On CDCL-based proof systems with the ordered decision strategy

Nathan Mull* Shuo Pang† Alexander Razborov‡

Abstract

We prove that conflict-driven clause learning 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 that stops after the first non-conflict clause when backtracking, they become equivalent to general resolution. To the best of our knowledge, along with [40] this is the first theoretical study of the interplay between specific decision strategies and clause learning.

For 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 discussing CDCL-based proof systems that allow for succinct and precise theorem statements.[1]

1. Introduction

SAT-solvers have become standard tools in many application domains such as hardware verification, software verification, automated theorem proving, scheduling and computational biology (see [24, 26, 16,

*University of Chicago, Department of Computer Science, nmull@cs.uchicago.edu.

†University of Chicago, Department of Mathematics, spang@math.uchicago.edu.

‡University of Chicago, USA, razborov@math.uchicago.edu and Steklov Mathematical Institute, Moscow, Russia, razborov@mi-ras.ru.

[1]An abbreviated version of this paper appeared in *Theory and Applications of Satisfiability Testing – SAT 2020* [34]. The current version contains the same results with complete proofs.

32, 19] among the others). Since their conception in the early 1960s, SAT-solvers have become significantly more efficient, but they have also become significantly more complex. Consequently, there has been increasing interest in understanding the theoretical limitations and strengths of contemporary SAT-solvers. Much of the recent literature has focused on the connections between SAT-solvers and subsystems of the resolution proof system originally introduced in [12, 38].

This connection essentially started with the Davis-Putnam-Logemann-Loveland procedure (DPLL) [22, 21], a backtracking search algorithm that builds partial assignments one literal at a time until a satisfying assignment is found or all assignments have been exhausted. Since DPLL is sound and complete, its computational trace when applied to an unsatisfiable formula is a *proof* of unsatisfiability. It is generally accepted as a folklore result that the computational trace of DPLL on an unsatisfiable formula can be converted into a tree-like resolution refutation. Thus, tree-like resolution lower bounds imply DPLL running time lower bounds. And in some sense, these lower bounds are tight: DPLL, given oracle access to a tree-like resolution refutation Π of the input formula, can run in time that is polynomial in the length of Π. That is, DPLL is essentially equivalent to tree-like resolution and thus can be viewed as a propositional proof system in the Cook-Reckhow sense [20].

Nearly all contemporary SAT-solvers are variants of DPLL augmented with modern algorithmic techniques and heuristics. The technique most often credited for their success is *conflict-driven clause learning* (CDCL) [27, 31], so these solvers are interchangeably called CDCL SAT-solvers, CDCL solvers, or simply CDCL (for further information regarding the design of SAT-solvers, see the *Handbook of Satisfiability* [11]). Just as with DPLL, the computational trace of CDCL can be converted into a resolution refutation, but may no longer be tree-like or even regular. Thus, general resolution lower bounds imply CDCL running time lower bounds, but it is unclear *a priori* whether these bounds are tight in the same sense as above.

The line of work on the question of whether CDCL solvers simulate general resolution was initiated by Beame et al. [6] and continued by many others [39, 35, 25, 17, 8, 37, 3, 23]. The primary difference between all these papers is in the details of the model, the models considered by Pipatsrisawat and Darwich [37] and Atserias et al. [3] being perhaps the most faithful to actual implementations of CDCL SAT-solvers. But almost all models appearing in the literature make a few nonstandard assumptions.

1. *Very frequent restarts.* The solver restarts roughly $O(n^2)$ times for every clause in the given resolution refutation Π (where n is the total number of variables). Though many solvers do restart frequently in practice [10], it is unclear if this is really necessary for the strength of CDCL.

2. *No clause deletion policy.* The solver has to keep every learned clause. In practice, some solvers periodically remove half of all learned clauses [4].

3. *Nondeterministic decision strategy.* The solver uses oracle access to Π to construct a very particular decision strategy. In practice, solvers use heuristics [30, 33, 29].

It is natural to ask whether these assumptions can be weakened or removed entirely. In this respect, the first two assumptions have become topics of recent interest. With regards to the first, much research has been dedicated to the study of *nonrestarting* SAT-solvers [39, 17, 18, 14, 7, 28]. The exact strength of CDCL without restarts is still unknown and, arguably, makes for the most interesting open problem in the area. With regards to the second, Elffers et al. [23] proved size-space tradeoffs in a very tight model of CDCL, which may be interpreted as results about aggressive clause deletion policies.

In this paper we are primarily concerned with the third assumption, i.e., how much does the efficiency of CDCL-solvers depend on the nondeterminism in the decision strategy? We study a simple decision strategy that we call the *ordered* decision strategy which is identical to the strategy studied by Beame et al. [5] in the context of DPLL without clause learning. It is defined naturally: when the solver has to choose a variable to assign, the ordered decision strategy dictates that it chooses the smallest unassigned variable according to some fixed order. There is still a choice in whether to fix the variable to 0 (*false*) or 1 (*true*), and we allow the solver to make this choice 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 to “cut the line” is precisely what makes the situation much 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 su-*

perpolynomial time for CDCL with any ordered decision strategy?

Before describing our contributions towards this question, let us briefly review analogous separations in the context of proof and computational complexities. Bonet et al. [15] proved that a certain family of formulas requires exponential-sized ordered resolution refutations but has polynomial-sized regular resolution refutations. Bollig et al. [13] proved that a certain boolean function requires exponential-sized ordered binary decision diagrams (OBDDs) but have polynomial-sized general BDDs. These results tell us that order tends to be a strong restriction, and the above question asks whether this same phenomenon occurs for CDCL. It is also worth noting that this question may be motivated as a way of understanding the strength of *static* decision strategies such as MINCE [1] and FORCE [2]. But since such decision strategies are rarely used in practice we will not dwell on this anymore.

Our contributions

Per the discussion above, a proof system that captures any class of CDCL solvers should be no stronger than general resolution. It can also be reasonably expected (and in two particular situations will be verified below as easy directions of Theorems 2.14, 2.15) that with any ordered decision strategy, they should be at least as strong as ordered resolution with respect to the same order. Our main results show that, for a nondetermistic model of CDCL in which the solver may arbitrarily choose conflict/unit clauses if there are several, may elect not to do conflict analysis/unit propagations at all, and may restart at any time, both extremes are attained. In this setting, the strength of the system depends on the *learning scheme* employed; that is, it depends on the method used to determine which clauses are learned after conflict analysis. 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 2.14). 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 2.15).

Remark 1 As the name suggests, DECISION-L is the same as the so-called DECISION learning scheme used in practice.[2] Hence these two results, taken together, go somewhat against the "common wisdom." Namely, it turns out that in the case of ordered decision strategy, an assertive learning scheme is badly out-performed by a scheme that, to the best of our knowledge, has not been used before. That said, FIRST-L is similar to the learning scheme FirstNewCut [6], and both schemes have the property that they are designed somewhat artificially to *target* particular resolution steps in a given refutation.

We also prove linear width lower bounds for CDCL with the ordered decision strategy (Theorem 2.16), which are in sharp contrast with the size-width relationship for general resolution proved by Ben-Sasson and Wigderson [9].

With the ability of possibly postponing conflict analysis and unit propagation, the model of CDCL we consider differs in these aspects from solvers that occur in practice. This is in part because our intention is to focus on the impact of decision strategies. But this substantial amount of nondeterminism also allows us to identify two proof systems that are, more or less straightforwardly, *equivalent* to the corresponding CDCL variant. (This correspondence is very much like the correspondence between *regWRTI* and a variant of CDCL with similar nonstandard features called *DLL-LEARN*, both introduced by Buss et al. [18, 14].) Determining the exact power of these systems constitutes the main technical part of this paper.

The first proof system might be of independent interest; we call it *half-ordered resolution.* For a given order on the variables, ordered resolution can be alternatively described by the requirement that in every application of the resolution rule, the resolved variable is larger than any other variable appearing in both of the two antecedent clauses. We relax this requirement by asking that this property holds for *at least one* of them, which reflects the inherent asymmetry in resolution rules resulting from clause learning in CDCL solvers. Somewhat surprisingly (at least to us), it turns out (Theorem 2.6) that this relaxation does not add any extra power, and half-ordered resolution is polynomially equivalent to ordered resolution with respect to the same order.

The second proof system, which we call *trail resolution*, extends half-ordered resolution and is more auxiliary in nature. It is based on the observation that with the amount of nondeterminism we allow, all

[2] We use this slightly different name so that it fits our naming conventions below.

trails[3] that a CDCL solver manages to create can be easily recreated when needed. Accordingly, the system works with lines of two types, one for clauses and another for trails. Clauses entail nontrivial trails via a unit propagation rule while trails can be used to enhance the half-ordered resolution rule. We show that trail resolution is polynomially equivalent to resolution (Theorem 2.18), and since it is by far our most difficult result, let us reflect a bit on the ideas in its proof.

Like other CDCL-based proof systems, trail resolution is not closed under restrictions or weakening, so many standard methods do not apply. Instead, we use two operations on resolution proofs (lifting and variable deletion) in tandem with some additional structural information to give us a fine-grained understanding of the size and structure of the general resolution refutation being simulated. The properties of these operators allow for a surgery-like process; we simulate small local pieces of the refutation and then stitch them together into a new global refutation.

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 [35, 3, 23]. However, the primary goal of our model is to *highlight* possible non-standard sources of nondeterminism in variants of CDCL, as opposed to creating a model completely faithful to applications. For example, Theorem 2.15 can be written in this language as:

> *For any order* π*,* CDCL(FIRST-L, π-D) *is equivalent to general resolution.*

We will also try to pay a special attention to finer details of the model sometimes left implicit in previous works. This entails several subtle choices to be made, and we interlace the mathematical description of our model with informal discussion of these choices.

The paper is organized as follows. In Section 2 we give all necessary definitions and formulate our main results as we go along.

In Section 3 we prove Theorem 2.14 on the power of CDCL with the ordered decision strategy and the DECISION-L learning strategy. Section 3.1 contains proof-complexity theoretic arguments about half-ordered resolution, while in Section 3.2 we establish its translation to the language of CDCL.

[3] A *trail* is essentially an ordered partial assignment constructed by CDCL during its execution.

In Section 4 we prove Theorem 2.15 on the power of CDCL with the ordered decision strategy and the FIRST-L learning strategy. To that end, in Section 4.1 we show the equivalence of this system to trail resolution (mentioned above) and in Section 4.2 we establish that trail resolution is actually equivalent to general resolution (Theorem 2.18).

In Section 5 we prove Theorem 2.16 that, roughly speaking, states that the simulation provided by Theorem 2.15 fails extremely badly with respect to width. Among other things, this implies that there does not seem to exist any useful width-size relation in the context of CDCL with ordered decision strategy.

We conclude in Section 6 with a few remarks and suggestions for future work.

Related works

Recently, Vinyals [40] studied the strength of decision heuristics in CDCL. He showed that CDCL with the popular VSIDS decision strategy (among others) does not simulate resolution, or more precisely, ordered resolution, as the hard tautology constructed there is easy for a certain order.

2. Preliminaries and main results

Throughout the paper, we assume that the set of propositional variables is fixed as $V \stackrel{\text{def}}{=} \{x_1, \dots, x_n\}$. 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, $\mathrm{Var}(C)$ is the set of variables appearing in C. A *CNF* is a set of clauses thought of as their conjunction. For a CNF τ, $\mathrm{Var}(\tau)$ is 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. A *w-CNF* is a CNF in which all clauses have width $\leq w$.

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 make use of the notation $\mathrm{Res}(C \vee x_i^a, D \vee x_i^{1-a})$ for the conclusion clause $C \vee D$.

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, $S_R(\tau \vdash C)$ is the minimal possible size of a resolution proof of the clause C from clauses in τ (∞ if C is not implied by τ). Likewise, $w(\tau \vdash C)$ is 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.

Remark 2 Despite the above distinction, it is often convenient to consider systems that do allow the weakening rule. We make it clear when we do this by adding the annotation '+ weakening' to the system. For example, resolution + weakening is the resolution proof system with the weakening rule included.

Resolution graphs

Our results depend on the careful analysis of the structure of resolution proofs. For example, it will be useful for us to maintain structural properties of the proof while changing the underlying clauses and derivations. We build up the following collection of definitions for this analysis, to which we will refer throughout the later sections. The reader may skip this section for now and return to it in the future as needed.

Definition 2.1 For a resolution + weakening proof Π, its *resolution graph,* $G(\Pi)$, is a directed acyclic graph (DAG) 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. Every node has in-degree 0, 1, or

2 if its corresponding clause is an axiom, derived by weakening, or derived by resolving two clauses, respectively. Denote the set of nodes by $V(\Pi)$, and the clause at $v \in V(\Pi)$ by $c_\Pi(v)$. We do *not* assume that c_Π is injective, that is we allow the same clause to appear in the proof several times.

There is a natural partial order on $V(\Pi)$ reflecting the order of appearances of clauses in Π: $v > u$ if and only if v is a descendant of u, or equivalently, there is a (directed) path from u to v. We sometimes say that v is *above* (resp. *below*) u if $v > u$ (resp. $v < u$). If, moreover, (u, v) is an edge (directed from u to v), we say that u is a *parent* of v. A set of nodes is *independent* if any two nodes in the set are incomparable. Note that we have defined this order so that we naturally view resolution graphs in *bottom-up orientation*, where axioms appear at the bottom and derivations flow upwards.

Maximal and *minimal* nodes of any nonempty $S \subseteq V(\Pi)$ are defined with respect to this partial order: $\max_\Pi S \stackrel{\mathrm{def}}{=} \{v \in S : \forall u \in S \neg(v < u)\}$, and similarly for $\min_\Pi S$.

Definition 2.2 Let $S \subseteq V(\Pi)$. The *upward closure* and *downward closure* of S in $G(\Pi)$ are $\mathrm{ucl}_\Pi(S) \stackrel{\mathrm{def}}{=} \{v \in V(\Pi) : \exists w \in S(v \geq w)\}$ and $\mathrm{dcl}_\Pi(S) \stackrel{\mathrm{def}}{=} \{v \in V(\Pi) : \exists w \in S(v \leq w)\}$, respectively. A subset of nodes S is *parent-complete* if for any $v \in S$ of in-degree 2, one parent of v being in S implies that the other parent of v is also in S. It is *path-complete* if for any directed path p in $G(\Pi)$, the two end points of p being in S implies all nodes of p are.

