented. The correspondence between proof systems and algorithms considered here is very similar to the correspondence between *regWRTI* and a variant of CDCL with similar features called DLL-LEARN, both introduced by Buss et al. [11]; nonstandard sources of nondeterminism manifest themselves naturally when translating CDCL into a proof system. Both lower and upper bounds on these systems are valuable; even if upper bounds do not apply directly to practice, they demonstrate, often nontrivially, what convenient features of simple proof systems must be dropped to potentially prove separations.

Finally, in order to aid the above work – and, perhaps, even facilitate further research in the area – we present a model and language for studying CDCL-based proof systems. This model is not meant to be novel, and is heavily influenced by previous work [3,13,17]. However, the primary goal of our model is to *highlight* possible nonstandard sources of nondeterminism in variants of CDCL, as opposed to creating a model completely faithful to applications. Our second result (Theorem 2) can be written in this language as:

> *For any order* π, $\mathsf{CDCL}(\mathsf{FIRST\text{-}L}, \pi\text{-}\mathsf{D})$ *is polynomially equivalent to general resolution.*

Due to space limitations, not all proofs are provided and there may be excluded details or remarks that, though not essential, are useful in understanding possible subtleties in the constructions and arguments. After presenting the preliminary material in Sect. 2, we give an nearly complete account of our first result mentioned above in Sect. 3, and reflect very briefly on our second result in Sect. 4. We refer the reader to the full version of this paper [16] for complete proofs and extended discussion.

2 Preliminaries

Throughout the paper, we assume that the set of propositional variables is fixed as $V \stackrel{\text{def}}{=} \{x_1, \ldots, x_n\}$. A *literal* is either a propositional variable or its negation. We will sometimes use the abbreviation x^0 for $\bar{x}$ and x^1 for x (so that the Boolean assignment $x = a$ *satisfies* the literal x^a). A *clause* is a set of literals, thought of as their disjunction, in which no variable appears together with its negation. For a clause C, let $\mathrm{Var}(C)$ denote the set of variables appearing in C. A *CNF* is a set of clauses thought of as their conjunction. For a CNF τ, let $\mathrm{Var}(\tau)$ denote

the set of variables appearing in τ, i.e., the union of $\mathrm{Var}(C)$ for all $C \in \tau$. We denote the empty clause by 0. The *width* of a clause is the number of literals in it.

The *resolution proof system* is a Hilbert-style proof system whose lines are clauses and that has only one *resolution rule*

$$\frac{C \vee x_i^a \qquad D \vee x_i^{1-a}}{C \vee D}, \ a \in \{0,1\}. \tag{1}$$

We will sometimes denote the result of resolving $C \vee x_i^a$ and $D \vee x_i^{1-a}$ by $\mathrm{Res}(C \vee x_i^a, D \vee x_i^{1-a})$.

The *size* of a resolution proof Π, denoted as $|\Pi|$, is the number of lines in it. For a CNF τ and a clause C, let $S_R(\tau \vdash C)$ denote the minimal possible size of a resolution proof of the clause C from clauses in τ (∞ if C is not implied by τ). Likewise, let $w(\tau \vdash C)$ denote the minimal possible width of such a proof, defined as the maximal width of a clause in it. For a proof Π that derives C from τ, the clauses in τ that appear in Π are called *axioms*, and if $C = 0$ then Π is called a *refutation*. Let $\mathrm{Var}(\Pi)$ denote the set of variables appearing in Π, i.e., the union of $\mathrm{Var}(C)$ for C appearing in Π.

Note that the *weakening rule*

$$\frac{C}{C \vee D}$$

is *not* included by default. In the full system of resolution it is admissible in the sense that $S_R(\tau \vdash 0)$ does not change if we allow it. But this will not be the case for some of the CDCL-based fragments we will be considering below. Despite this, it is often convenient in analysis to consider intermediate systems that do allow the weakening rule. We make it clear when we do this by adding the annotation '+ weakening' to the system.

Resolution Graphs. Our results depend on the careful analysis of the structure of resolution proofs. It will, for example, be useful for us to maintain structural properties of the proof while changing the underlying clauses and derivations.

Definition 1. *For a resolution + weakening proof Π, its* ***resolution graph****, $G(\Pi)$, is an acyclic directed graph representing Π in the natural way: each clause in Π has a distinguished node, and for each node there are incoming edges from the nodes corresponding to the clauses from which it is derived. The set of nodes of $G(\Pi)$ is denoted by $V(\Pi)$, and the clause at $v \in V(\Pi)$ is denoted by $c_\Pi(v)$.*[3]

In the following collection of definitions, let Π be an arbitrary resolution + weakening proof and let S be an arbitrary subset of $V(\Pi)$. A vertex u is *above* a vertex v in $G(\Pi)$, written $u > v$, if there is a directed path from v to u. We also say v is *below* u. Moreover, v is a *parent* of u if (v, u) is an edge in

[3] We do *not* assume that c_Π is injective; we allow the same clause to appear in the proof several times.

$G(\Pi)$. S is *independent* if any two of its nodes are incomparable. The *maximal* and *minimal* nodes of S are $\max_\Pi S \stackrel{\text{def}}{=} \{v \in S \mid \forall u \in S\ (\neg(v < u))\}$ and $\min_\Pi S \stackrel{\text{def}}{=} \{v \in S \mid \forall u \in S\ (\neg(v > u))\}$, respectively. The *upward closure* and *downward closure* of S in $G(\Pi)$ are $\mathrm{ucl}_\Pi(S) \stackrel{\text{def}}{=} \{v \in V(\Pi) \mid \exists w \in S\ (v \geq w)\}$ and $\mathrm{dcl}_\Pi(S) \stackrel{\text{def}}{=} \{v \in V(\Pi) \mid \exists w \in S\ (v \leq w)\}$, respectively. S is *parent-complete* if it contains either both parents or neither parent of each of its nodes. S is *path-complete* if it contains all nodes along any path in $G(\Pi)$ whose endpoints are in S. A resolution graph is *connected* if $|\max_\Pi V(\Pi)| = 1$, i.e., it has a unique sink. These definitions behave naturally, as demonstrated by the following useful proposition, which is easily verified.

