Electronic Colloquium on Computational Complexity, Report No. 105 (2020)

Automating Regular or Ordered Resolution is NP-Hard

Zoë Bell[*]
Harvey Mudd College
zbell@hmc.edu

July 14, 2020

Abstract

We show that is hard to find regular or even ordered (also known as Davis-Putnam) Resolution proofs, extending the breakthrough result for general Resolution from Atserias and Müller [AM19] to these restricted forms. Namely, regular and ordered Resolution are automatable if and only if $\mathsf{P} = \mathsf{NP}$. Specifically, for a CNF formula F the problem of distinguishing between the existence of a polynomial-size ordered Resolution refutation of F and an at least exponential-size general Resolution proof being required to refute F is NP-complete.

[*]This research was done at the University of Washington, supported by NSF-SHF grant CCF-1714593.

ISSN 1433-8092

1 Introduction

A propositional proof system is a sound and complete polynomial-time verifier for purported proofs of propositional statements. Such a proof system is called *automatable* (in polynomial time) if there is an algorithm that on input of a CNF formula F will find a proof refuting F in time polynomial in the length of F and the shortest proof that exists in the system. It has long been an question of interest whether a natural proof system called Resolution, which is the basis of all CDCL SAT solvers, is automatable. Getting within an additive polynomial or constant factor of minimum Resolution proof size, stricter requirements than automatability, have been known to be NP-hard for a while [Iwa97, ABMP01]. Results supporting the idea that Resolution is not automatable either under various assumptions include those in [AR08, EGG08, MPW19].

In 2019, there was a breakthrough result from Atserias and Müller [AM19] that was able to show that Resolution is not automatable in polynomial time under the optimal assumption that $\mathsf{P} \neq \mathsf{NP}$ (optimal since if $\mathsf{P} = \mathsf{NP}$, then Resolution would have to be polynomial-time automatable). Over the last year, several more exciting results followed that extended their results to several algebraic and semialgebraic proof systems, due to Göös et al. [GNP$^+$20, GKMP20].

More precisely, altogether, these results showed that Resolution (**Res**), Nullstellensatz (**NS**), Polynomial Calculus (**PC**), Sherali-Adams (**SA**), and Cutting Planes (**CP**) are each automatable in polynomial time if and only if $\mathsf{P} = \mathsf{NP}$. They are also not automatable in subexponential time unless the Exponential Time Hypothesis (ETH) fails, which is the conjecture that 3-SAT on n-variable formulas requires $2^{\Omega(n)}$ time. Further, it is not possible to approximate minimum proof length in each system within subexponential error unless $\mathsf{P} = \mathsf{NP}$. These results are all consequences of the following finding of these papers:

Proposition 1.1 ([AM19, GNP$^+$20, GKMP20])**.** *For any proof system $\boldsymbol{S} = \boldsymbol{Res}, \boldsymbol{NS}, \boldsymbol{PC}, \boldsymbol{SA}, \boldsymbol{CP}$, there is a polynomial-time algorithm $\mathcal{A}$ that on input of an n-variate 3-CNF formula F outputs a CNF formula $\mathcal{A}(F)$ such that:*

- *If F is satisfiable, then $\mathcal{A}(F)$ admits an $\boldsymbol{S}$-refutation of size at most $n^{O(1)}$.*
- *If F is unsatisfiable, then $\mathcal{A}(F)$ requires $\boldsymbol{S}$-refutations of size at least $2^{n^{\Omega(1)}}$.*

One major implication of these results is that if $\mathsf{P} \neq \mathsf{NP}$, any SAT solver that is based on any of the above proof systems will sometimes take superpolynomial time to find a proof even when a polynomial-size proof exists. Further, they will sometimes take exponential time unless ETH fails. This includes the most common kind of SAT solver, CDCL SAT solvers, because they produce Resolution proofs.

A question that naturally arises is whether this result can also be extended to several restricted forms of Resolution with exponential separation from general Resolution. Perhaps if there is a short proof in a more restricted form of Resolution with fewer possibilities then this would allow us to find a short proof efficiently. Indeed, [BP96] noted that tree-like Resolution is automatable in quasi-polynomial time using a divide-and-conquer method related to the size-width tradeoff. Thus it is not expected that Proposition 1.1 can be extended to tree-like Resolution since if it could be, ETH would fail. After the talk given by Atserias at the 2020 Banff Proof Complexity conference, whether the result extends to another natural restriction of Resolution called regular Resolution was raised as an open question [Ats20].