Remark 3 The following are some basic facts about these definitions.

- The upward closure $\mathrm{ucl}_\Pi(S)$ is path-complete but need not be parent-complete.
- The downward closure $\mathrm{dcl}_\Pi(S)$ is always both path-complete and parent-complete.
- Path-completeness does not imply upward-closedness.
- The complement of any upward-closed set is downward-closed.

Also, these definitions behave naturally, as demonstrated by the following proposition.

Proposition 2.3 *Let $S \subseteq V(\Pi)$ be a nonempty set of nodes that is both parent-complete and path-complete. Then the induced subgraph on S in $G(\Pi)$ is the graph of a proof which derives $\max_\Pi S$ from $\min_\Pi S$.*

Proof. Let $S^* \subseteq S$ be the set of all nodes in S "provable" from $\min_\Pi S$ inside S. Formally, it is the closure of $\min_\Pi S$ according to the following rule: if $v \in S$ and all its parents are in S^* then v is also in S^*. We need to show that $S^* = S$.

Assume not, and fix an arbitrary $v \in \min_\Pi(S \setminus S^*)$. Since $v \notin \min_\Pi S$, there exists $w \in S$ below v. Since S is path-complete, we can assume w.l.o.g. that w is a parent of v, and since S is parent-complete, all parents of v are in S. Now, since v is minimal in $S \setminus S^*$, all of them must be actually in S^*. Hence $v \in S^*$, a contradiction.■

In the sequel, we refer to a proof (refutation) defined on a subgraph in this way as a *subproof* (*subrefutation*).

Definition 2.4 A resolution graph is *connected* if $|\max_\Pi V(\Pi)| = 1$, i.e., there is a unique sink.

This is **not** the usual definition of connectedness for directed graphs. But it implies that every node can be connected to the unique sink by a directed path, and thus implies the weak connectedness in the usual sense (i.e., there is an undirected path between any two nodes).

Remark 4 For a resolution proof Π and $v \in V(\Pi)$, the subgraph on $\mathrm{dcl}_\Pi(\{v\})$ is a connected resolution graph whose axiom nodes are among axiom nodes of $G(\Pi)$.

Ordered and half-ordered resolution

Fix now an order $\pi \in S_n$. For any literal $l = x_k^a$, $\pi(l) \stackrel{\text{def}}{=} \pi(k)$. 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):

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

That is, the two antecedents are i-small. We note that in the literature this system is usually defined differently, namely in a top-down manner (see e.g. [15]). It is easy to see, however, that our version is equivalent.

Definition 2.5 *π-half-ordered resolution* is the subsystem of resolution in which the rule (1) is restricted by the requirement

$$\forall l \in C \ (\pi(l) < \pi(x_i)). \tag{2}$$

That is, at least one of the antecedents is i-small.

Recall [20] that a proof system P p-simulates another proof system Q if there exists a polynomial time algorithm that takes any Q-proof to a P-proof from the same axioms (in particular, the size of the P-proof is bounded by a polynomial in the size of the original proof). Two systems P and Q are *polynomially equivalent* if they p-simulate each other.

We are now ready to state our first result.

Theorem 2.6 *For any order $\pi \in S_n$, π-ordered resolution is polynomially equivalent to π-half-ordered resolution.*

The next proof system, π-*trail resolution*, is even more heavily motivated by CDCL solvers. For this reason we interrupt our proof-complexity exposition to define the corresponding model. As we noted in the introduction, we will try to highlight certain subtle points in the definition of the model by injecting informal remarks.

2.1. CDCL-based proof systems

A *unit clause* is a clause consisting of a single literal. An *assignment* is an expression of the form $x_i = a$ $(1 \leq i \leq n,\ a \in \{0,1\})$. A *restriction* ρ is a set of assignments in which all variables are pairwise distinct. We denote by $\mathrm{Var}(\rho)$ the set of all variables appearing in ρ. Restrictions naturally act on clauses, CNFs, resolution proofs, etc.; we denote by $C|_\rho,\ \tau|_\rho,\ \Pi|_\rho \ldots$ the result of this action. Note that both π-ordered resolution and π-half-ordered resolution are closed under restrictions, i.e., if Π is a π-(half)-ordered resolution proof, then $\Pi|_\rho$ is a $\pi|_\rho$-(half)-ordered resolution proof of no-bigger size, where $\pi|_\rho$ is the order induced by π on $V \setminus \mathrm{Var}(\rho)$.

Remark 5 Restrictions of proofs also act on resolutions graphs, i.e., they give rise to a transformation from $G(\Pi)$ to $G(\Pi|_\rho)$. For example, if a clause is satisfied by a restriction ρ, its node will be immediately removed. And even if a clause is not satisfied, its node still might be not used in constructing $G(\Pi|_\rho)$ since e.g. a parent is removed.

An *annotated assignment* is an expression of the form $x_i \stackrel{*}{=} a$ $(1 \leq i \leq n,\ a \in \{0,1\}, * \in \{d,u\})$. Informally, a CDCL solver builds (ordered) restrictions one assignment at a time, and the annotation indicates in what way the assignment is made: 'd' means by a decision, and 'u' means by unit propagation. See Definition 2.8 and Remark 8 below for details about these annotations.

Definition 2.7 A *trail* is an ordered list of annotated assignments in which all variables are again pairwise distinct. A trail acts on clauses, CNFs, etc., 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 $[x_{i_1} \stackrel{*_1}{=} a_1, \ldots, x_{i_s} \stackrel{*_s}{=} a_s]$ $(0 \leq s \leq r)$ denoted by $t[\leq s]$. Λ is the empty trail.

A *state* is a pair $(\mathbb{C}, t)$, where $\mathbb{C}$ is a CNF and t is a trail. The state $(\mathbb{C}, t)$ is *terminal* if either $C|_t \equiv 1$ for all $C \in \mathbb{C}$ or $\mathbb{C}$ 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 $\mathbb{S}_n^{\mathrm{o}} \subset \mathbb{S}_n$ be the set of all nonterminal states.

Remark 6 As unambiguous as Definition 2.7 may seem, it already reflects one important choice, to consider only *positional*[4] solvers, i.e., those that are allowed to carry along only CNFs and trails, but not any other auxiliary information. The only mathematical ramification of this restriction is that we will have to collapse the whole clause learning stage into one step, but that is a sensible thing to do anyway. From the practical perspective, however, this restriction is far from obvious and we will revisit this issue in our concluding remarks (Section 6).

Remark 7 We are now about to describe the core of our (or, for that matter, any other) model, which can be viewed as a *labeled transition system* consisting of the state space $\mathbb{S}_n$ and possible labeled transitions between states. But since this definition is the longest one, we prefer to change gears and *precede* it with some informal remarks rather than give them after the definition.

Proof systems attempting to capture performance of modern CDCL solvers are in general much bulkier than their logical counterparts and are built from several heterogeneous blocks. At the same time, most papers highlight the impact of one or a few of the features, with a varying degrees of nondeterminism allowed, while the features out of focus are treated in often unpredictable and implicit ways. We have

[4] The name is suggested by a similar term "positional strategy" in game theory.

found this state of affairs somewhat impending for the effort of trying to compare different results to each other or to build useful structure around them of the kind existing in "pure" proof complexity. Therefore, we adapt an approach that in a sense is the opposite. Namely, 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. These deviations will take the form of "amendments" forbidding certain forms of behavior or, potentially, allowing for new ones.

Besides this point, there are only few (although sometimes subtle) differences from the previous models, so our description is given more or less matter-of-factly.

Definition 2.8 For a nonterminal state $S = (\mathbb{C}, t) \in \mathbb{S}_n^o$, we define the finite set $\mathrm{Actions}(S)$ and the function $\mathrm{Transition}_S : \mathrm{Actions}(S) \longrightarrow \mathbb{S}_n$; the fact $\mathrm{Transition}_S(A) = S'$ will be usually abbreviated to $S \overset{A}{\Longrightarrow} S'$. Those are described as follows:

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

where the letters D, U, and L naturally stand for *decision*, *unit propagation*, and *learning*.[5]

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

$$(\mathbb{C}, t) \overset{x_i \overset{d}{=} a}{\Longrightarrow} (\mathbb{C}, [t, x_i \overset{d}{=} a]). \tag{3}$$

- $U(S)$ consists of all those assignments $x_i \overset{u}{=} a$ for which $\mathbb{C}|_t$ contains the unit clause x_i^a; the transition function is given by the same formula (3) but with a different annotation:

$$(\mathbb{C}, t) \overset{x_i \overset{u}{=} a}{\Longrightarrow} (\mathbb{C}, [t, x_i \overset{u}{=} a]). \tag{4}$$

- As should be expected, $L(S)$ is the most sophisticated part of the definition (cf. [3, Section 2.3.3]). It consists of clause-trail pairs (C, t^*) where C is a learnable clause and t^* is a prefix of t with the assignments that persist after learning C and backtracking.

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

The exact class of such pairs is given in Definition 2.9. The transition function is then defined naturally:

$$(\mathbb{C}, t) \overset{(C,t^*)}{\Longrightarrow} (\mathbb{C} \cup \{C\}, t^*).$$

Definition 2.9 Given $(\mathbb{C}, t) \in \mathbb{S}_n^o$, the set of learnable clauses from S is defined as follows. Let $t = [x_{i_1} \overset{*_1}{=} a_1, \ldots, x_{i_r} \overset{*_r}{=} a_r]$. 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

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

be the set of all *conflict clauses.*

For $1 \leq k \leq r$, we do the following: if the k-th assignment of t is of the form $x_{i_k} \overset{d}{=} a_k$, then $\mathbb{C}_k(S) \overset{\text{def}}{=} \mathbb{C}_{k+1}(S)$. Otherwise, it is of the form $x_{i_k} \overset{u}{=} a_k$, 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 $C \in \mathbb{C}$ such that $C|_{t[\leq k-1]} = x_{i_k}^{a_k}$ and include into $\mathbb{C}_k(S)$ all the results $\mathrm{Res}(C, D)$. 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 2.10 $D|_t = 0$ *for every* $D \in \mathbb{C}_k(S)$.

Finally, we let

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

and

$$L(S) \overset{\text{def}}{=} \begin{cases} \{(0, \Lambda)\} \text{ if } 0 \in \mathbb{C}(S); \\ \{ (C, t^*) \mid C \in (\mathbb{C}(S) \setminus \mathbb{C}),\ t^* \text{ a prefix of } t \text{ such that } C|_{t^*} \neq 0 \} \text{ otherwise.} \end{cases} \quad (5)$$

Example 1 Consider the scenario in which

$$\mathbb{C} = \{x_1 \vee \overline{x_4},\ \overline{x_3} \vee x_4,\ x_1 \vee x_3 \vee x_4,\ x_1 \vee \overline{x_3} \vee x_4\}$$
$$t = [x_1 \stackrel{d}{=} 0,\ x_4 \stackrel{u}{=} 0,\ x_3 \stackrel{u}{=} 1]$$
$$S = (\mathbb{C}, t).$$

Then

$$\mathbb{C}_4(S) = \{\overline{x_3} \vee x_4,\ x_1 \vee \overline{x_3} \vee x_4\}$$
$$\mathbb{C}_3(S) = \{x_1 \vee x_4\}$$
$$\mathbb{C}_2(S) = \{x_1\}$$
$$\mathbb{C}_1(S) = \mathbb{C}_2(S)$$

so $\mathbb{C}(S) = \{x_1,\ x_1 \vee x_4\}$ and, finally,

$$L(S) = \{(x_1, \Lambda),\ (x_1 \vee x_4, \Lambda),\ (x_1 \vee x_4, (x_1 \stackrel{d}{=} 0))\}.$$

This completes the description of the basic model.

Remark 8 For nearly all modern implementations of CDCL, the annotations are redundant because CDCL solvers typically require unit propagation always to be performed when it is applicable (in our language of amendments, this feature will be called ALWAYS-U). Nevertheless, the presence of annotations makes the basic model flexible enough to carry on various, sometimes subtle, restrictions and extensions. In particular, we consider solvers that are not required to *record* unit propagations as such. This allows for the situation in which $x_i \stackrel{d}{=} a$ *and* $x_i \stackrel{u}{=} a$ are in Actions(S), and the set of learnable clauses is sensible to this.

Remark 9 In certain pathological cases, mostly resulting from neglecting to *do* unit propagation, the set Actions$(\mathbb{C}, t)$ may turn out to be empty even if $(\mathbb{C}, t)$ is nonterminal and $\mathbb{C}$ is contradictory. But for the reasons already discussed above, we prefer to keep the basic model as clean as possible *syntactically*, postponing such considerations for later.

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 $S' \in \text{im}(\text{Transition}_S)$. It is easy to see (by double induction on $(|\mathbb{C}|, n - |t|)$) that Γ_n is acyclic. Moreover, both the set

$\{ (S, A) \mid A \in \text{Actions}(S) \}$ and the function $(S, A) \mapsto \text{Transition}_S(A)$ are polynomial-time[6] computable. These observations motivate the following definition.

Definition 2.11 Given a CNF $\mathbb{C}$, a *partial run* on $\mathbb{C}$ from the state S to the state T is a sequence

$$S = S_0 \overset{A_0}{\Longrightarrow} S_1 \overset{A_1}{\Longrightarrow} \dots S_{L-1} \overset{A_{L-1}}{\Longrightarrow} S_L = T, \tag{6}$$

where $A_k \in \text{Actions}(S_k)$. In other words, a partial run is a path in Γ_n, with annotations restored. A *successful run* is a partial run from $(\mathbb{C}, \Lambda)$ to a terminal state. A *CDCL solver* is a partial function[7] μ on $\mathbb{S}_n^o$ such that $\mu(S) \in \text{Actions}(S)$ whenever $\mu(S)$ is defined. The above remarks imply that when we apply a CDCL solver μ to any initial state $(\mathbb{C}, \Lambda)$, it will always result in a finite sequence like (6), with T being a terminal state (successful run) or such that $\mu(T)$ is undefined (failure).

Remark 10 Theoretical analysis usually deals with *classes* (i.e., sets) of individual solvers rather than with individual implementations, and there might be several different approaches to defining such classes. One might consider for example various complexity restrictions like demanding that μ be polynomial-time computable. But in this paper we are more interested in classes defined by prioritizing and restricting various actions.