Proposition 1. *Let $S \subseteq V(\Pi)$ be a nonempty set of nodes that is both parent-complete and path-complete. Then the induced subgraph of $G(\Pi)$ on S is the graph of a subproof in Π of $\max_\Pi S$ from $\min_\Pi S$.*

Ordered Resolution. Fix now an order $\pi \in S_n$. For any literal x_k^a, define $\pi(x_k^a) \stackrel{\text{def}}{=} \pi(k)$. For $k \in [n]$, let Var_π^k denote the k smallest variables according to π. A clause C is *k-small* with respect to π if $\mathrm{Var}(C) \subseteq \mathrm{Var}_\pi^k$.

The proof system *π-ordered resolution* is the subsystem of resolution defined by imposing the following restriction on the resolution rule (1):

$$\forall l \in C \vee D\ (\pi(l) < \pi(x_i)).$$

In the literature this system is usually defined differently, namely in a top-down manner (see, e.g., [10]). It is easy to see, however, that our version is equivalent.

CDCL-Based Proof Systems. Our approach to modeling CDCL is, in a sense, the opposite of what currently exists in the literature. Rather than attempting to model CDCL solver implementations as closely as possible and allowing nondeterminism in various features, we rigorously describe a *basic* model that is very liberal and nondeterministic and intends to approximate the union of most conceivable features of CDCL solvers. Then models of actual interest will be defined by their *deviations* from the basic model. Due to space limitations, we present our model rather tersely (see the full version of this paper [16] for further details).

A few more definitions are in order before proceeding. A *unit clause* is a clause consisting of a single literal. An *assignment* is an expression of the form $x_i = a$ where $1 \leq i \leq n$ and $a \in \{0, 1\}$. A *restriction* ρ is a set of assignments in which all variables are pairwise distinct. Let $\mathrm{Var}(\rho)$ denote the set of all variables appearing in ρ. Restrictions naturally act on clauses, CNFs, and resolution proofs; we denote the result of this action by $C|_\rho$, $\tau|_\rho$, and $\Pi|_\rho$, respectively. An *annotated assignment* is an expression of the form $x_i \stackrel{*}{=} a$ where $1 \leq i \leq n$, $a \in \{0, 1\}$, and $* \in \{d, u\}$. See Definition 3 below for details about these annotations.

The underlying structure of our model is a labeled transition system whose states represent data maintained by a CDCL solver during runtime and whose

labeled transitions are possible actions taken by a solver during runtime. We first define explicitly what constitutes a state.

Definition 2. *A **trail** is an ordered list of annotated assignments in which all variables are pairwise distinct. A trail acts on clauses, CNFs, and proofs just in the same way as does the restriction obtained from it by disregarding the order and the annotations on assignments. For a trail t and an annotated assignment $x_i \stackrel{*}{=} a$ such that x_i does not appear in t, we denote by $[t, x_i \stackrel{*}{=} a]$ the trail obtained by appending $x_i \stackrel{*}{=} a$ to its end. $t[k]$ is the kth assignment of t. A **prefix** of a trail $t = [x_{i_1} \stackrel{*_1}{=} a_1, \ldots, x_{i_r} \stackrel{*_r}{=} a_r]$ is any trail of the form $[x_{i_1} \stackrel{*_1}{=} a_1, \ldots, x_{i_s} \stackrel{*_s}{=} a_s]$ where $0 \leq s \leq r$ and is denoted by $t[\leq s]$. Λ is the empty trail.*

*A **state** is a pair (τ, t), where τ is a CNF and t is a trail. The state (τ, t) is **terminal** if either $C|_t \equiv 1$ for all $C \in \tau$ or τ contains 0. All other states are nonterminal. We let $\mathbb{S}_n$ denote the set of all states (recall that n is reserved for the number of variables), and let $\mathbb{S}_n^o \subseteq \mathbb{S}_n$ be the set of all nonterminal states.*

We now describe the core of our (or, for that matter, any other) model, that is, transition rules between states.

Definition 3. *For a (nonterminal) state $S = (\tau, t) \in \mathbb{S}_n^o$, we define the finite set $Actions(S)$ and the function $Transitions_S : Actions(S) \longrightarrow \mathbb{S}_n$; the fact $Transitions_S(A) = S'$ will be usually abbreviated to $S \stackrel{A}{\Longrightarrow} S'$. Those are described as follows:*

$$Actions(S) \stackrel{\text{def}}{=} D(S) \dot\cup U(S) \dot\cup L(S),$$

where the letters D, U, L have the obvious meaning[4]*.*

- *$D(S)$ consists of all annotated assignments $x_i \stackrel{d}{=} a$ such that x_i does not appear in t and $a \in \{0, 1\}$. We naturally let*

$$(\tau, t) \stackrel{x_i \stackrel{d}{=} a}{\Longrightarrow} (\tau, [t, x_i \stackrel{d}{=} a]). \tag{2}$$

- *$U(S)$ consists of all those assignments $x_i \stackrel{u}{=} a$ for which $\tau|_t$ contains the unit clause x_i^a; the transition function is given by the same formula (2) but with a different annotation:*

$$(\tau, t) \stackrel{x_i \stackrel{u}{=} a}{\Longrightarrow} (\tau, [t, x_i \stackrel{u}{=} a]). \tag{3}$$