Definition 2.12 A *local* class of CDCL solvers is described by a collection of subsets $\text{AllowedActions}(S) \subseteq \text{Actions}(S)$, $S \in \mathbb{S}_n^o$. It consists of all those solvers μ for which $\mu(S) \in \text{AllowedActions}(S)$, 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 $\text{Actions}(S)$ to form $\text{AllowedActions}(S)$. Without further ado, let us give a few examples illustrating how familiar restrictions look in this language. Throughout the description, we fix a nonterminal state $S = (\mathbb{C}, t)$.

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

[6]in the size of the state S, not in n

[7]It is possible for $\text{Actions}(S)$ to be empty, see Remark 9.

ALWAYS-U If $\mathbb{C}|_t$ 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: under the latter, if both a conflict and a unit clause are present, the solver must do conflict analysis while under the former both unit propagation and conflict analysis are permitted.

ALWAYS-R In definition (5) of $L(S)$ we keep only those (C, t^*) for which $t^* = \Lambda$.

NEVER-R In definition (5) of $L(S)$, 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 to have the aforementioned properties. Furthermore, all open questions pertaining to this amendment remain open for either version.

ASSERTING-L In definition (5) of $L(S)$, we shrink $\mathbb{C}(S)\backslash\mathbb{C}$ to $(\bigcup_{k=1}^{s}\mathbb{C}_k(S))\backslash \mathbb{C}$, where $s < r$ is the largest index for which $x_{i_s} = a_s$ is annotated as 'd' in t. This amendment is meaningful (and mostly used) only when combined with ALWAYS-C and ALWAYS-U, in which case we can state expected properties like the fact that every learned clause contains the literal $x_{i_s}^{1-a_s}$ (we do not need this fact in this paper, so we leave its proof to the reader).

DECISION-L In definition (5) of $L(S)$, we shrink $\mathbb{C}(S)\backslash\mathbb{C}$ to $\mathbb{C}_1(S)\backslash\mathbb{C}$. This amendment has appeared in practice as a natural asserting learning scheme. By induction on length of the trail t, it is not hard to see that the learned clause according to this amendment is falsified by just the decisions in t. The clause could consist of a strict subset of those decided variables.

FIRST-L In definition (5) of $L(S)$, we shrink $\mathbb{C}(S) \setminus \mathbb{C}$ to those clauses that are obtained by resolving, in the notation of Definition 2.8, between pairs C and D with $D \in \mathbb{C}$. As noted in the introduction, this is similar to the scheme FirstNewCut [6] but one is not a generalization of the other. FIRST-L is applicable in more settings (FirstNewCut was designed in a setting with mandatory conflict analysis). And the "New" in FirstNewCut refers to its ability to perform more resolutions in order to derive a clause not currently in the formula, which is not modeled by FIRST-L.

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 w.r.t. π that does not appear in t. Note that this amendment does not have any effect upon $U(S)$, and the main technical contributions of our paper can be also phrased as determining under which circumstances this "loophole" can circumvent the severe restriction placed on the set $D(S)$.

WIDTH-w, where w is an integer In definition (5) of $L(S)$, 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.

SPACE-s, where s is an integer If $|\mathbb{C}| \geq s$, then $L(S)$ is entirely removed from Actions(S). This amendment makes sense when accompanied by the possibility to do bookkeeping by removing "unnecessary" clauses. We will briefly discuss positive amendments in Remark 12 below.

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.

Definition 2.13 For a finite set $\mathcal{A}_1, \ldots, \mathcal{A}_r$ of poly-time computable amendments,[8] we let CDCL($\mathcal{A}_1, \ldots, \mathcal{A}_r$) be the (possibly incomplete) proof system whose proofs are those successful runs (6) in which none of the actions A_i is affected by any of the amendments $\mathcal{A}_1, \ldots, \mathcal{A}_r$.

Remark 11 The amendments ALWAYS-C, ALWAYS-U are present in most previous work and, arguably, it is precisely what distinguishes conflict-driven clause learning techniques. Nonetheless, we have decided against including them into the basic model as they may be distracting in theoretical studies focusing on other features; our work is one example.

[8] An amendment is poly-time computable if determining whether an action $\mu(S)$ is in AllowedActions(S) is poly-time decidable given S and $\mu(S)$.

Remark 12 Let us briefly discuss the possibility of *extending* the basic model rather than restricting it. The most substantial deviation would be to forfeit the assumption of positionality (see Remark 6) or, in other words, to allow the solver to carry along more information than just a set of clauses and a trail. Two such examples are *dynamic variable ordering* and *phase saving*. The first is very pertinent to the technical part of our paper, so we defer the corresponding discussion to Section 6.

For positional solvers, extending the basic model amounts to introducing *positive amendments* enlarging the sets Actions(S) instead of decreasing them. Here are a few suggestions we came across during our deliberations.

CLAUSE DELETION For $S = (\mathbb{C}, t) \in \mathbb{S}_n^o$, we add to Actions($S$) all subsets $\mathbb{C}_0 \subseteq \mathbb{C}$. The transition function is obvious:

$$(\mathbb{C}, t) \stackrel{\mathbb{C}_0}{\Longrightarrow} (\mathbb{C}_0, t).$$

This is the space model whose study was initiated in [23], and like in that paper, we do not see compelling reasons to differentiate between original clauses and the learned ones.

MULTI-CLAUSE LEARNING In the definition (5) of $L(S)$, we can allow arbitrary nonempty subsets $\mathbb{C}_0 \subseteq \mathbb{C}(S) \setminus \mathbb{C}$ instead of a single clause C and require that $C|_{t^*} \neq 0$ for *any* $C \in \mathbb{C}$, with the obvious transition

$$(\mathbb{C}, t) \stackrel{(\mathbb{C}_0, t^*)}{\Longrightarrow} (\mathbb{C} \cup \mathbb{C}_0, t^*).$$

Though existing SAT-solver implementions tend not to do this, it is natural to consider when thinking of Pool resolution or RTL proof systems as variants of CDCL (see e.g. [39, 18]).

INCOMPLETE LEARNING In the definition (5) of $L(S)$, we could remove the restriction $C|_{t^*} \neq 0$ on the prefix t^*. This positive amendment could make sense in the absence of ALWAYS-C, that is, if we are prepared for delayed conflict analysis.

In this language, the (nonalgorithmic part of the) main result from [3, 37] can be roughly summarized as

CDCL(ALWAYS-C, ALWAYS-U, ALWAYS-R, ASSERTING-L)
is polynomially equivalent to resolution.[9]

[9]Their result is actually stronger in that the choice of which unit to propagate and which clause to learn can be made adversarially.

The algorithmic part from [3] roughly says that *any* CDCL solver in the associated class, subject to the only condition that the choice of actions from $D(S)$ (when it is allowed by the amendments) is random, polynomially simulates bounded-width resolution[10]. 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 (We took the liberty to remove those amendments that do not appear to be relevant to the question.) Finally, we would like to abstract the "no-restarts" question as

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

where we have again removed all other amendments in the hope that this will make the question more clean mathematically.

2.2. Technical contributions

As they had already been discussed in the introduction, here we formulate our results (in the language just introduced) without additional exposition.

Theorem 2.14 *For any fixed order π on the variables, the system* CDCL(π-D*,* DECISION-L) *is polynomially equivalent to π-ordered resolution.*

Theorem 2.15 *For any fixed order π on the variables, the system* CDCL(π-D*,* FIRST-L) *is polynomially equivalent to general resolution.*

Theorem 2.16 *For any fixed order π on the variables and every $\epsilon > 0$ there exist contradictory CNFs τ_n with $w(\tau_n \vdash 0) = O(1)$ not provable in* CDCL(π-D, WIDTH-$(1-\epsilon)n$).

Finally, let us mention that while CDCL($\mathcal{A}_1, \ldots, \mathcal{A}_r$) can be naturally regarded as a (possibly incomplete) proof system where proofs are efficiently checkable, it need not necessarily be a Hilbert-style proof system, operating with "natural" lines and inference rules. Assume, however, that the set AllowedActions(S) additionally satisfies the following two properties:

[10]That is, has running time $n^{O(w(\tau_n \vdash 0))}$ with high probability, given a contradictory CNF τ_n as an input.

1. whenever AllowedActions(S) $\cap L(S) \neq \emptyset$, it contains an action leading to a state of the form $(\mathbb{C}, \Lambda)$ (i.e, restarts are allowed);
2. (monotonicity) If $S = (\mathbb{C}, t)$, $S' = (\mathbb{C}', t)$ and $\mathbb{C} \subseteq \mathbb{C}'$ then AllowedActions(S) $\cap (D(S) \mathbin{\dot\cup} U(S)) \subseteq$ AllowedActions(S') $\cap (D(S') \mathbin{\dot\cup} U(S'))$.

Then every trail t that appears in a run can always be *recreated*, at a low cost, when it is needed again. Thus, under these restrictions we get a "normal" proof system with nice properties.

Note that property 2) might not hold in the presence of ALWAY-C, ALWAYS-U, and this is the main reason why we do not include them in the basic model for studying CDCL as a proof system. Let us now formulate this system explicitly for the case π-D we are mostly interested in.

Definition 2.17 Fix an order π on the variables. π-*trail resolution* is the following (two-typed) proof system. Its lines are either clauses or trails (where the empty trail is an axiom), and it has the following rules of inference:

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

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

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

where $C|_t = x_i^a$;

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

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

It is straightforward to see that without the unit propagation rule, this is just π-half-ordered resolution.

Then, the main technical part in proving Theorem 2.15 is the following.

Theorem 2.18 *For every fixed order π on the variables, π-trail resolution is polynomially equivalent to general resolution.*

3. $\mathsf{CDCL}(\pi\text{-}\mathsf{D}, \mathsf{DECISION\text{-}L}) =_p \pi$-ordered

In this section we prove Theorem 2.14. The proof is made up of two parts (Theorem 2.6, Theorem 3.4), with half-ordered resolution as the intermediary.

3.1. π-half-ordered $=_p$ π-ordered

Half-ordered resolution trivially p-simulates ordered resolution, so the core of Theorem 2.6 is the other direction. In this section we will depend heavily on resolution graphs (Definition 2.1) and related definitions from Section 2.

Definition 3.1 A resolution refutation Π is *ordered up to k* (with respect to an order π) if it satisfies the property that if any two clauses are resolved on a variable $x_i \in \mathrm{Var}_\pi^k$, then all resolution steps above it are on variables in $\mathrm{Var}_\pi^{\pi(i)-1}$. We note that π-ordered resolution proofs are precisely those that are ordered up to $n-1$.

We now prove the main part of Theorem 2.6, namely that π-ordered resolution p-simulates π-half-ordered resolution.

Proof. *(of theorem 2.6)* Let Π be a π-half-ordered resolution refutation of τ. Without loss of generality, assume that $\pi = \mathrm{id}$ (otherwise rename variables).

We will construct by induction on k ($0 \le k \le n-1$) a half-ordered resolution refutation Π_k of τ, which is ordered up to k. For the base case, let $\Pi_0 = \Pi$. Suppose Π_k has been constructed; without loss of generality we can 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 this set is parent-complete. We claim it is also upward-closed. Indeed, let u be any node in this set (i.e., $c(u) = c_{\Pi_k}(u)$ is k-small) and v be a child of u. Then $c(v)$ is obtained by resolving a variable $x_i \in \mathrm{Var}_\pi^k$ since we disallow weakenings. The fact that Π_k is ordered up to k implies $\mathrm{Var}(c(v)) \subseteq \mathrm{Var}_\pi^{i-1} \subseteq \mathrm{Var}_\pi^k$ (otherwise, some variable in $c(v)$ would have remained unresolved on a path connecting v to the sink by connectedness), thus $c(v)$ is also k-small i.e. v is in this set. Therefore, by induction, any node above u is in the set.

Upward-closedness implies path-completeness (see Remark 3), so by Proposition 2.3, the set $\{v \mid c(v) \text{ is } k\text{-small}\}$ defines a subrefutation

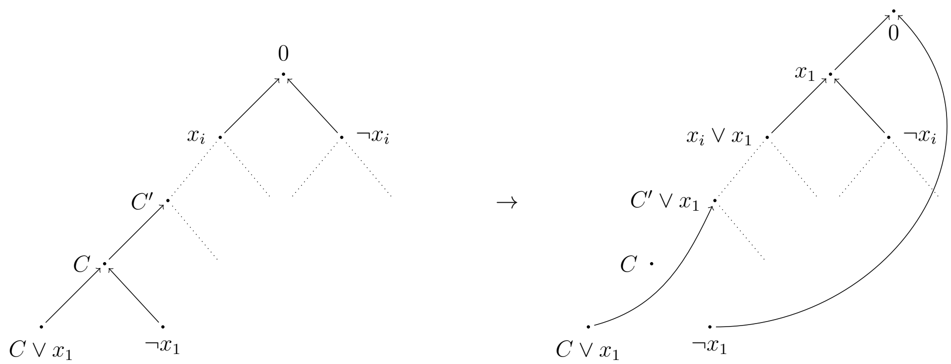

Figure 1: A toy example of the transformation from Π_0 to Π_1

of the clauses labeling the independent set

$$\mathcal{L}_k \stackrel{\text{def}}{=} \min_{\Pi_k}\{v|\ c(v) \text{ is } k\text{-small}\}. \tag{7}$$

Denote $\mathcal{U} := \mathrm{ucl}_{\Pi_k}(\mathcal{L}_k)$ $(= \{v|\ c(v) \text{ is } k\text{-small}\})$ and $\mathcal{D} := \mathrm{dcl}_{\Pi_k}(\mathcal{L}_k)$ where $\mathcal{D}, \mathcal{U}, \mathcal{L}$ stands for *downward*, *upward*, and *layer*, respectively. Then we have the following.

- $\mathcal{U} \cap \mathcal{D} = \mathcal{L}_k$ (from the independence of nodes in $\mathcal{L}_k$);
- $\mathcal{D}$ is also a subproof (by Remark 3 and Proposition 2.3);
- $\Pi_k = \mathcal{U} \cup \mathcal{D}$. To see this, note by connectedness any node v can be connected by a directed path p in Π_k to the unique sink—the empty clause which belongs to $\mathcal{U}$. Consider the first $u \in p$ that is in $\mathcal{U}$. If $u = v$ then $v \in \mathcal{U}$, otherwise $u \in \mathcal{L}_k$ (since $\mathcal{U}$ is path- and parent-complete) and then $v \in \mathcal{D}$.