- *As should be expected, $L(S)$ is the most sophisticated part of the definition (cf.* [3, Section 2.3.3]*). Let $t = [x_{i_1} \stackrel{*_1}{=} a_1, \ldots, x_{i_r} \stackrel{*_r}{=} a_r]$. By reverse induction on $k = r + 1, \ldots, 1$ we define the set $\mathbb{C}_k(S)$ that, intuitively, is the set of clauses that can be learned by backtracking up to the prefix $t[\leq k]$. We let*

$$\mathbb{C}_{r+1}(S) \stackrel{\text{def}}{=} \{\, D \in \tau \mid D|_t = 0 \,\}$$

[4] Restarts will be treated as a part of the learning scheme.

be the set of all ***conflict clauses****.*

For $1 \le k \le r$, we do the following: if the kth assignment of t is of the form $x_{i_k} \stackrel{d}{=} a_k$, then $\mathbb{C}_k(S) \stackrel{\text{def}}{=} \mathbb{C}_{k+1}(S)$. Otherwise, it is of the form $x_{i_k} \stackrel{u}{=} a_k$, and we build up $\mathbb{C}_k(S)$ by processing every clause $D \in \mathbb{C}_{k+1}(S)$ as follows.

- *If D does not contain the literal $\overline{x_{i_k}^{a_k}}$ then we include D into $\mathbb{C}_k(S)$ unchanged.*
- *If D contains $\overline{x_{i_k}^{a_k}}$, then we resolve D with all clauses $C \in \tau$ such that $C|_{t[\le k-1]} = x_{i_k}^{a_k}$ and include all the results in $\mathbb{C}_k(S)$. The clause D itself is not included.*

To make sure that this definition is sound, we have to guarantee that C and D are actually resolvable (that is, they do not contain any other conflicting variables but x_{i_k}). For that we need the following observation, easily proved by reverse induction on k, simultaneously with the definition:

Claim. $D|_t = 0$ *for every* $D \in \mathbb{C}_k(S)$.

Finally, we let

$$\mathbb{C}(S) \stackrel{\text{def}}{=} \bigcup_{k=1}^{r} \mathbb{C}_k(S),$$

$$L(S) \stackrel{\text{def}}{=} \begin{cases} \{(0, \Lambda)\} & 0 \in \mathbb{C}(S) \\ \{(C, t^*) \mid C \in (\mathbb{C}(S) \setminus \tau) \text{ and} \\ \quad t^* \text{ is a prefix of } t \text{ such that } C|_{t^*} \neq 0\} & \text{otherwise} \end{cases} \tag{4}$$

and

$$(\tau, t) \stackrel{(C,t^*)}{\Longrightarrow} (\tau \cup \{C\}, t^*).$$

This completes the description of the basic model.

The *transition graph* Γ_n is the directed graph on $\mathbb{S}_n$ defined by erasing the information about actions; thus $(S, S') \in E(\Gamma_n)$ if and only if $S' \in \mathrm{im}(\mathrm{Transition}_S)$. It is easy to see (by double induction on $(|\tau|, n - |t|)$) that Γ_n is acyclic. Moreover, both the set $\{(S, A) \mid A \in \mathrm{Actions}(S)\}$ and the function $(S, A) \mapsto \mathrm{Transition}_S(A)$ are polynomial-time[5] computable. These observations motivate the following definition.

Definition 4. *Given a CNF τ, a* ***partial run*** *on τ from the state S to the state T is a sequence*

$$S = S_0 \stackrel{A_0}{\Longrightarrow} S_1 \stackrel{A_1}{\Longrightarrow} \dots S_{L-1} \stackrel{A_{L-1}}{\Longrightarrow} S_L = T, \tag{5}$$

where $A_k \in \mathit{Actions}(S_k)$. In other words, a partial run is a labeled path in Γ_n. A ***successful run*** *is a partial run from (τ, Λ) to a terminal state. A* ***CDCL***

[5] That is, polynomial in the size of the state S, not in n.

solver *is a partial function*[6] μ *on* $\mathbb{S}_n^o$ *such that* $\mu(S) \in Actions(S)$ *whenever* $\mu(S)$ *is defined. The above remarks imply that when we repeatedly apply a CDCL solver* μ *starting at any initial state* (τ, Λ)*, it will always result in a finite sequence like* (5)*, with* T *being a terminal state (successful run) or such that* $\mu(T)$ *is undefined (failure).*

Theoretical analysis usually deals with *classes* (i.e., sets) of individual solvers rather than with individual implementations. We define such classes by prioritizing and restricting various actions.

Definition 5. *A* ***local*** *class of CDCL solvers is described by a collection of subsets* $AllowedActions(S) \subseteq Actions(S)$ *where* $S \in \mathbb{S}_n^o$*. It consists of all those solvers* μ *for which* $\mu(S) \in AllowedActions(S)$*, whenever* $\mu(S)$ *is defined.*

We will describe local classes of solvers in terms of *amendments* prescribing what actions should be *removed* from the set Actions(S) to form AllowedActions(S). The examples presented below illustrate how familiar restrictions look in this language. Throughout their description, we fix a nonterminal state $S = (\tau, t)$.

ALWAYS-C If $\tau|_t$ contains the empty clause, then $D(S)$ and $U(S)$ are removed from Actions(S). In other words, this amendment requires the solver to perform conflict analysis if it can do so.

ALWAYS-U If $\tau|_t$ contains a unit clause, then $D(S)$ is removed from Actions(S). This amendment insists on unit propagation, but leaves to nondeterminism the choice of the unit to propagate if there are several choices. Note that as defined, ALWAYS-U is a lower priority amendment than ALWAYS-C: if both a conflict and a unit clause are present, the solver must do conflict analysis.

DECISION-L In the definition (4), we shrink $\mathbb{C}(S) \setminus \tau$ to $\mathbb{C}_1(S) \setminus \tau$.

FIRST-L In the definition (4), we shrink $\mathbb{C}(S) \setminus \tau$ to those clauses that are obtained by resolving a conflict clause with one other clause in τ. Such clauses are the first learnable clauses encountered in the process from Definition 3.

π-D, **where** $\pi \in S_n$ **is an order on the variables** We keep in $D(S)$ only the two assignments $x_i \stackrel{d}{=} 0$, $x_i \stackrel{d}{=} 1$, where x_i is the *smallest* variable with respect to π that does not appear in t. Note that this amendment does not have any effect on $U(S)$, and our main technical contributions can be phrased as answering under which circumstances this "loophole" can circumvent the severe restriction placed on the set $D(S)$.

NEVER-R In the definition (4), we require that t^* is the *longest* prefix of t satisfying $C|_{t^*} \neq 0$ (in which case $C|_{t^*}$ is necessarily a unit clause). As described, this amendment does not model nonchronological backtracking or require that the last assignment in the trail is a decision. However, this version is easier to state and it is not difficult to modify it to have the aforementioned properties.

WIDTH-w, **where** w **is an integer** In the definition (4), we keep in $\mathbb{C}(S) \setminus \mathbb{C}$ only clauses of width $\leq w$. Note that this amendment still allows us to use wide clauses as intermediate results *within* a single clauses learning step.

[6] It is possible for Actions(S) to be empty.

Thus, our preferred way to specify local classes of solvers and the corresponding proof systems is by listing one or more amendments, with the convention that their effect is cumulative: an action is removed from Actions(S) if and only if it should be removed according to at least one of the amendments present. More formally,

Definition 6. *For a finite set $\mathcal{A}_1, \ldots, \mathcal{A}_r$ of polynomial-time computable*[7] *amendments, we let* CDCL($\mathcal{A}_1, \ldots, \mathcal{A}_r$) *be the (possibly incomplete) proof system whose proofs are those successful runs in which none of the actions A_i is affected by any of the amendments $\mathcal{A}_1, \ldots, \mathcal{A}_r$.*

Using this language, the main result from [18] can be very roughly summarized as

> CDCL(DECISION-L, ALWAYS-C, ALWAYS-U) *is polynomially equivalent to general resolution.*

The open question asked in [3, Section 2.3.4] can be reasonably interpreted as whether CDCL(ALWAYS-C, ALWAYS-U, WIDTH-w) is as powerful as width-w resolution, perhaps with some gap between the two width constraints. Our width lower bound mentioned in the introduction can be cast in this language as

> *For any fixed order π on the variables and every $\epsilon > 0$ there exist contradictory CNFs τ_n with $w(\tau_n \vdash 0) \leq O(1)$ not provable in* CDCL(π-D, WIDTH-$(1-\epsilon)n$).

Finally, we would like to mention the currently open question about the exact strength of CDCL without restarts. This is one of the most interesting open questions in the area and has been considered heavily in the literature (see [6,9,11,12,15,19] among others). It may be abstracted as

> *Does* CDCL(ALWAYS-C, ALWAYS-U, NEVER-R) (*or at least* CDCL(NEVER-R)) *simulate general resolution?*

For both open questions mentioned above, we have taken the liberty of removing those amendments that do not appear immediately relevant.

At this point, since we discuss our main results in the introduction, we formulate them here more or less matter-of-factly.

Theorem 1. *For any fixed order π on the variables, the system* CDCL(DECISION-L, π-D) *is polynomially equivalent to π-ordered resolution.*

Theorem 2. *For any fixed order π on the variables, the system* CDCL(FIRST-L, π-D) *is polynomially equivalent to general resolution.*

[7] An amendment is polynomial-time computable if determining whether an action in *Action*(S) is allowed by the amendment is polynomial-time checkable, given the state S.

3 CDCL(DECISION-L, π-D) $=_p$ π-Ordered Resolution

The proof of Theorem 1 is divided into two parts: we prove that each system is equivalent to an intermediate system we call *π-half-ordered resolution*.

Recall that, for the system π-ordered resolution, the variable that is resolved on must be π-maximal in each antecedent. In π-half-ordered resolution, this is required of *at least one* of the antecedents. That is, *π-half-ordered resolution* is the subsystem of resolution defined by imposing the restriction

$$(\forall l \in C\ (\pi(l) < \pi(x_i))) \vee (\forall l \in D\ (\pi(l) < \pi(x_i)))$$

on the resolution rule (1). Clearly, π-half-ordered resolution simulates π-ordered resolution but, somewhat surprisingly, it doesn't have any additional power over it.

Theorem 3. *For any fixed order π on the variables, π-ordered resolution is polynomially equivalent to π-half-ordered resolution.*

We prove Theorem 3 by applying a sequence of transformations to a π-half-ordered refutation that, with the aid of the following definition, can be shown to make it incrementally closer to a π-ordered resolution refutation.

Definition 7. *A resolution refutation is π-**ordered up to** k if it satisfies the property that if any two clauses are resolved on a variable $x_i \in Var_\pi^k$, then all resolution steps above it are on variables in $Var_\pi^{\pi(i)-1}$.*

The π-ordered refutations are then precisely those that are π-ordered up to $n-1$. Now in order for these transformations not to blow up the size of the refutation, we need to carefully keep track of its structure throughout the process. As such, the proof of Theorem 3 depends heavily on resolution graphs and related definitions introduced in Sect. 2.

Proof. Let Π be a π-half-ordered resolution refutation of τ. Without loss of generality, assume $\pi = \mathrm{id}$; otherwise, rename variables. We will construct by induction on k (satisfying $0 \le k \le n-1$) a π-half-ordered resolution refutation Π_k of τ which is ordered up to k. For the base case, let $\Pi_0 \stackrel{\text{def}}{=} \Pi$. Suppose now Π_k has been constructed. Without loss of generality, assume that Π_k is connected; otherwise, take the subrefutation below any occurrence of 0.