That is, the "layer" $\mathcal{L}_k$ splits Π_k into two subproofs $\mathcal{U}$, $\mathcal{D}$ and they meet at $\mathcal{L}_k = \min_{\Pi_k}(\mathcal{U}) = \max_{\Pi_k}(\mathcal{D})$. $\mathcal{U}$ contains all nodes labeled by a k-small clause, and $\mathcal{D}$ is the union of $\mathcal{L}_k$ and the set of all nodes whose labeled clause is not k-small. In particular, all axioms are in $\mathcal{D}$, all resolutions in $\mathcal{U}$ are on the variables in Var_π^k and, since Π_k is ordered up to k, all resolutions in $\mathcal{D}$ are on the variables not in Var_π^k.

Define

$$M \stackrel{\text{def}}{=} \min_{\mathcal{D}}\{w|\ c(w) \text{ is the result of resolving two clauses on } x_{k+1}\} \tag{8}$$

where $\min_{\mathcal{D}}$ is taken with respect to the topological order in the proof $\mathcal{D}$ (cf. the last paragraph in Definition 2.1). If M is empty, $\Pi_{k+1} \stackrel{\mathrm{def}}{=} \Pi_k$. Otherwise, suppose $M = \{w_1, \ldots, w_s\}$ (where $w_1, \ldots, w_s$ are independent nodes in $\mathcal{D}$), and define

$$A_i \stackrel{\mathrm{def}}{=} \mathrm{ucl}_{\mathcal{D}}(\{w_i\}). \tag{9}$$

We will eliminate all resolutions on x_{k+1} in $\mathcal{D}$ by the following process; it should be emphasized that *the set of nodes stays the same* during this process. Only the edges and clause-labeling function change. More precisely, we update $\mathcal{D}$ in s rounds, defining π-half-ordered resolution + *weakening* proofs $\mathcal{D}_1, \mathcal{D}_2, \ldots, \mathcal{D}_s$. Initially $\mathcal{D}_0 = \mathcal{D}$, $i = 1$. Let c_{i-1} denote the clause-labeling $c_{\mathcal{D}_{i-1}}$. To define the transition $\mathcal{D}_{i-1} \to \mathcal{D}_i$, we need the following structural properties of $\mathcal{D}_{i-1}$ (that will also be proved by induction simultaneously with the definition).

Claim 3.2 *Let u and v be arbitrary vertices in $V(\mathcal{D})$.*

a. If v is not above u in $\mathcal{D}$, then the same is true in $\mathcal{D}_{i-1}$;

b. $c_{i-1}(v)$ is equal to $c_{\mathcal{D}}(v)$, $c_{\mathcal{D}}(v) \vee x_{k+1}$ or $c_{\mathcal{D}}(v) \vee \overline{x_{k+1}}$;

c. If $v \notin \bigcup_{j=1}^{i-1} A_j$ then $c_{i-1}(v) = c_{\mathcal{D}}(v)$, and $c_{i-1}(v)$ is obtained in $\mathcal{D}_{i-1}$ via application of the same resolution rule as in $\mathcal{D}_i$;

d. $\mathcal{D}_{i-1}$ is a π-half-ordered resolution + weakening proof.

In the base case ($i = 1$), Claim 3.2 holds simply because $\mathcal{D}_0 = \mathcal{D}$.

Let us construct $\mathcal{D}_i$. By Claim 3.2(c), the resolution step at w_i (which is not in $\bigcup_{j=1}^{i-1} A_j$ by independence) is unchanged from $\mathcal{D}$ to $\mathcal{D}_{i-1}$. Assume that it resolves $c_{\mathcal{D}}(w') = B \vee x_{k+1}$ and $c_{\mathcal{D}}(w'') = C \vee \overline{x_{k+1}}$. Since Π_k is 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 $\mathcal{D}$ on variables in Var_π^k. Thus, for all $v \in A_i$, it follows that B is a subclause of $c_{\mathcal{D}}(v)$, and by Claim 3.2(b), we get the following crucial property:

$$\text{For all } v \in A_i,\ B \text{ is a subclause of } c_{i-1}(v). \tag{10}$$

Note that A_i is upward closed in $\mathcal{D}_{i-1}$ by Claim 3.2(a). Accordingly, as the first step, for any $v \notin A_i$ we set $c_i(v) := c_{i-1}(v)$ and do not change its incoming edges.

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

First we set $c_i(w_i) \coloneqq c_{i-1}(w_i) \vee \overline{x_{k+1}}$ (recall that $c_{i-1}(w_i) = c_{\mathcal{D}}(w_i)$ by Claim 3.2(c) and hence does not contain x_{k+1} by (8)), and replace incoming edges by a weakening edge from w''.

For $v \in A_i \setminus \{w_i\}$ (as a reminder, it might be that $v \in \bigcup_{j<i} A_j$), 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 (10), and note for the record that $w' <_{\mathcal{D}} w_i <_{\mathcal{D}} v$.
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 $c_i(v) \coloneqq c_{i-1}(v) \vee \overline{x_{k+1}}$, 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 $c_i(v) \coloneqq c_{i-1}(v) \vee c_i(u)$. 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, $x_{k+1} \notin \{c_{i-1}(u), c_{i-1}(w)\}$. Set $c_i(v) \coloneqq \mathrm{Res}(c_i(u), c_i(w))$ 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 $c_i(w)$. Note that since $\ell > k+1$, this step remains π-half-ordered.

This completes our description of $\mathcal{D}_i$; we have to check Claim 3.2 for it. For (a), note that the only new edges were added in item 1, and see the remark made there. The items (b) and (c) are straightforward. For (d), the only different resolution steps were introduced in item 4; again, see the remark made there.

The next claim summarizes the necessary properties of the end result, $\mathcal{D}_s$.

Claim 3.3

a. *$\mathcal{D}_s$ is a π-half-ordered resolution + weakening proof without resolutions on x_{k+1}.*

b. *If $c_s(v) \neq c_{\mathcal{D}}(v)$ for some $v \in \mathcal{D}$, then there is a vertex w in $\mathrm{dcl}_{\mathcal{D}}(M) \setminus \{M\}$ such that $c_{\mathcal{D}}(v) = Res(c_s(w), c_s(v))$ on x_{k+1}, and this resolution is half-ordered. In fact, w is a parent (in $\mathcal{D}$'s topology) of some node in M.*

Proof.

a. No new resolution on the variable x_{k+1} has been introduced, while all old ones are in $A_1 \cup \cdots \cup A_s$ and thus have been eliminated. The conclusion follows from this observation together with Claim 3.2(d).

b. Suppose $c(v)$ was changed in $\mathcal{D}_{i-1} \to \mathcal{D}_i$ (and hence stayed unchanged afterwards by Claim 3.2(b)) then in particular $v \in A_i$. Set $w := w'$ where w' is the parent of w_i we chose in the paragraph above (10). Note that $c_s(w) = c_{\mathcal{D}}(w)$ since the latter contains the literal (say) x_{k+1}. Then we readily have $c_{\mathcal{D}}(v) = c_{\mathcal{D}_{i-1}}(v) = \text{Res}(B \vee x_{k+1}, c_{\mathcal{D}}(v) \vee \overline{x_{k+1}})$ by (10), and it is half-ordered since B is k-small. ∎

Now to get Π_{k+1}, we try to reconnect $\mathcal{D}_s$ with $\mathcal{U}$ along $\mathcal{L}_k$ (again, their node sets have been unchanged so far), then clear out weakenings. The problem with this approach is the added appearances of x_{k+1}^a (where a may be 0 or 1) in $c_s(v)$ for $v \in \mathcal{L}_k$, as in Claim 3.2(b). We introduce new nodes to deal with them. Namely, let $\mathcal{L}_k^{bad} := \{v \in \mathcal{L}_k \mid c_s(v) \neq c_{\mathcal{D}}(v)\}$, and for each node $v \in \mathcal{L}_k^{bad}$, keeping in mind Claim 3.3(b), create a **new** node denoted by $\widetilde{v}$ labeled with the clause $\text{Res}(c_s(w), c_s(v)) = c_{\mathcal{D}}(v)$. Denote the set of new vertices by $\mathcal{N}$. Define $\widetilde{\Pi}_{k+1}$ to be the result of connecting $\mathcal{D}_s \sqcup \mathcal{N}$ and $\mathcal{U}$ along $(\mathcal{L}_k \backslash \mathcal{L}_k^{bad}) \sqcup \mathcal{N}$.[11]

Before this operation neither $\mathcal{D}_s$ nor $\mathcal{U}$ contained resolutions on x_{k+1}, and hence $\widetilde{\Pi}_{k+1}$ is a half-ordered refutation (with weakenings) that is *ordered up to* $k+1$. Let Π_{k+1} be obtained by contracting[12] all weakening rules in $\widetilde{\Pi}_{k+1}$. Then Π_{k+1} is also half-ordered up to $k+1$ since contracting weakening rules preserves this property. It only remains to analyze the size, $|\Pi_{k+1}|$ (note that *a priori* it can be doubled at every step, which is unacceptable).

Since

$$|\Pi_{k+1}| \leq |\Pi_k| + |\mathcal{L}_k|, \tag{11}$$

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

$$|\text{dcl}_{\Pi_{k+1}}(\mathcal{L}_{k+1})| \leq |\text{dcl}_{\Pi_k}(\mathcal{L}_k)|. \tag{12}$$

[11]For example, suppose $v \in \mathcal{L}_k$ and $c_{\mathcal{D}}(v) = x_1 \vee x_2$ but $c_s(v) = x_1 \vee x_2 \vee \overline{x_{k+1}}$ (assuming $k > 1$). By Claim 3.3(b), there is a vertex w with $c_s(w) \subseteq x_1 \vee x_2 \vee x_{k+1}$ and

$$c(\widetilde{v}) := \text{Res}(c_s(w), c_s(v)) = x_1 \vee x_2 = c_{\mathcal{D}}(v),$$

so we will use $\widetilde{v}$ instead of v in connecting $\mathcal{D}_s$ and $\mathcal{U}$ to construct $\widetilde{\Pi}_{k+1}$.

[12]By contraction here we mean the process implicit in showing that weakening rules can be eliminated in a resolution refutation without increasing its size.

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

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 $(v \mapsto v^*)\,|_{\mathrm{dcl}_{\Pi_{k+1}}(\mathcal{L}_{k+1})}$ (which is injective by definition) is the desired injection. We have to check that its image is contained in $\mathrm{dcl}_{\Pi_k}(\mathcal{L}_k)$.

Fix $v \in \mathrm{dcl}_{\Pi_{k+1}}(\mathcal{L}_{k+1})$. Then *either v is an axiom or both its parents are not* $(k+1)$*-small* (by (7)). By the above mentioned facts about the contraction $\widetilde{\Pi}_{k+1} \to \Pi_{k+1}$, this property is inherited by v^*. In particular, $v^* \notin \mathcal{N}$ (the set of the newly added nodes when constructing Π_k) because all nodes in this set have at least one $(k+1)$-small parent (the w node in Claim 3.3(b)). Finally, since the corresponding clauses in $\mathcal{D}$ and $\mathcal{D}_s$ differ only in the variable x_{k+1}, v^* cannot be in U, for the same reason (recall that all axioms are in $\mathcal{D}$). Hence $v^* \in V(\mathcal{D}_s) = V(\mathcal{D}) = \mathrm{dcl}_{\Pi_k}(\mathcal{L}_k)$.

Having thus proved (12), we conclude by the obvious induction that $|\mathcal{L}_k| \leq |\mathrm{dcl}_{\Pi_k}(\mathcal{L}_k)| \leq |\mathrm{dcl}_{\Pi_0}(\mathcal{L}_0)| \leq |\Pi|$. Then (11) implies $|\Pi_{n-1}| \leq n|\Pi|$, as desired.■

3.2. π-half-ordered $=_p$ CDCL(π-D, DECISION-L)

In this section, we prove the following theorem.

Theorem 3.4 *The systems* CDCL(π-D, DECISION-L) *and π-half-ordered resolution are p-equivalent.*

One direction is almost trivial.

Proposition 3.5 CDCL(π-D, DECISION-L) *p-simulates π-half-ordered resolution.*

Proof. As usual, assume $\pi = \mathrm{id}$. Suppose $C \vee D = \mathrm{Res}(C \vee x_i,\ D \vee \overline{x_i})$ is any half-ordered resolution, and without loss of generality assume

C is i-small. It is enough to present a partial run from (τ, Λ) to $(\tau \cup \{C \vee D\}, \Lambda)$ of length at most $n+1$, where τ is any clause set containing $C \vee x_i$ and $D \vee \overline{x_i}$.

Let x_j be the largest variable in C (thus $j < i$). Consider a trail of the form

$$t = [x_1 \stackrel{d}{=} a_1, \ldots, x_j \stackrel{d}{=} a_j, x_i \stackrel{u}{=} 1, x_{j+1} \stackrel{d}{=} a_{j+1}, \ldots, x_{i-1} \stackrel{d}{=} a_{i-1}, x_{i+1} \stackrel{d}{=} a_{i+1}, \ldots, x_n \stackrel{d}{=} a_n]$$

such that $(C \vee D)|_t = 0$. By definition, $t[l] \in \text{AllowedActions}((\tau, t[\leq l-1]))$ for all $l \neq j+1$. But since C is i-small, $(C \vee x_i)|_{t[\leq j]} = x_i$ and thus $x_i \stackrel{u}{=} 1 \in \text{AllowedActions}((\tau, t[\leq j]))$ as well. Therefore,

$$(\tau, \Lambda) \xRightarrow{t[1]} (\tau, t[\leq 1]) \xRightarrow{t[2]} (\tau, t[\leq 2]) \ldots \xRightarrow{t[n]} (\tau, t)$$

is a partial run from (τ, Λ) to (τ, t). It now suffices to show $(\tau \cup \{C \vee D\}, \Lambda) \in L((\tau, t))$. This follows by verifying Definition 2.8 directly: $(D \vee \overline{x_i})|_t = 0$ so $D \vee \overline{x_i} \in \mathbb{C}_{n+1}((\tau, t))$. For $j' > j+1$, the assignment $t[j']$ is a decision, so $D \vee \overline{x_i} \in \mathbb{C}_{j+2}((\tau, t))$. Since $(C \vee x_i)|_{t[\leq j]} = x_i$, $C \vee D = \text{Res}(C \vee x_i, D \vee \overline{x_i}) \in \mathbb{C}_{j+1}((\tau, t))$. Finally, for $j' \leq j$, $t[j']$ is a decision, so $C \vee D \in \mathbb{C}_1(\tau, t)$ and $(\tau \cup \{C \vee D\}, \Lambda) \in \text{AllowedActions}((\tau, t))$.■

The other direction of Theorem 3.4 is less obvious. We begin with some additional notation.

Previous works describe standard learning schemes like DECISION-L with respect to so-called *trivial resolution* on a set of particular clauses (e.g., in [37, 6]). We can recast this notion in our model by the following lemma. Let

$$D \circ^x C \stackrel{\text{def}}{=} \begin{cases} \text{Res}(D, C) & \text{if } C \text{ and } D \text{ are resolvable on } x \\ D & \text{otherwise (“null case”)} \end{cases}$$

and we extend this definition by left associativity:

$$C_0 \circ^{x_{i_1}} C_1 \circ^{x_{i_2}} \cdots \circ^{x_{i_k}} C_k \stackrel{\text{def}}{=} (\ldots(C_0 \circ^{x_{i_1}} C_1) \circ^{x_{i_2}} \ldots) \circ^{x_{i_k}} C_k.$$

Note if x_{i_j} appears maximally in C_j (according to π) for each $j \in [k]$, then all the resolutions are π-half-ordered.

Lemma 3.6 *Assume, in the notation of in Definition 2.8, a clause D is learned from state $S = (\mathbb{C}, t = [y_1 \stackrel{*_1}{=} a_1, \ldots, y_r \stackrel{*_r}{=} a_r])$.*

Assume $D \in \mathbb{C}_j(S)$ for some $j \in [r+1]$. Then there exist $k \leq r$, indices $j \leq i_1 < \cdots < i_k \leq r$ and clauses $C_1, \ldots, C_{k+1} \in \mathbb{C}$ such that $*_{i_1} = \ldots = *_{i_k} = u$ (i.e. the indices correspond to some unit propagations in t) and the following properties hold.

1. $C_{k+1}|_t = 0$.
2. $C_\nu|_{t[\leq i_\nu - 1]} = y_{i_\nu}^{a_{i_\nu}}$ *for* $\nu \in [k]$.
3. $D = C_{k+1} \circ^{y_{i_k}} C_k \cdots \circ^{y_{i_1}} C_1$ *where all operators are not null.*
4. *For any $\ell \in [j, r]$ with $*_\ell = u$, if variable y_ℓ appears in C_ν for some $\nu \in [k+1]$ then $\ell \in \{i_1, \ldots, i_k\}$.*

Proof. This is by directly tracing Definition 2.8. Suppose $D \in \mathbb{C}_j$, we use reverse induction on j to define the desired clauses and indices. If $j = r+1$, let $k = 0$ and $C_{k+1} = D$ (the conflict clause). If $j \leq r$, either $D \in \mathbb{C}_{j+1}(S)$, or there are clauses $D' \in \mathbb{C}_{j+1}(S)$, $C \in \mathbb{C}$ such that $C|_{t[\leq j-1]} = y_j^{a_j}$ and $D = \text{Res}(D', C)$ on y_j. In the first case, the clauses and indices are the same as for $D \in \mathbb{C}_{j+1}(S)$, and in the second case, enlarge the index list by adding j and the clause list by appending C to be the first. Items 1,2,3 follow immediately by this definition.

Now we can represent the learning of D as a sequence of clauses $X_{r+1}(= C_{k+1}) \xrightarrow{\text{step } r} X_r \rightarrow \ldots \xrightarrow{\text{step } 1} X_j(= D)$ where at each step $i_\mu \in \{i_1, ..., i_k\}$ a resolution $X_{i_\mu} = \text{Res}(X_{i_\mu+1}, C_\mu)$ happens, and at each step $a \in [j, r] \backslash \{i_1, ..., i_k\}$, $X_a = X_{a+1}$. To prove item 4, assume $\ell \in [r]$, $\nu \in [k+1]$ satisfy the assumption there. Let $i_{k+1} := r+1$ for convenience. First, it cannot be that $i_\nu < \ell$ since $\text{Var}(C_\nu) \subset \text{Var}(t[\leq i_\nu])$ but $y_\ell \notin \text{Var}(t[\leq i_\nu])$. So assume $i_\nu > \ell$ (if $i_\nu = \ell$ then we are done). By items 1 and 2, for any ν' with $i_{\nu'} > \ell$, $C_{\nu'}$ does not contain $y_\ell^{a_\ell}$; so in particular, $y_\ell^{1-a_\ell} \in C_\nu$ by assumption. Then $y_\ell^{1-a_\ell} \in X_{i_\nu} = (C_{k+1} \circ^{y_{i_k}} C_k \cdots \circ^{y_{i_\nu}} C_\nu)$ (where all resolutions are not null), and the literal $y_\ell^{1-a_\ell}$ appears in $X_{i_\nu} ... X_{\ell+1}$. Hence, a resolution step on y_ℓ must happen at step ℓ, which means $\ell \in \{i_1, ..., i_k\}$.■

In short, C_{k+1} is a conflict clause and the other C_ν's are clauses in $\mathbb{C}$ chosen to do resolutions while backtracking in a learning step. These clauses are not necessarily unique, but we fix a choice arbitrarily.

Example 2 Let us consider the same scenario as in Example 1 with $\pi = \text{id}$ (so that t is a legitimate trail). One way to learn the clause

$x_1 \in \mathbb{C}_1(S)$ is to take the clauses

$$C_1 = x_1 \vee \overline{x_4}$$
$$C_2 = x_1 \vee x_3 \vee x_4$$
$$C_3 = x_1 \vee \overline{x_3} \vee x_4$$

so that $(C_3 \circ^{x_3} C_2) \circ^{x_4} C_1 = x_1$.

The following Proposition 3.7 will complete the proof of Theorem 3.4, and with Theorem 2.6 this completes the proof of Theorem 2.14.

Proposition 3.7 *π-half-ordered resolution p-simulates* $\mathsf{CDCL}(\pi\text{-}\mathsf{D}, \mathsf{DECISION\text{-}L})$.

Proof. Fix a successful run in $\mathsf{CDCL}(\pi\text{-}\mathsf{D}, \mathsf{DECISION\text{-}L})$. Since the clause set only changes after a learning step, it suffices to show that for each learning step $S = (\mathbb{C}, t) \overset{(D,t^*)}{\Longrightarrow} (\mathbb{C} \cup \{D\}, t^*)$, there is a short half-ordered resolution proof of D from $\mathbb{C}$. Suppose $t = [y_1 \overset{*_1}{=} a_1, \ldots, y_r \overset{*_r}{=} a_r]$ and assume $\pi = \mathrm{id}$, as usual. Fix the clauses C_ν for $\nu \in [k+1]$ and the set $\{i_1, ..., i_k\}$ as in Lemma 3.6 where we take $j = 1$ (that is, $D \in \mathbb{C}_1(S)$) due to $\mathsf{DECISION\text{-}L}$.

Recall that

$$D = C_{k+1} \circ^{y_{i_k}} C_k \cdots \circ^{y_{i_1}} C_1. \tag{13}$$

The high-level idea is the following. The sequence of resolutions (13) is not all half-ordered only if some $y_{i_\nu}^{a_{i_\nu}}$ is not the largest in C_ν (which may happen since the assignments in t need not necessarily respect the order π). Our goal is thus to replace in (13), this time going from right to left, each clause C_ν for $\nu \in [k]$ by a clause C'_ν in which y_{i_ν} is the largest. This will give the desired half-ordered resolution.

First, let $C'_1 = C_1$. For $\nu \in [2, k+1]$, let

$$C'_\nu \overset{\mathrm{def}}{=} C_\nu \circ^{y_{i_{\nu-1}}} C'_{\nu-1} \cdots \circ^{y_{i_1}} C'_1 \tag{14}$$

where this time some operators may be null.

It is immediate from (14) and Lemma 3.6(2) that

$$y_{i_\nu}^{a_\nu} \in C'_\nu \text{ for all } \nu \in [k] \tag{15}$$

and

$$C'_\nu \subseteq \bigcup_{\mu=1}^{\nu} C_\mu \text{ for all } \nu \in [k+1] \text{ (by induction on } \nu). \tag{16}$$

Lemma 3.8 *For any $\mu < \nu$, variable y_{i_μ} does not appear in the clause $C_\nu \circ^{y_{i_{\nu-1}}} C'_{\nu-1} \cdots \circ^{y_{i_\mu}} C'_\mu$.*

Proof. We first prove the fact that, for all $\nu \in [k+1]$ and $\mu \leq \nu$,

$$(C_\nu \circ^{y_{i_{\nu-1}}} C'_{\nu-1} \cdots \circ^{y_{i_\mu}} C'_\mu)|_{t[\leq i_\nu - 1]} = y_{i_\nu}^{a_{i_\nu}} \quad \text{where } y_{i_{k+1}}^{a_{i_{k+1}}} := 0.$$

For this we use double induction, first on ν and then on $\mu = \nu \ldots 1$. For $\mu = \nu$, this is Lemma 3.6(2) (and Lemma 3.6(1) when $\mu = \nu = k+1$). For $\mu < \nu$ let $E \stackrel{\text{def}}{=} (C_\nu \circ^{y_{i_{\nu-1}}} C'_{\nu-1} \cdots \circ^{y_{i_{\mu+1}}} C'_{\mu+1})$; we have to prove that $(E \circ^{y_{i_\mu}} C'_\mu)|_{t[\leq i_\nu - 1]} = y_{i_\nu}^{a_{i_\nu}}$ from $E|_{t[\leq i_\nu - 1]} = y_{i_\nu}^{a_{i_\nu}}$. We can assume without loss of generality that this operator is not null, and then note $C'_\mu\ |_{t[\leq i_\mu - 1]} = y_{i_\mu}^{a_\mu}$ by the inductive assumption applied to the pair $\nu := \mu,\ \mu := 1$.

Now we prove the lemma. Again let $E = C_\nu \circ^{y_{i_{\nu-1}}} C'_{\nu-1} \cdots \circ^{y_{i_{\mu+1}}} C'_{\mu+1}$. Note that $y_{i_\mu}^{a_{i_u}} \in C'_{i_\mu}$ by (15) so $y_{i_\mu}^{a_{i_\mu}} \notin E$ (otherwise $E\ |_{t[\leq i_\nu - 1]} = 1$, contradicting the above fact), and the two clauses are consistent on other variables in C'_μ (according to $t[\leq i_\mu - 1]$). Then a simple case analysis on whether or not $E \circ^{y_{i_\mu}} C'_{i_\mu}$ is a null operator shows that the result does not contain y_{i_μ} or $\overline{y_{i_\mu}}$.■

Example 3 Considering the same clauses as in Example 2, note that the resolution $C_3 \circ^{x_3} C_2$ is not (id)-half-ordered. The derivation from the above lemma would yield the clauses

$$\begin{aligned} C'_1 &= C_1 \\ C'_2 &= C_2 \circ^{x_4} C'_1 = x_1 \vee x_3 \\ C'_3 &= (C_3 \circ^{x_3} C'_2) \circ^{x_4} C'_1 = x_1 \end{aligned}$$

where all resolutions are now half-ordered.

We now complete the proof of Proposition 3.7. By Lemma 3.8, the variable y_{i_μ} does not appear in $C_\nu \circ^{y_{i_{\nu-1}}} C'_{\nu-1} \cdots \circ^{y_{i_\mu}} C'_\mu$ ($\mu < \nu$). Also, it does not appear in $C_{\mu-1}, \ldots, C_1$ (by Lemma 3.6(2)) and thus not in $C'_{\mu-1}, \ldots, C'_1$ (by (16)). Hence it does not appear in C'_ν and we arrive at the following strengthening of (16):

$$\forall \nu \in [k+1],\ C'_\nu \subseteq (\bigcup_{\mu=1}^{\nu} C_\mu) \backslash (\bigcup_{\mu=1}^{\nu-1} \{y_{i_\mu}, \overline{y_{i_\mu}}\}). \tag{17}$$

By Lemma 3.6(4) and the fact that $j = 1$ (here we use $\mathsf{DECISION\text{-}L}$), (17) means any variable different from y_{i_ν} in C'_ν is labeled as d in $t_{[\leq i_\nu - 1]}$. This implies y_{i_ν} ($\nu \in [k]$) is maximal in C'_ν since we are in π-D. Thus for all $\nu \in [k+1]$ the sequence $C_\nu \circ^{y_{i_{\nu-1}}} C'_{\nu-1} \cdots \circ^{y_{i_1}} C'_1$ is half-ordered. Taken together, these sequences yield a half-ordered derivation of C'_{k+1} with $O(k^2)$ steps in total.

Finally, by (17) $C'_{k+1} \subseteq (\bigcup_{\mu=1}^{k+1} C_\mu) \backslash (\bigcup_{\mu=1}^{k} \{y_{i_\mu}, \overline{y_{i_\mu}}\}) \subset D$ where the latter inclusion follows by Lemma 3.6(3). This suffices for proving the proposition since the weakening rule is admissible in π-half-ordered resolution.■

4. $\mathsf{CDCL}(\pi\text{-}\mathsf{D}, \mathsf{FIRST\text{-}L}) =_p$ resolution

In this section we prove Theorem 2.15. We first show that $\mathsf{CDCL}(\pi\text{-}\mathsf{D}, \mathsf{FIRST\text{-}L})$ and π-trail resolution (see Definition 2.17) are p-equivalent and then prove size upper bounds for π-trail resolution.