Consider the set of nodes whose clauses are k-small. Note that this set is parent-complete. We claim that it is also upward-closed and, hence, path-complete. Indeed, let u be a parent of v and assume that $c_{\Pi_k}(u)$ is k-small. Then, since we disallow weakenings, $c_{\Pi_k}(v)$ is obtained by resolving on a variable $x_i \in \mathrm{Var}_\pi^k$. Since Π_k is ordered up to k, it follows that $\mathrm{Var}(c_{\Pi_k}(v)) \subseteq \mathrm{Var}_\pi^{i-1} \subseteq \mathrm{Var}_\pi^k$; otherwise, some variable in $c_{\Pi_k}(v)$ would have remained unresolved on a path connecting v to the sink (here we use the fact that Π_k is connected). Hence $c_{\Pi_k}(v)$ is also k-small.

By Proposition 1, this set defines a subrefutation of the clauses labeling the independent set

$$L_k \stackrel{\text{def}}{=} \min_{\Pi_k}\{v \mid c_{\Pi_k}(v) \text{ is } k\text{-small}\}. \tag{6}$$

Furthermore, L_k splits Π_k into two parts, i.e., $V(\Pi_k) = \mathrm{ucl}_{\Pi_k}(L_k) \cup \mathrm{dcl}_{\Pi_k}(L_k)$ and $L_k = \mathrm{ucl}_{\Pi_k}(L_k) \cap \mathrm{dcl}_{\Pi_k}(L_k)$. Let D denote the subproof on $\mathrm{dcl}_{\Pi_k}(L_k)$ and let U denote the *subrefutation* on $\mathrm{ucl}_{\Pi_k}(L_k)$. Note that D is comprised of all nodes in Π that are labeled by a clause that is not k-small or belong to L_k, and U is comprised of all nodes labeled by a k-small clause. In particular, all axioms are in D, all resolutions in U are on the variables in Var_π^k, and, since Π_k is ordered up to k, all resolutions in D are on the variables not in Var_π^k. Define

$$M \stackrel{\text{def}}{=} \min_D\{w \mid c_{\Pi_k}(w) \text{ is the result of resolving two clauses on } x_{k+1}\}. \tag{7}$$

If M is empty, $\Pi_{k+1} \stackrel{\text{def}}{=} \Pi_k$. Otherwise, suppose $M = \{w_1, \ldots, w_s\}$ and define $A_i \stackrel{\text{def}}{=} \mathrm{ucl}_D(\{w_i\})$. We will eliminate all resolutions on x_{k+1} in D by the following process, during which *the set of nodes stays the same*, while edges and clause-labeling function possibly change. More precisely, we update D in s rounds, defining a sequence of π-half-ordered resolution + *weakening* proofs $D_1, D_2, \ldots, D_s$. Initially $D_0 \stackrel{\text{def}}{=} D$. Fix now an index i. Let c_{i-1} denote the clause-labeling $c_{D_{i-1}}$. To define the transformation of D_{i-1} to D_i, we need the following structural properties of D_{i-1}, which are easily verified by induction simultaneously with the definition.

Claim. In the following properties, let u and v be arbitrary vertices in $V(D)$.

a. If v is not above u in D, then the same is true in D_{i-1};
b. the clause $c_{i-1}(v)$ is equal to $c_D(v)$ or $c_D(v) \vee x_{k+1}$ or $c_D(v) \vee \overline{x_{k+1}}$;
c. if $v \notin \bigcup_{j=1}^{i-1} A_j$ then $c_{i-1}(v) = c_D(v)$ and, moreover, this clause is obtained in D_{i-1} with the same resolution as in D;
d. D_{i-1} is a π-half-ordered resolution + weakening proof.

Let us construct D_i from D_{i-1}. By property (c) and the fact that M is independent, the resolution step at w_i is unchanged from D to D_{i-1}. Let w' and w'' denote the parents of w_i in D and let $c_D(w') = B \vee x_{k+1}$ and $c_D(w'') = C \vee \overline{x_{k+1}}$. Since Π_k is π-half-ordered, either B or C is k-small. Assume without loss of generality that B is k-small.

Recall that there is no resolution in D on variables in Var_π^k. Thus, for all $v \in A_i$, it follows that B is a subclause of $c_D(v)$, and by property (b), we have the following crucial property:

$$\text{For all } v \in A_i, B \text{ is a subclause of } c_{i-1}(v). \tag{8}$$

By property (a), A_i remains upward closed in D_{i-1}. Accordingly, as the first step, for any $v \notin A_i$ we set $c_i(v) := c_{i-1}(v)$ and we leave its incoming edges unchanged.

Next, we update vertices $v \in A_i$ in an arbitrary D-topological order maintaining the property that $c_i(v) = c_{i-1}(v)$ or $c_i(v) = c_{i-1}(v) \vee \overline{x_{k+1}}$. In particular, $c_i(v) = c_{i-1}(v)$ whenever $c_{i-1}(v)$ contains the variable x_{k+1}.

First we set $c_i(w_i) := c_{i-1}(w_i) \vee \overline{x_{k+1}}$ and replace the incoming edges by a weakening edge from w''. This is possible since $c_{i-1}(w_i) = c_D(w_i)$ by property (c) and, hence, does not contain x_{k+1} by virtue of being in M.

For $v \in A_i \setminus \{w_i\}$, we proceed as follows.