4.1. π-trail resolution $=_p$ $\mathsf{CDCL}(\pi\text{-}\mathsf{D}, \mathsf{FIRST\text{-}L})$

Theorem 4.1 *For any fixed order π, the systems $\mathsf{CDCL}(\pi\text{-}\mathsf{D})$, $\mathsf{CDCL}(\pi\text{-}\mathsf{D}, \mathsf{FIRST\text{-}L})$ and π-trail resolution are p-equivalent.*

Proof. Let Π be a π-trail resolution refutation of a contradictory CNF τ. We simulate Π step-by-step in $\mathsf{CDCL}(\pi\text{-}\mathsf{D}, \mathsf{FIRST\text{-}L})$ by directly deriving each clause in Π. Suppose we have arrived at a state $(\mathbb{C}, \Lambda)$, where $\mathbb{C}$ contains both premises in the inference

$$\frac{C \vee x_i^a \qquad D \vee x_i^{1-a} \qquad t}{C \vee D}, \tag{18}$$

as well as all preceding clauses, and assume that all variables in C appear before x_i in t. Let $t = [x_{j_1} \stackrel{*_1}{=} a_1, \ldots, x_{j_r} \stackrel{*_r}{=} a_r, x_i \stackrel{*}{=} a, \ldots]$ and (for ease of notation) $t_s \stackrel{\text{def}}{=} t[\leq s]$. To derive $C \vee D$, we first build the trail t_r; note that since t might be derived in Π using the Unit Propagation rule, the sequence $j_1, \ldots, j_r$ need not necessarily be π-increasing.

We build the trail t_r simply by performing the corresponding actions in $\mathsf{CDCL}(\pi\text{-}\mathsf{D}, \mathsf{FIRST\text{-}L})$ for decisions and unit propagations. By induction, assume that we have already built t_{s-1}, $s \le r$. If $*_s = d$ then x_{j_s} is the smallest variable according to π that is not in t_{s-1}, so by definition $x_{j_s} \stackrel{d}{=} a_s \in D((\mathbb{C}, t_{s-1}))$. In the case of the Unit Propagation rule ($*_s = u$), there is a clause E in Π preceding (18) such that $E|_{t_{s-1}} = x_{s-1}^{a_{s-1}}$. Since $E \in \mathbb{C}$ by assumption, $x_{j_s} \stackrel{u}{=} a_{j_s} \in U((\mathbb{C}, t_{s-1}))$.

Next, we build $[t_r, x_i \stackrel{u}{=} a]$ from t_r (note that it is different from t_{r+1} if $* = d$), which is possible since $C \vee x_i^a \in \mathbb{C}$ by our assumption. Then we further extend $[t_r, x_i \stackrel{u}{=} a]$ by making decisions in π-ascending order on the remaining variables $\{x_1, ..., x_n\} \setminus (\mathrm{Var}(t_s) \cup \{x_i\})$ until $D \vee x_i^{1-a}$ becomes a conflict clause. Denote the resulting state by $S = (\mathbb{C}, t')$.

Since all assignments after x_i in t' are decisions, $D \vee x_i^{1-a} \in \mathbb{C}_{r+2}(S)$, in the notation of Definition 2.8. Therefore, $C \vee D \in \mathbb{C}_{r+1}(S)$, and hence $(C \vee D, \Lambda)$ is in AllowedActions(S) even in the presence of FIRST-L. Induction completes the simulation.

The other direction is more straightforward: π-trail resolution p-simulates CDCL(π-D) by design. Whenever a run arrives at a state $(\mathbb{C}, t)$, we infer in π-trail resolution all clauses $C \in \mathbb{C}$ as well as all prefixes of t, including t itself. More specifically, for a transition $(\mathbb{C}, t) \stackrel{A}{\Longrightarrow} (\mathbb{C}', t')$, if A is a decision action or a unit propagation action, then we can derive prefixes of t' using the Decision rule and the Unit propagation rule, respectively. If A is a learning action, then it suffices to make the following simple observation: by construction, for any $\gamma \in [|t|]$, the clauses in $\mathbb{C}_\gamma((\mathbb{C}, t))$ can be derived from clauses in $\mathbb{C}$ and $\mathbb{C}_{\gamma+1}((\mathbb{C}, t))$ using the Learning rule with the trail t.

It is easy to see that both simulations increase size by at most a multiplicative factor n.■

4.2. π-trail resolution $=_p$ resolution

It remains to prove that π-trail resolution simulates resolution. This is the interesting direction of Theorem 2.18 and follows from Theorem 4.8 below.

Throughout this section, assume that $\pi = \mathrm{id}$. We first introduce operators for *lifting* π-trail resolution proofs to include appearances of the literal x_1 and *deleting variables* from resolution refutations, both of which we use extensively in the proof of Theorem 4.8.

The lifting operator is primarily a bookkeeping mechanism for managing auxiliary appearances of the literal x_1 in proofs.

Definition 4.2 Let ψ and τ be CNFs such that $x_1 \notin \mathrm{Var}(\psi)$ and for each $C \in \psi$, τ contains either C or $C \vee x_1$. For $C \in \psi$, define $\mathrm{Lift}_\tau(C)$ to be C if $C \in \tau$, and $C \vee x_1$ otherwise. For a π-trail resolution proof Π from ψ define $\mathrm{Lift}_\tau(\Pi)$ to be the π-trail resolution proof resulting from the following operations on Π.

- Add the derivation of $[x_1 \stackrel{d}{=} 0]$ by the Decision rule to the beginning of Π.
- Replace each trail t in Π with $[x_1 \stackrel{d}{=} 0, t]$.
- Replace each axiom A appearing in Π with $\mathrm{Lift}_\tau(A)$ and then let the added appearances of x_1 be naturally inherited throughout the clauses of Π.

It is straightforward to verify that $\mathrm{Lift}_\tau(\Pi)$ is a π-trail resolution proof and if Π derives C from ψ then $\mathrm{Lift}_\tau(\Pi)$ derives C or $C \vee x_1$ from τ. Note also that this is only possible because x_1 is the smallest variable according to π and hence does not interfere with the Learning rule. In the proof of Theorem 4.8, we will want to construct Π but will only be able to derive clauses in τ, so we construct $\mathrm{Lift}_\tau(\Pi)$ instead and then manage the additional appearances of x_1.

The second operator, *variable deletion*, is an analog of restriction for sets of variables (as opposed to assignments). Let $S \subseteq V$ be a set of variables. For a clause C, let $\mathrm{Del}_S(C)$ denote the result of removing from C all literals whose underlying variables are in S. For a CNF τ, define $\mathrm{Del}_S(\tau) \stackrel{\mathrm{def}}{=} \{\mathrm{Del}_S(C) : C \in \tau\} \setminus \{0\}$. Here we see the first interesting feature of variable deletion, namely that we ignore clauses that become 0 after removing variables from S. But, as we show below, if τ is contradictory and the subset S is proper then $\mathrm{Del}_S(\tau)$ is also contradictory. This is not true in general for $\tau|_\rho \setminus \{0\}$ of course.

The action of variable deletion on refutations will be described in Definition 4.3. It is presented as a (linear time) algorithm that operates on the underlying resolution graph as its input, by recursively changing edges and clauses while nodes keep their identity (although some may be deleted). This is similar to the approach we took in Section 3.1. In order to more easily keep the node structure fixed, the algorithm first produces a proof in the subsystem of resolution + weakening in which all applications of the weakening rule are dummy (that is, are of the form $\frac{C}{C}$). We call proofs in this system *generalized resolution proofs*. We further emphasize that variable deletion is defined only on connected refutations, as connectedness is necessary for the output to be a refutation (cf. Claim 4.4(1)). Consequently, we ensure in the proof of Theorem 4.8 that we only apply it to connected refutations.

In the following, recall $c_\Pi(v)$ denotes the corresponding clause at node v in a proof Π.

Definition 4.3 Let Π be a **connected** resolution refutation of τ and let S be a **proper** subset of $\mathrm{Var}(\Pi)$. Let Γ be the generalized resolution refutation of $\mathrm{Del}_S(\tau)$ whose resolution graph is output by the algorithm below. The resolution refutation $\mathrm{Del}_S(\Pi)$ is the result of contracting dummy applications of the weakening rule in Γ.

Deletion Algorithm

1. For each axiom node v, set $c(v) \coloneqq \mathrm{Del}_S(c_\Pi(v))$. If $c(v)$ becomes 0 (that is, when $\mathrm{Var}(c_\Pi(v)) \subseteq S$), delete it.
2. Processing nodes in topological order, let v be a resolution node and let v_1, v_2 be its parents.
   (a) If *both* v_1 and v_2 were previously deleted, delete v as well.
   (b) If only one of them was deleted or none was deleted but $c(v_1), c(v_2)$ are no longer resolvable, then one of them, say, $c(v_1)$ is a subclause of $\mathrm{Del}_S(c(v))$ (we will see this in Claim 4.4). Set $c(v) \coloneqq c(v_1)$, and replace incoming edges with a dummy weakening edge from v_1.
   (c) If both v_1 and v_2 survived and $c(v_1), c(v_2)$ are resolvable, set $c(v) \coloneqq \mathrm{Res}(c(v_1), c(v_2))$.

After processing the root v of Π, output the current downward-closure of v.

We claim that this algorithm is well-defined (that is, the condition in step 2b is always met) and that the root vertex v is not deleted and $c(v) = 0$ (that is, the algorithm produces a generalized resolution refutation of $\mathrm{Del}_S(\tau)$). Both statements are immediate corollaries of the following claim.

Claim 4.4

1. *A vertex v is deleted if and only if for every axiom node $w \in$ $dcl_\Pi(v)$ it holds that* $\mathrm{Var}(c_\Pi(v)) \subseteq S$. *In particular:*

- *The root vertex is not deleted (recall that* Π *is connected);*
- *If* v *is deleted then* $\mathrm{Var}(c_\Pi(v)) \subseteq S$.

2. *For every remaining vertex* v*,* $c(v)$ *is a subclause of* $\mathrm{Del}_S(c_\Pi(v))$.
3. *In the situation of step 2b, there indeed exists* v_i *such that* $c(v_i)$ *is a subclause of* $\mathrm{Del}_S(c_\Pi(v))$.

Proof. These are proved by induction, simultaneously with the construction. In the base case, axioms, the three items clearly hold. In the inductive step, we prove them by analyzing each case in Deletion Algorithm. The only interesting case is step 2b. If precisely one of the two vertices (say, v_2) was deleted, then $\mathrm{Var}(c_\Pi(v_2)) \subseteq S$ by Claim 4.4(1) and hence $c_\Pi(v)$ was obtained by resolving on a variable x_i in S. Applying Claim 4.4(2) to the other parent v_1, we see that $c(v_1)$ is a subclause of $\mathrm{Del}_S(c_\Pi(v_1))$ which in turn is a subclause of $\mathrm{Del}_S(c_\Pi(v))$ since $x_i \in S$. Similarly, if both parents of v are alive but become non-resolvable, then by Claim 4.4(2) the resolved variable is no longer in one of the parents (say v_1) and $c(v_1)$ is a subclause of $\mathrm{Del}_S(c_\Pi(v))$.■

One key difference between variable deletion and restriction is that $\Pi|_\rho$ may be trivial, in the sense that it is a single empty clause, while $\mathrm{Del}_{\mathrm{Var}(\rho)}(\Pi)$ is not. As a simple example, consider the CNF $\{x_1, \overline{x_1} \vee x_2, \overline{x_2}\}$ and the refutation

$$\cfrac{\cfrac{x_1 \qquad \overline{x_1} \vee x_2}{x_2} \qquad \overline{x_2}}{0}$$

If $\rho = \{x_1 = 0\}$, then $\Pi|_\rho$ is trivial, whereas $\mathrm{Del}_{\{x_1\}}(\Pi)$ is

$$\frac{x_2 \qquad \overline{x_2}}{0}$$

The final property of $\mathrm{Del}_S(\Pi)$ is that its size can be characterized with respect to the relationship between Π and S. This allows us to "slough off" parts of the Π that we might have already seen before.

Lemma 4.5 *Let* Π *be a connected resolution refutation and let* $S \subsetneq \mathrm{Var}(\Pi)$*. Let* t *denote the number of resolution steps* $Res(C, D)$ *in* Π *on variables in* S*. Then*

$$|\mathrm{Del}_S(\Pi)| \leq |\Pi| - t.$$

Proof. By Claim 4.4(2), all remaining resolution steps 2c are on variables that do not belong to S.■

Remark 13 (*Restriction as intersection, deletion as projection.*) If we view a clause C semantically as the set $C^{-1}(0) \subset \{0,1\}^n$, then the restriction operator (say by ρ) on any clause C means to take *intersection* with the subcube $\rho^{-1}(1)$: $C^{-1}(0) \to C^{-1}(0) \cap \rho^{-1}(1)$; while the deletion operator $\mathrm{Del}_S(\cdot)$ corresponds to the *projection* $\{0,1\}^n \to \{0,1\}^{S^c}$ induced on $C^{-1}(0)$.

We now have sufficient machinery to prove Theorem 4.8. As is sometimes useful, the simulation we define is more ambitious than necessary. Rather than outputting a refutation, it outputs a proof that derives all literals (as unit clauses) appearing in the input. The motivation for this is twofold. First, unit clauses make π-trail resolution significantly more powerful because they grant more control over the trails that can be derived. In particular, if all literals appearing in a refutation Π have been derived, then Π can be simulated in $n|\Pi|$ steps by directly simulating each resolution appearing in it. Second, in reference to the deletion operator, all clauses of $\mathrm{Del}_S(\tau)$ can be derived using clauses of τ and unit clauses x^0 and x^1 for $x \in S$.

Our simulation algorithm is based on the obvious restrict-and-branch method, by which one recurses on $\Pi|_{\{x_i=0\}}$ and $\Pi|_{\{x_i=1\}}$, lifts the resulting proofs to have axioms in τ, and then derives 0 (if it has not been derived already) by resolving the unit clauses x_i and $\overline{x_i}$. The clear issue with this approach is that we cannot afford to recurse on *both* restricted proofs: there are parts of Π that are "double counted" as a consequence of its DAG structure and the size may blow up. But recursing on just $\Pi|_{\{x_i=0\}}$ may ignore relevant parts of Π, namely those resolutions on variables not even appearing in $\Pi|_{\{x_i=0\}}$. This is the purpose of the deletion operator. The refutation $\mathrm{Del}_{\mathrm{Var}(\Pi|_{\{x_i=0\}})}(\Pi)$ is a refutation with resolutions that correspond to resolutions in Π but not in $\Pi|_{\{x_i=0\}}$, so we can recurse on it without worrying about this double counting issue. This can be iterated so that we eventually see all literals appearing in Π without considering a particular resolution more than once. So an incomplete but instructive outline of our algorithm is this: recurse on $\Pi|_{\{x_i=0\}}$ and lift the proof to axioms of τ, iterate the deletion operator to derive all literals appearing in Π with possible additional appearances of x_i, and then simulate $\Pi|_{\{x_i=1\}}$ directly to derive $\overline{x_i}$ and remove all additional appearances of x_i.

Before we finally state and prove Theorem 4.8, we present two simple lemmas that are factored out of the proof to simplify its presentation. The first essentially states that a variable in a connected refutation must play a nontrivial role, which intuitively should be true if we want to derive its corresponding literals. The second tells us that once we can directly simulate a connected π-trail refutation, we can also directly simulate a proof of all its literals; this is essentially a stronger version of the observation in the previous paragraphs that is more suited to the goal of deriving all literals.

Lemma 4.6 *Let* Π *be a connected resolution refutation of* τ *such that* $x \in \mathrm{Var}(\Pi)$ *and let* Π' *be the downward closure of any appearance of* 0 *in* $\Pi|_{\{x=a\}}$. *Then there is a clause* $C \in \tau$ *that contains* x^{1-a} *and appears restricted in* Π'.

Proof. Suppose for contradiction that there is no such clause. Then all axioms in Π' are axioms in Π not containing the variable x, so in the standard definition of restriction no edges are contracted and $G(\Pi')$ is a downward-closed subgraph of $G(\Pi)$ with identical labels. Since Π is connected it has a unique appearance of 0 (otherwise, 0 would be the premise of some resolution step that is impossible). Therefore $\Pi' = \Pi$ which contradicts the fact that $x \in \mathrm{Var}(\Pi)$.■

Lemma 4.7 *For any connected resolution refutation* Π *of* τ, *there is a resolution proof from* τ *of size at most* $|\Pi| + 2n^2$ *that derives, as unit clauses, all literals of variables in* $\mathrm{Var}(\Pi)$.

Proof. It suffices to note that if literals of all variables in $\mathrm{Res}(C \vee x_i^0, D \vee x_i^1)$ have been derived as unit clauses, then there is a proof of size at most $2n$ that derives x_i^0 and x_i^1. This process can be repeated on clauses in Π in reverse topological order (skipping clauses for which x_i^0 and x_i^1 have already been derived). Connectedness guarantees that every clause appearing in Π (and hence every variable) is processed.■

Theorem 4.8 *There is a polynomial time algorithm that, given a connected resolution refutation* Π *of* τ, *outputs a* π*-trail proof of size* $O(n^2|\tau||\Pi|)$ *that derives, as unit clauses, all literals of variables in* $\mathrm{Var}(\Pi)$.

Proof. We present the algorithm Sim which is recursively called on derivations with fewer variables.

Simulation Algorithm (Sim)

1. If $|\mathrm{Var}(\Pi)| = 1$, then for some variable x_i, Π contains only a resolution of x_i and $\neg x_i$. In this case, output the axioms x_i and $\overline{x_i}$.

2. Assume without loss of generality that all variables appear in Π. Define Π^0 to be the downward closure of some appearance of 0 in $\Pi|_{\{x_1=0\}}$. Derive $\mathrm{Lift}_\tau(\mathsf{Sim}(\Pi^0))$ (note that $|\mathrm{Var}(\Pi^0)| < |\mathrm{Var}(\Pi)|$, which justifies the recursive call to Sim) and let $l_{y,a} \in \{y^a, y^a \vee x_1\}$ for $y \in \mathrm{Var}(\Pi^0)$ denote the lifted unit clauses appearing in it. Note that Π^0 might be trivial, in which case x_1 is an axiom in τ and the next step can be skipped.

3. If x_1 appears in any $l_{y,a}$ from the previous step, then derive $x_1 = \mathrm{Res}(l_{y,0}, l_{y,1})$. Otherwise, by Lemma 4.6, there is a clause $C \in \tau$ containing the literal x_1. Derive x_1 by consecutively resolving C with literals $\overline{\ell_{y,a}}$, for all $\ell_{y,a}$ in C. We note here that these are half-ordered resolutions and hence admissible in π-trail resolution, but we refrain from pointing this out in similar cases below.

4. Derive the clauses $\{C \circ^{x_1} x_1 : C \in \tau\}$.

   At this point we have derived a set of clauses τ^* such that for every clause C in

   $$\psi \overset{\mathrm{def}}{=} \mathrm{Del}_{\{x_1\}}(\tau) \cup \bigcup_{y \in \mathrm{Var}(\Pi^0)} \{y^0, y^1\},$$

   the set τ^* contains either C or $C \vee x_1$.

5. Set $\mathcal{S} \coloneqq \mathrm{Var}(\Pi^0)$. While $\mathcal{S} \cup \{x_1\} \neq V$ perform the following procedure constructing a π-trail resolution proof from the set of axioms ψ. We maintain that at the start of each iteration, all unit clauses in $\bigcup_{y\in\mathcal{S}}\{y^0, y^1\}$ have been derived. Also, to make clear, the proof constructed in this step is **not** part of the output, but its lifted verison will be (in step 6).

   (a) Construct the clauses of $\mathrm{Del}_{\mathcal{S}\cup\{x_1\}}(\tau)$ by resolving each clause in $\mathrm{Del}_{\{x_1\}}(\tau)$ with the unit clauses x^a for $x \in \mathcal{S}$. Then build $\mathrm{Del}_{\mathcal{S}\cup\{x_1\}}(\Pi)$ using the deletion algorithm.

(b) Assume without loss of generality that $\mathrm{Del}_{\mathcal{S}\cup\{x_1\}}(\Pi)$ is connected; otherwise, as usual, take the downward closure of any appearance of 0. Construct the proof $\mathsf{Sim}(\mathrm{Del}_{\mathcal{S}\cup\{x_1\}}(\Pi))$. (This is the other recursive call to Sim.)

(c) Set $\mathcal{S} \coloneqq \mathcal{S} \cup \mathrm{Var}(\mathrm{Del}_{\mathcal{S}\cup\{x_1\}}(\Pi))$.

6. Since $\mathrm{Del}_{\mathcal{S}\cup\{x_1\}}(\Pi)$ is always nontrivial when $\mathcal{S}\cup\{x_1\} \neq V$ (this follows from the well-definedness of the Deletion operator, Claim 4.4), the previous step terminates. Call the resulting proof Υ; it is the union of the proofs from step 5 (Υ need not be connected), which derives from ψ all unit clauses x_i^a for $i \in [2, n]$. Derive the proof $\mathrm{Lift}_{\tau^*}(\Upsilon)$, where τ^* is the set of clauses in step 4. This proof derives (this time from τ) $l_{i,a} \in \{x_i^a, x_i^a \vee x_1\}$ for $i \in [2, n]$. It remains to derive $\overline{x_1}$.

7. For that purpose, it is now possible to build any trail (up to annotations) that extends $[x_1 \overset{d}{=} 0]$ by using the Unit Propagation Rule with the lifted unit clauses from the previous step. Therefore, we can simulate any resolution proof not containing the variable x_1 by directly simulating each resolution step. Do this to the resolution proof extending $\Pi|_{\{x_1=1\}}$ that derives all literals appearing in it (Lemma 4.7).

8. By Lemma 4.6, there is a clause $C \in \tau$ containing $\overline{x_1}$ that appears restricted in $\Pi|_{\{x_1=1\}}$. Derive $\overline{x_1}$ by resolving C with all new literals from the previous step, when possible.

9. Derive all remaining literals by resolving $l_{i,a}$ with $\overline{x_1}$ when necessary.

Let $f(n, m)$ and $s(n, m)$ be upper bounds on the running time of Sim and the size of π-trail proof output by Sim, respectively, when Sim is run on a proof containing at most n variables and whose size is at most m. Our primary focus is on understanding the contributions of step 2 and 5 since the algorithm is called recursively in these steps. Step 2 adds at most $s(n-1, |\Pi^0|)$ to $s(n, |\Pi|)$ and

$$f(n-1, |\Pi^0|) + O(n \cdot s(n-1, |\Pi|))$$

to $f(n, |\Pi|)$.

Suppose that step 5 iterates T (which is $\leq n$) times. For $i \in [T]$, define $\mathcal{S}^i$ to be the state of $\mathcal{S}$ before the i^{th} iteration and define Π^i to be $\mathrm{Del}_{\mathcal{S}^i\cup\{x_1\}}(\Pi)$. Then steps 5-6 contribute at most $\sum_{i=1}^{T} s(n-1, |\Pi^i|)$

to the size bound and

$$\sum_{i=1}^{T} f(n-1, |\Pi^i|) + O(n|\tau| \cdot |\Pi|)$$

to the running time bound.

The most important fact here is that, by Lemma 4.5, $\sum_{i=0}^{T} |\Pi^i| \leq |\Pi|$. This is because the sets $\mathrm{Var}(\Pi^i)$ for $i \in [0, T]$ are pairwise disjoint and so the resolutions in each proof Π^i correspond to unique resolutions in Π. Note the special case of Π^0, which uses the fact that restrictions, like variable deletion, have the property that all resolutions in the resulting proof correspond to resolutions in Π on the same variable.

The auxiliary operations performed throughout the algorithm (e.g., recreating trails by adding assignments to x_1 at the start) are clearly $O(n|\tau| \cdot |\Pi|)$ that yields the bounds

$$s(n, |\Pi|) \leq \sum_{i=0}^{T} s(n-1, |\Pi^i|) + O(n|\tau| \cdot |\Pi|)$$

and

$$f(n, |\Pi|) \leq \sum_{i=0}^{T} f(n-1, |\Pi^i|) + O\left(\sum_{i=0}^{T} s(n-1, |\Pi^i|)\right) + O(n^2|\tau| \cdot |\Pi|).$$

By induction on n, first for s and then f, it follows that $s(n, \Pi) = O(n^2|\tau| \cdot |\Pi|)$ and $f(n, |\Pi|) = O(n^3|\tau| \cdot |\Pi|)$.■

5. Width lower bound

Our last piece of technical work is Theorem 2.16, which demonstrates the limitations of bounded width clause learning in the presence of the ordered decision strategy. Using the connection to π-trail resolution from the previous section, Theorem 2.16 follows from a general width lower bound for the latter. Some of the formulas to which this bound applies have constant width (that implies polynomial size) refutations and hence, by Theorem 2.15, automatically have polynomial size π-trail refutations. Thus this result also shows that there is no size-width relationship for π-trail resolution like the one for resolution proved by Ben-Sasson and Wigderson [9].

Say that a clause C is *almost-k-small* with respect to π if $|\mathrm{Var}(C) \setminus \mathrm{Var}_\pi^k| \le 1$, and that a trail $t = [x_{i_1} \stackrel{*_1}{=} a_1, \ldots, x_{i_r} \stackrel{*_r}{=} a_r]$ is *k-trivial* if for $s \stackrel{\mathrm{def}}{=} \min(r, k)$, all assignments in $t[\le s]$ are decisions on variables in Var_π^k in π-increasing order: $t[\le s] = [x_{\pi(1)} \stackrel{d}{=} a_1, \ldots, x_{\pi(s)} \stackrel{d}{=} a_s]$.

Definition 5.1 The order π is *k-robust* for a contradictory CNF τ if for any restriction ρ such that $|\mathrm{Var}(\rho) \setminus \mathrm{Var}_\pi^k| \le 1$, the following properties hold:

- the formula $\tau|_\rho$ is minimally unsatisfiable, i.e., all strict subsets of $\tau|_\rho$ are satisfiable;
- for all $i \in [n]$, if $(x_i = a) \in \rho$ then there is a clause in τ that appears restricted in $\tau|_\rho$, i.e., it is not satisfied by ρ and contains the literal x_i^{1-a}.

Example 4 For a CNF τ_n, the *r-ary parity substitution* of τ_n, denoted by $\tau_n[\oplus_r]$, is the formula in which for all $i \in [n]$, each variable x_i is replaced with $\bigoplus_{j=1}^r y_{i,j}$ where the variables $y_{i,1}, y_{i,2}, \ldots, y_{i,r}$ are new and distinct. As described, $\tau_n[\oplus_r]$ is technically not a CNF, but its encoding as a CNF is straightforward and natural; see [36] for full details. It is also straightforward to check that whenever τ_n is minimally unsatisfiable and contains all variables $x_1, \ldots, x_n$, the order π on the variables of $\tau_n[\oplus_r]$ given by

$$\begin{aligned}&\pi(y_{1,1}) < \pi(y_{2,1}) < \cdots < \pi(y_{n,1}) < \\ &\pi(y_{1,2}) < \pi(y_{2,2}) < \cdots < \pi(y_{n,2}) < \cdots < \\ &\pi(y_{1,r}) < \pi(y_{2,r}) < \cdots < \pi(y_{n,r})\end{aligned}$$

is $((r-2)n)$-robust. In fact, this readily follows from the observation that any restriction ρ that satisfies $|\mathrm{Var}(\rho) \setminus \mathrm{Var}_\pi^{(r-2)n}| \le 1$ must leave unassigned at least one variable in each group $\{y_{i,1}, \ldots, y_{i,r}\}$.

The following theorem shows that robustness implies large width in π-trail resolution.

Theorem 5.2 *Let τ be a contradictory CNF formula and let π be a w-robust order for τ. Then the width of any π-trail refutation of τ is at least w.*

Proof. Assume without loss of generality that $\pi = \mathrm{id}$. Let Π be a π-trail refutation of τ and let C be the first almost-w-small clause appearing in Π. We will actually prove that $\mathrm{Var}_\pi^w \subseteq \mathrm{Var}(C)$.

First, we claim that all trails that appear before C in Π are $(w+1)$-trivial. Suppose otherwise and let t be the first trail in Π that is not. Since Π contains all prefixes of t, and all such prefixes precede t, it follows that t is of the form $[t', x_i \stackrel{u}{=} a]$, where $t' = [x_1 \stackrel{d}{=} a_1, x_2 \stackrel{d}{=} a_2 \ldots, x_j \stackrel{d}{=} a_j]$, $i \geq j+2$ and $j \leq w$. Suppose that t follows from t' by the Unit Propagation rule with the clause D. This means $D|_{t'}$ is a unit clause, which implies D is almost-w-small, contradicting the assumption that C is the first almost-w-small clause in Π.