1. If $x_{k+1} \in c_{i-1}(v)$, keep the clause but replace incoming edges with a weakening edge (w', v). This is well-defined by (8). Also, since $w' < w < v$ in D, we maintain property (a).
2. If $c_{i-1}(v) = \mathrm{Res}(c_{i-1}(u), c_{i-1}(w))$ on x_{k+1} where $\overline{x_{k+1}} \in c_{i-1}(u)$, set $c_i(v) := c_{i-1}(v) \vee \overline{x_{k+1}}$ – equivalently, $c_{i-1}(v) \vee c_i(u)$ – and replace incoming edges by a weakening edge (u, v).
3. If $c_{i-1}(v)$ is weakened from $c_{i-1}(u)$ and $x_{k+1} \notin c_{i-1}(v)$, set $c_i(v) := c_{i-1}(v) \vee c_i(u)$. In other words, we append the literal $\overline{x_{k+1}}$ to $c_i(v)$ if and only if this was previously done for $c_i(u)$.
4. Otherwise, $x_{k+1} \notin c_{i-1}(v)$ and $c_{i-1}(v) = \mathrm{Res}(c_{i-1}(u), c_{i-1}(w))$ on some x_ℓ where $\ell > k+1$. In particular, $x_{k+1} \notin \mathrm{Var}(c_{i-1}(u)) \cup \mathrm{Var}(c_{i-1}(w))$. Set $c_i(v) := \mathrm{Res}(c_i(u), c_i(w))$ that is, like in the previous item, we append $\overline{x_{k+1}}$ if and only if it was previously done for either $c_i(v)$ or $c_i(w)$. Since $\ell > k+1$, this step remains π-half-ordered.

This completes our description of D_i. It is straightforward to verify that D_s is a π-half-ordered resolution + weakening proof *without* resolutions on x_{k+1}.

To finally construct Π_{k+1}, we reconnect D_s to U along L_k and then remove any weakenings introduced in D_s. This may require adding new nodes, as it may be the case that $c_s(v) \neq c_D(v)$ for some $v \in L_k$. But, in this case, it is straightforward to verify, using (8), that there is a vertex $w \in \mathrm{dcl}_D(M) \setminus \{M\}$ such that $c_D(v) = \mathrm{Res}(c_s(w), c_s(v))$ on x_{k+1}, and this resolution is half-ordered. In fact, w can be taken to be a parent in D of some nodes in M. Thus, when necessary we add to D_s a new node $\tilde{v}$ labeled by $\mathrm{Res}(c_s(w), c_s(v))$ and add the edges $(v, \tilde{v})$ and $(w, \tilde{v})$.

Denote by $\widetilde{\Pi}_{k+1}$ the result of connecting D_s and U along the vertices in L_k and this newly added collection of vertices. Since neither D_s nor U contain resolutions on x_{k+1} except for those in the derivations of the clauses just added to D_s, it follows that $\widetilde{\Pi}_{k+1}$ is a π-half-ordered resolution + weakening refutation that is *ordered up to* $k+1$. Let Π_{k+1} be obtained by contracting all weakenings.

It only remains to analyze its size (note that *a priori* it can be doubled at every step, which is unacceptable). Since

$$|\Pi_{k+1}| \leq |\Pi_k| + |L_k|, \tag{9}$$

we only have to control $|L_k|$. For that we will keep track of the invariant $|\mathrm{dcl}_{\Pi_k}(L_k)|$; more precisely, we claim that

$$|\mathrm{dcl}_{\Pi_{k+1}}(L_{k+1})| \leq |\mathrm{dcl}_{\Pi_k}(L_k)|. \tag{10}$$

Let us prove this by constructing an injection from $\mathrm{dcl}_{\Pi_{k+1}}(L_{k+1})$ to $\mathrm{dcl}_{\Pi_k}(L_k)$; we will utilize the notation from above.

First note that the resolution + weakening refutation $\widetilde{\Pi}_{k+1}$ and its weakening-free contraction Π_{k+1} can be related as follows. For every node $v \in V(\Pi_{k+1})$ there exists a node $v^* \in V(\widetilde{\Pi}_{k+1})$ with $c_{\widetilde{\Pi}_{k+1}}(v^*) \supseteq c_{\Pi_{k+1}}(v)$ which is *minimal* among those contracting to v. If v is an axiom node of Π_{k+1} then so is v^* in $\widetilde{\Pi}_{k+1}$. Otherwise, if u and w are the two parents of v, and if u' and w' are the corresponding parents of v^* (v^* may not be obtained by weakening due to the minimality assumption), then $c_{\widetilde{\Pi}_{k+1}}(u^*)$ is a subclause of $c_{\widetilde{\Pi}_{k+1}}(u')$ and $c_{\widetilde{\Pi}_{k+1}}(w^*)$ is a subclause of $c_{\widetilde{\Pi}_{k+1}}(w')$. We claim that $(v \mapsto v^*)\,|_{\mathrm{dcl}_{\Pi_{k+1}}(L_{k+1})}$ (which is injective by definition) is the desired injection. We have to check that its image is contained in $\mathrm{dcl}_{\Pi_k}(L_k)$.

Fix $v \in \mathrm{dcl}_{\Pi_{k+1}}(L_{k+1})$. Then by definition of L_k, *either v is an axiom or both its parents are not $(k+1)$-small*. By the above mentioned facts about the contraction $\widetilde{\Pi}_{k+1} \to \Pi_{k+1}$, this property is inherited by v^*. In particular, $v^* \notin \{\tilde{w} \mid w \in L_k\}$ as all nodes in this set have at least one $(k+1)$-small parent due to half-orderedness. Finally, since the corresponding clauses in D and D_s differ only in the variable x_{k+1}, v^* cannot be in U, for the same reason (recall that all axioms are in D). Hence $v^* \in V(D_s) = V(D) = \mathrm{dcl}_{\Pi_k}(L_k)$.

Having thus proved (10), we conclude by the obvious induction that $|L_k| \leq |\mathrm{dcl}_{\Pi_k}(L_k)| \leq |\mathrm{dcl}_{\Pi_0}(L_0)| \leq |\Pi|$. Then (9) implies $|\Pi_{n-1}| \leq n|\Pi|$, as desired.

At last, we must show the equivalence holds for the corresponding CDCL system. We provide a sketch of the proof.