It then follows that all resolutions (corresponding to applications of the Learning rule) that appear before C are on variables not in Var_π^{w+1}. Indeed, suppose that the inference

$$\frac{D \vee x_i^a \qquad E \vee \overline{x_i^a} \qquad t}{D \vee E}$$

appears before C in Π. By the claim in the previous paragraph, t is $(w+1)$-trivial. Therefore if $x_i \in \mathrm{Var}_\pi^{w+1}$, then it is actually assigned in $t[\leq w+1]$ and so are all variables appearing in D. This implies D is almost-w-small, contradicting the assumption that C is the first such clause.

Finally let Π^* be the *resolution refutation* corresponding to Π; that is, the refutation constructed from Π by ignoring all trails. Let Γ be the connected subproof of C in Π^* on the downward closure of C. By the remark in the previous paragraph, all resolutions in Γ are on variables not in Var_π^{w+1}. Lastly, let ρ be any restriction with the domain $\mathrm{Var}_\pi^w \cup \mathrm{Var}(C)$ that falsifies C, so that $\Gamma|_\rho$ is a refutation of $\tau|_\rho$. By the first property in the definition of robustness, $\tau|_\rho$ is minimal, which implies that all clauses in $\tau|_\rho$ appear as axioms of $\Gamma|_\rho$. Therefore, there are paths from these clauses (unrestricted) to C in Γ. By the second property of robustness, each variable in Var_π^w appears in at least one of these clauses. Since all resolutions in Γ are on variables not in Var_π^{w+1}, it follows that $\mathrm{Var}_\pi^w \subseteq \mathrm{Var}(C)$.■

Finally, we prove Theorem 2.16, which is restated here for convenience. The proof is a simple variation of the one above (we only have to make sure that the variables in Var_π^w appear in a *learned* clause).

Theorem 5.3 *(Theorem 2.16 restated) For any fixed order π on the variables and every $\epsilon > 0$ there exist contradictory CNFs τ_n with $w(\tau_n \vdash 0) = O(1)$ not provable in* CDCL(π-D, WIDTH-$(1-\epsilon)n$).

Proof. The formula used here is $\mathrm{Ind}_m[\oplus_r]$ where Ind_m is the *Induction principle*

$$x_1 \wedge \bigwedge_{i=1}^{m-1} (\overline{x_i} \vee x_{i+1}) \wedge \overline{x_m},$$

and r will be chosen as a sufficiently large constant. The natural resolution refutation of this formula has width $O(r)$.

Fix $\epsilon > 0$. Let R be a successful run in $\mathsf{CDCL}(\pi\text{-}\mathsf{D})$ on $\mathrm{Ind}_m[\oplus_r]$ and let Π be the natural π-trail simulation of this run given by Theorem 4.1. We begin with some observations about Π that are easily verified by examining the proof of Theorem 4.1. First, all clauses learned in R are derived exactly in Π, in the order they appear in R. Second, for any learning step (C, t') in R from the state $(\mathbb{C}, t)$, the proof Π contains the connected subproof of C from $\mathbb{C}$ corresponding exactly to the sequence of resolutions used to learn C (Lemma 3.6). Furthermore, the trail t appears before this subproof in Π.

Let $w = (r-2)m$ and let D be the first almost-w-small clause in Π. Similar to the proof of Theorem 5.2, it follows that $\mathrm{Var}_\pi^w \subseteq \mathrm{Var}(D)$ and all trails appearing before D in Π are $(w+1)$-trivial. If D is not a learned clause, then it appears in the subproof of some learned clause C. Suppose that C follows from the state $(\mathbb{C}, t)$ in R. As is made clear in Lemma 3.6, all resolutions in the subproof of C are on variables whose assignments are unit propagations in t. Since t appears before D, it is $(w+1)$-trivial, so none of the variables in Var_π^w are resolved on to derive C. This implies all variables in Var_π^w are inherited in C from D.

The result follows by taking $r > 2/\epsilon$ so that $(r-2)m > (1-\epsilon)rm$. ∎

6. Conclusion

This paper 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 simplest version that always requires to choose the first available variable, under a fixed orderings. 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.

The result that $\mathsf{CDCL}(\pi\text{-}\mathsf{D}, \mathsf{ALWAYS\text{-}C}, \mathsf{ALWAYS\text{-}U}, \mathsf{DECISION\text{-}L})$ is not as powerful as resolution supports the observation from practice that CDCL solvers with the ordered decision strategy are *usually* less efficient than those with more dynamic decision strategies. What can be proved if DECISION-L is replaced with some other amendment modeling a different, possibly more practical learning scheme? Furthermore, is it possible that CDCL(π-D, ALWAYS-C, ALWAYS-U) does not simulate general resolution?

Just as in [6, 37, 3], our simulations use very frequent restarts. Perhaps the most interesting open question in this area is whether it is actually necessary. In the language we have introduced, this amounts to understanding the power of proof systems CDCL(NEVER-R) and CDCL(ALWAYS-C, ALWAYS-U, NEVER-R), the latter version being more oriented towards actual CDCL solvers.

We have also proved that our simulations fail quite badly with respect to width (as opposed to size): there are contradictory CNFs τ_n refutable in constant width but not belonging to CDCL(π-D, WIDTH-$(1-\epsilon)n$). The ordering π in our result, however, essentially depends on τ_n. Is a uniform version possible? That is, do there exist contradictory CNFs τ_n refutable in small width that do not belong to (say) CDCL(π-D, WIDTH-$\Omega(n)$) for *any* ordering π? Another interesting question, extracted from [3], asks if τ_n refutable in small width are always in (say) CDCL(ALWAYS-C, ALWAYS-U, WIDTH-$\Omega(n)$).

Finally (cf. Remark 6) our model is geared towards "positional solvers", i.e., those that are allowed to carry along only the set of learned clauses and the current trail but are otherwise oblivious to the history of the run. This restriction is of little importance in the theoretical, nondeterministic part of the spectrum, but it will make a big difference if we would like to study dynamic decision strategies (like those mentioned in [40]), further strengthen the amendments ALWAYS-C and ALWAYS-U by postulating the behavior in the presence of multiple choices, etc. It would be interesting to develop a rigorous mathematical formalism that would include nonpositional behavior as well.

References

[1] Fadi A. Aloul, Igor L. Markov, and Karem A. Sakallah. MINCE: A Static Global Variable-Ordering for SAT and BDD. In *International Workshop on Logic and Synthesis*, pages 1167–1172, 2001.

[2] Fadi A. Aloul, Igor L. Markov, and Karem A. Sakallah. FORCE: A Fast & Easy-to-Implement Variable-Ordering Heuristic. In *Proceedings of the 13th ACM Great Lakes symposium on VLSI*, pages 116–119. ACM, 2003.

[3] Albert Atserias, Johannes Klaus Fichte, and Marc Thurley. Clause-Learning Algorithms with Many Restarts and Bounded-Width Resolution. *Journal of Artificial Intelligence Research*, 40:353–373, 2011.

[4] Gilles Audemard and Laurent Simon. Predicting learnt clauses quality in modern SAT solvers. In *IJCAI*, volume 9, pages 399–404, 2009.

[5] Paul Beame, Richard Karp, Toniann Pitassi, and Michael Saks. The efficiency of resolution and Davis-Putnam procedures. *SIAM Journal on Computing*, 31(4):1048–1075, 2002.

[6] Paul Beame, Henry Kautz, and Ashish Sabharwal. Towards Understanding and Harnessing the Potential of Clause Learning. *Journal of Artificial Intelligence Research*, 22:319–351, 2004.

[7] Paul Beame and Ashish Sabharwal. Non-Restarting SAT Solvers with Simple Preprocessing Can Efficiently Simulate Resolution. In *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence*, pages 2608–2615, 2014.

[8] Eli Ben-Sasson and Jan Johannsen. Lower Bounds for Width-Restricted Clause Learning on Small Width Formulas. In *Theory and Applications of Satisfiability Testing – SAT 2010*, pages 16–29. Springer, 2010.

[9] Eli Ben-Sasson and Avi Wigderson. Short Proofs Are Narrow – Resolution Made Simple. *Journal of the ACM*, 48(2):149–169, 2001.

[10] Armin Biere and Andreas Fröhlich. Evaluating CDCL restart schemes. In *Proceedings POS-15. Sixth Pragmatics of SAT workshop*, 2015.

[11] Armin Biere, Marijn Heule, Hans van Maaren, and Toby Walsh, editors. *Handbook of Satisfiability*, volume 336 of *Frontiers in Artificial Intelligence and Applications*. IOS Press, 2021.

[12] Archie Blake. *Canonical expressions in boolean algebra*. PhD thesis, University of Chicago, 1938.

[13] Beate Bollig, Martin Löbbing, Martin Sauerhoff, and Ingo Wegener. On the complexity of the hidden weighted bit function for various BDD models. *RAIRO-Theoretical Informatics and Applications*, 33(2):103–115, 1999.

[14] Maria Luisa Bonet, Sam Buss, and Jan Johannsen. Improved Separations of Regular Resolution from Clause Learning Proof Systems. *Journal of Artificial Intelligence Research*, 49:669–703, 2014.

[15] Maria Luisa Bonet, Juan Luis Esteban, Nicola Galesi, and Jan Johannsen. On the Relative Complexity of Resolution Refinements and Cutting Planes Proof Systems. *SIAM Journal on Computing*, 30(5):1462–1484, 2000.

[16] Maria Luisa Bonet and Katherin St. John. Efficiently calculating evolutionary tree measures using SAT. In *International Conference on Theory and Applications of Satisfiability Testing*, pages 4–17. Springer, 2009.

[17] Samuel R. Buss, Jan Hoffmann, and Jan Johannsen. Resolution Trees with Lemmas: Resolution Refinements that Characterize DLL-Algorithms with Clause Learning. *Logical Methods in Computer Science*, 4(4), 2008.

[18] Samuel R. Buss and Leszek Aleksander Kołodziejczyk. Small Stone in Pool. *Logical Methods in Computer Science*, 10(2), 2014.

[19] Eldan Cohen, Guoyu Huang, and Christopher J. Beck. (I can get) satisfaction: Preference-based scheduling for concert-goers at multi-venue music festivals. In *International Conference on Theory and Applications of Satisfiability Testing*, pages 147–163. Springer, 2017.

[20] Stephen A. Cook and Robert A. Reckhow. The relative efficiency of propositional proof systems. *The Journal of Symbolic Logic*, 44(1):36–50, 1979.

[21] Martin Davis, George Logemann, and Donald Loveland. A machine program for theorem-proving. *Communications of the ACM*, 5(7):394–397, 1962.

[22] Martin Davis and Hilary Putnam. A computing procedure for quantification theory. *Journal of the ACM (JACM)*, 7(3):201–215, 1960.

[23] Jan Elffers, Jan Johannsen, Massimo Lauria, Thomas Magnard, Jakob Nordström, and Marc Vinyals. Trade-offs Between Time and Memory in a Tighter Model of CDCL SAT Solvers. In *Theory and Applications of Satisfiability Testing – SAT 2016*, pages 160–176. Springer, 2016.

[24] Aarti Gupta, Malay Ganai, and Chao Wang. SAT-based verification methods and applications in hardware verification. *Formal Methods for Hardware Verification*, pages 108–143, 2006.

[25] Philipp Hertel, Fahiem Bacchus, Toniann Pitassi, and Allen Van Gelder. Clause Learning Can Effectively P-Simulate General Propositional Resolution. In *Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence*, pages 283–290, 2008.

[26] Franjo Ivančić, Zijiang Yang, Malay K. Ganai, Aarti Gupta, and Pranav Ashar. Efficient SAT-based bounded model checking for software verification. *Theoretical Computer Science*, 404(3):256–274, 2008.

[27] Roberto J. Bayardo Jr. and Robert C. Schrag. Using CSP look-back techniques to solve real-world SAT instances. In *AAAI/IAAI*, pages 203–208, 1997.

[28] Chunxiao Li, Noah Fleming, Marc Vinyals, Toniann Pitassi, and Vijay Ganesh. Towards a Complexity-theoretic Understanding of Restarts in SAT solvers. In *Theory and Applications of Satisfiability Testing – SAT 2020*, pages 233–249. Springer, 2020.

[29] Jia Hui Liang, Vijay Ganesh, Ed Zulkoski, Atulan Zaman, and Krzysztof Czarnecki. Understanding VSIDS branching heuristics in conflict-driven clause-learning SAT solvers. In *Haifa Verification Conference*, pages 225–241. Springer, 2015.

[30] João P. Marques-Silva. The impact of branching heuristics in propositional satisfiability algorithms. *Progress in Artificial Intelligence*, pages 850–850, 1999.

[31] João P. Marques-Silva and Karem A. Sakallah. GRASP: A search algorithm for propositional satisfiability. *IEEE Transactions on Computers*, 48(5):506–521, 1999.

[32] Ralph Eric McGregor. *Automated theorem Proving using SAT*. Clarkson University, 2011.

[33] Matthew W. Moskewicz, Conor F. Madigan, Ying Zhao, Lintao Zhang, and Sharad Malik. Chaff: Engineering an efficient SAT

solver. In *Proceedings of the 38th Annual Design Automation Conference*, pages 530–535. ACM, 2001.

[34] Nathan Mull, Shuo Pang, and Alexander Razborov. On CDCL-based proof systems with the ordered decision strategy. In *Theory and Applications of Satisfiability Testing – SAT 2020*, pages 149–165. Springer, 2020.

[35] Robert Nieuwenhuis, Albert Oliveras, and Cesare Tinelli. Solving SAT and SAT Modulo Theories: From an Abstract Davis–Putnam–Logemann–Loveland Procedure to DPLL(T). *Journal of the ACM*, 53(6):937–977, 2006.

[36] Jakob Nordström. Pebble games, proof complexity, and time-space trade-offs. *Logical Methods in Computer Science*, 9, 2013.

[37] Knot Pipatsrisawat and Adnan Darwiche. On the power of clause-learning SAT solvers as resolution engines. *Artificial Intelligence*, 175(2):512–525, 2011.

[38] John Alan Robinson. A machine-oriented logic based on the resolution principle. *Journal of the ACM (JACM)*, 12(1):23–41, 1965.

[39] Allen Van Gelder. Pool Resolution and Its Relation to Regular Resolution and DPLL with Clause Learning. In *Logic for Programming, Artificial Intelligence, and Reasoning – LPAR 2005*, pages 580–594. Springer, 2005.

[40] Marc Vinyals. Hard Examples for Common Variable Decision Heuristics. In *Proceedings of the 34th AAAI Conference on Artificial Intelligence (AAAI'20)*, 2020.