Theorem 4. *For any fixed order π on the variables,* $\mathsf{CDCL}(\pi\text{-}\mathsf{D}, \mathsf{DECISION\text{-}L})$ *is polynomially equivalent to π-half-ordered resolution.*

Proof. As above, assume $\pi = \mathrm{id}$. The fact that $\mathsf{CDCL}(\pi\text{-}\mathsf{D}, \mathsf{DECISION\text{-}L})$ polynomially simulates π-half ordered resolution is almost trivial. A π-half-ordered resolution step deriving $\mathrm{Res}(C \vee x_i, D \vee \overline{x_i})$ can be directly simulated by constructing a trail t that falsifies $C \vee D$ and contains a single unit propagation on x_i. This is possible since C or D is i-small. Then $C \vee D$ can be easily learned using t.

The other direction is just slightly more involved. It suffices to show that for a learning step $(\tau, t) \overset{(D,t^*)}{\Longrightarrow} (\tau \cup \{D\}, t^*)$, there is a short π-half-ordered resolution proof of D from τ. Any learned clause can be thought of naturally as the result of a sequence of resolutions; there are clauses $C_1, \ldots, C_{k+1}$ in τ and variables $x_{i_1}, \ldots, x_{i_k}$ assigned by unit propagation in t from which we can inductively define

$$C'_{k+1} \overset{\mathrm{def}}{=} C_{k+1} \qquad \text{and} \qquad C'_j \overset{\mathrm{def}}{=} \mathrm{Res}(C_j, C'_{j+1})$$

where C_j and C'_{j+1} are resolvable on x_{i_j} and $D = C'_1$. These resolutions may not all be π-half-ordered, but they can be reordered and duplicated to derive the same clause while maintaining π-half-orderedness. Formally, we define by

double induction a different collection of derivable clause: for γ and j in $[k+1]$ satisfying $j < \gamma$, $C_{\gamma,\gamma} \stackrel{\text{def}}{=} C_\gamma$ and

$$C_{\gamma,j} \stackrel{\text{def}}{=} \begin{cases} \text{Res}(C_{j,1}, C_{\gamma,j+1}) & C_{j,1} \text{ and } C_{\gamma,j+1} \text{ are resolvable on } x_{i_j} \\ C_{\gamma,j+1} & \text{otherwise.} \end{cases}$$

Because of π-D, the only literals appearing in C_j that are potentially larger than x_{i_j} (with respect to π) are the other variables assigned by unit propagation in t and resolved on in the derivation of C_1'. One can show that the clause $C_{j,1}$ is the result of "washing out" these other literals so that x_{i_j} appears maximally. Since all resolutions on are clauses of this form, they are all π-half-ordered. And for the learning scheme DECISION-L, the learned clause C_1' contains *only* decision variables in t, so this reorganizing of resolutions does not affect the final derived clause; it can be verified that $C_{k+1,1} = C_1' = D$. In total, this derivation of $C_{k+1,1}$ (and, hence, D) is π-half ordered and has at most n^2 resolutions.

4 CDCL(FIRST-L, π-D) $=_p$ General Resolution

This result is by far the most technical. It would have been impossible to give a satisfying treatment in the space available, but in the interest of providing some idea of its formal aspects, we briefly discuss our approach. As in the previous section, the proof is divided into two parts: we prove that each system is equivalent to an intermediate system we call *π-trail resolution.*

Definition 8. *Fix an order π on the variables. The proof system π-trail resolution is defined as follows. Its lines are either clauses or trails, where the empty trail is an axiom. It has the following rules of inference:*

$$\frac{t}{[t, x_i \stackrel{d}{=} a]}, \quad \textit{(Decision rule)}$$

where x_i is the π-smallest index such that x_i does not appear in t and $a \in \{0,1\}$ is arbitrary;

$$\frac{t \qquad C}{[t, x_i \stackrel{u}{=} a]}, \quad \textit{(Unit propagation rule)}$$

where $C|_t = x_i^a$;

$$\frac{C \vee x_i^a \qquad D \vee x_i^{1-a} \qquad t}{C \vee D}, \quad \textit{(Learning rule)}$$

where $(C \vee D)|_t = 0$, $(x_i \stackrel{*}{=} a) \in t$ and all other variables of C appear before x_i in t.

Without the unit propagation rule, this is just π-half-ordered resolution, modulo additional traffic in trails. It follows almost directly from its definition that π-trail resolution is polynomially equivalent to $\mathsf{CDCL}(\mathsf{FIRST\text{-}L}, \pi\text{-}\mathsf{D})$ (and even $\mathsf{CDCL}(\pi\text{-}\mathsf{D})$). Our main technical contribution is proving the following.

Theorem 5. *For any fixed order π on the variables, π-trail resolution polynomially simulates general resolution.*

The key observation is that, due to the unit propagation rule, π-trail resolution becomes significantly more powerful when the underlying formula has many unit clauses. Thus, we design the simulation algorithm to output a derivation of *all* unit clauses appearing in the given refutation Π and then recursively apply it to gain access to more unit clauses throughout the procedure. At first glance, it might seem reasonable to recursively apply the simulation algorithm to various restrictions of Π, but restriction as an operation has two flaws with regards to our objectives. First, the results of different restrictions on proofs often *overlap*; for example, when viewing restriction as an operation on resolution graphs, the graphs of $\Pi|_{x_i=0}$ and $\Pi|_{x_i=1}$ will likely share vertices from $G(\Pi)$. This leads to an exponential blow-up in the size of the output if one is not careful. Second, restrictions may *collapse* parts of the Π; for example, if ρ falsifies an axiom of Π, then $\Pi|_\rho$ is the trivial refutation and it is impossible to extract anything from it by recursively applying our simulation algorithm.

To make this approach feasible, we introduce a new operator, which may be of independent interest, called *variable deletion*; it is an analogue of restriction for sets of variables as opposed to sets of variable assignments. This operator has the property that it always yields a nontrivial refutation (for proper subsets of variables), and its size and structure are highly regulated by the size and structure of the input refutation. This allows for a surgery-like process; we simulate small local pieces of the refutation and then stitch them together into a new global refutation. For complete details, see the full version of this paper [16].

5 Conclusion

Our work continues the line of research aimed at better understanding theoretical limitations of CDCL solvers. We have focused on the impact of decision strategies, and we have considered the simple strategy that always chooses the first available variable under a fixed ordering. We have shown that, somewhat surprisingly, the power of this model heavily depends on the learning scheme employed and may vary from ordered resolution to general resolution.

In practice, the fact that $\mathsf{CDCL}(\mathsf{DECISION\text{-}L}, \pi\text{-}\mathsf{D}, \mathsf{ALWAYS\text{-}C}, \mathsf{ALWAYS\text{-}U})$ is not as powerful as resolution supports the observation that CDCL solvers with the ordered decision strategy are often less efficient than those with more powerful decision strategies like VSIDS. But, although DECISION-L is an asserting learning strategy, most solvers use more efficient asserting strategies like *1-UIP*. A natural open question then is what can be proved if DECISION-L is replaced with some other amendment modeling a different, possibly more

practical asserting learning scheme? Furthermore, what is the exact strength of CDCL(π-D, ALWAYS-C, ALWAYS-U)?

References

1. Aloul, F.A., Markov, I.L., Sakallah, K.A.: MINCE: a static global variable-ordering for SAT and BDD. In: International Workshop on Logic and Synthesis, pp. 1167–1172 (2001)
2. Aloul, F.A., Markov, I.L., Sakallah, K.A.: FORCE: a fast & easy-to-implement variable-ordering heuristic. In: Proceedings of the 13th ACM Great Lakes Symposium on VLSI, pp. 116–119. ACM (2003)
3. Atserias, A., Fichte, J.K., Thurley, M.: Clause-learning algorithms with many restarts and bounded-width resolution. J. Artif. Intell. Res. **40**, 353–373 (2011)
4. Beame, P., Karp, R., Pitassi, T., Saks, M.: The efficiency of resolution and Davis-Putnam procedures. SIAM J. Comput. **31**(4), 1048–1075 (2002)
5. Beame, P., Kautz, H., Sabharwal, A.: Towards understanding and harnessing the potential of clause learning. J. Artif. Intell. Res. **22**, 319–351 (2004)
6. Beame, P., Sabharwal, A.: Non-restarting SAT solvers with simple preprocessing can efficiently simulate resolution. In: Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence, pp. 2608–2615 (2014)
7. Ben-Sasson, E., Johannsen, J.: Lower bounds for width-restricted clause learning on small width formulas. In: Strichman, O., Szeider, S. (eds.) SAT 2010. LNCS, vol. 6175, pp. 16–29. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-14186-7_4
8. Ben-Sasson, E., Wigderson, A.: Short proofs are narrow - resolution made simple. J. ACM **48**(2), 149–169 (2001)
9. Bonet, M.L., Buss, S., Johannsen, J.: Improved separations of regular resolution from clause learning proof systems. J. Artif. Intell. Res. **49**, 669–703 (2014)
10. Bonet, M.L., Esteban, J.L., Galesi, N., Johannsen, J.: On the relative complexity of resolution refinements and cutting planes proof systems. SIAM J. Comput. **30**(5), 1462–1484 (2000)
11. Buss, S.R., Hoffmann, J., Johannsen, J.: Resolution trees with lemmas: resolution refinements that characterize DLL-algorithms with clause learning. Logical Methods Comput. Sci. **4**(4), 1–28 (2008)
12. Buss, S.R., Kołodziejczyk, L.A.: Small stone in pool. Logical Methods Comput. Sci. **10**(2), 1–22 (2014). https://lmcs.episciences.org/852/pdf
13. Elffers, J., Johannsen, J., Lauria, M., Magnard, T., Nordström, J., Vinyals, M.: Trade-offs between time and memory in a tighter model of CDCL SAT solvers. In: Creignou, N., Le Berre, D. (eds.) SAT 2016. LNCS, vol. 9710, pp. 160–176. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-40970-2_11
14. Hertel, P., Bacchus, F., Pitassi, T., Van Gelder, A.: Clause learning can effectively p-simulate general propositional resolution. In: Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence, pp. 283–290 (2008)
15. Li, C., Fleming, N., Vinyals, M., Pitassi, T., Ganesh, V.: Towards a complexity-theoretic understanding of restarts in SAT solvers. In: Pulina, L., Seidl, M. (eds.) Theory and Applications of Satisfiability Testing - SAT 2020. LNCS, vol. 12178, pp. 233–249. Springer, Cham (2020)
16. Mull, N., Pang, S., Razborov, A.: On CDCL-based proof systems with the ordered decision strategy. arXiv preprint arXiv:1909.04135 (2019)

17. Nieuwenhuis, R., Oliveras, A., Tinelli, C.: Solving SAT and SAT modulo theories: from an abstract Davis-Putnam-Logemann-Loveland procedure to DPLL(T). J. ACM **53**(6), 937–977 (2006)
18. Pipatsrisawat, K., Darwiche, A.: On the power of clause-learning SAT solvers as resolution engines. Artif. Intell. **175**(2), 512–525 (2011)
19. Van Gelder, A.: Pool resolution and its relation to regular resolution and DPLL with clause learning. In: Sutcliffe, G., Voronkov, A. (eds.) LPAR 2005. LNCS (LNAI), vol. 3835, pp. 580–594. Springer, Heidelberg (2005). https://doi.org/10.1007/11591191_40
20. Vinyals, M.: Hard examples for common variable decision heuristics. In: Proceedings of the 34th AAAI Conference on Artificial Intelligence (AAAI 2020) (2020)