In this paper, we answer this question and show that Proposition 1.1 can be extended to the restricted version of Resolution given by the proof system of regular Resolution ($\mathbf{Res_{reg}}$). Further, we are able to show that these results hold for an even more restricted special case of regular Resolution, ordered Resolution ($\mathbf{Res_{ord}}$). This proof system is also known as Davis-Putnam Resolution, and goes back to the initial development of Resolution-based proof search algorithms in [DP60]. Ordered Resolution is a very restricted form of Resolution that is incomparable with tree-like Resolution since neither proof system can polynomially simulate the other, but both can be compared with regular Resolution as the smallest proofs in each system are regular [Urq95, BDP09]. Unlike with tree-like Resolution however, it is also NP-hard to find ordered Resolution proofs, or to even distinguish between when there is a short (polynomial-size) ordered Resolution proof and when a long (exponential-size) proof is necessary even using general Resolution or other more powerful proof systems from Proposition 1.1.

To establish these results, we will extend the proof systems Proposition 1.1 applies to and show

Theorem 1.2. *There is a polynomial-time algorithm $\mathcal{A}$ that on input of an n-variate 3-CNF formula F outputs a CNF formula $\mathcal{A}(F)$ such that for either proof system $\boldsymbol{S} = \boldsymbol{Res_{reg}}$ or $\boldsymbol{Res_{ord}}$,:*

- *If F is satisfiable, then $\mathcal{A}(F)$ admits an $\boldsymbol{S}$-refutation of size at most $n^{O(1)}$.*
- *If F is unsatisfiable, then $\mathcal{A}(F)$ requires $\boldsymbol{S}$-refutations of size at least $2^{n^{\Omega(1)}}$.*

It was shown in [GNP$^+$20] that all of the Proposition 1.1 results—except for CP—can be proven using the same formula $\mathcal{A}(F)$. We will use this same formula to establish Theorem 1.2. We thus inherit the lower bound in the case that F is unsatisfiable from the previous treatment of Resolution, and need only establish the upper bound in the case that F is satisfiable within these restricted systems. Thereby using the same $\mathcal{A}(F)$ (plus a slight adjustment for CP) also implies

Theorem 1.3. *For any proof system $\boldsymbol{S} = \boldsymbol{Res}, \boldsymbol{NS}, \boldsymbol{PC}, \boldsymbol{SA}$, $\boldsymbol{CP}, \boldsymbol{Res_{reg}}, \boldsymbol{Res_{ord}}$, there is a polynomial-time algorithm $\mathcal{A}$ that on input of an n-variate 3-CNF formula F outputs a CNF formula $\mathcal{A}(F)$ such that:*

- *If F is satisfiable, then $\mathcal{A}(F)$ admits an $\boldsymbol{Res_{ord}}$-refutation of size at most $n^{O(1)}$.*
- *If F is unsatisfiable, then $\mathcal{A}(F)$ requires $\boldsymbol{S}$-refutations of size at least $2^{n^{\Omega(1)}}$.*

In particular, even if there is a short ordered Resolution proof, there is no algorithm that will be able to find a short general Resolution proof in polynomial time (if $\mathsf{P} \neq \mathsf{NP}$) or in subexponential time (if ETH holds). It is even NP-hard to distinguish between the existence of a polynomial-size ordered Resolution proof and an exponential-size Resolution proof being required in general.

2 Preliminaries

Resolution is a proof system for CNF formulas. A *Resolution refutation* of an unsatisfiable CNF formula F with clauses $\mathcal{C}_1, \mathcal{C}_2, \ldots, \mathcal{C}_m$ is a sequence of clauses $(\mathcal{B}_1, \mathcal{B}_2, \ldots, \mathcal{B}_s)$ where each $\mathcal{B}_i$ is derived one of two ways and we end with $\mathcal{B}_s = \emptyset$ (which implies that if F is satisfied, the empty clause will be too, which is impossible, giving us our refutation). One option is that $\mathcal{B}_i$ can be the weakening of one of the $\mathcal{C}_k$ axioms. The other is that is can be the result of weakening the $A \vee B$ clause given by the following rule:

$$\frac{A \vee x \qquad B \vee \overline{x}}{A \vee B}$$

where x is a variable and $\mathcal{B}_{i_L} = A \vee x$ (the left premise) and $\mathcal{B}_{i_R} = B \vee \overline{x}$ (the right premise) are two previous clauses in the refutation ($i_L, i_R < i$).

Two complexity measures commonly used for Resolution are *size* and *width*. The size of a Resolution refutation is s, the number of clauses in it. The minimum size of any Resolution refutation of a CNF formula F is denoted by $\mathbf{Res}(F)$. The width of a Resolution refutation is the maximum number of literals in any clause in the refutation. In this paper, a generalization of the concept of width will be more useful. After defining a partitioning of the variables of F into blocks, the *block-width* of a Resolution refutation is the maximum number of blocks touched by any clause in the refutation. Notice that the block-width is the same as the width if each variable is defined to be in a separate block. We will denote the block-width of a Resolution refutation $\mathcal{P}$ by $\mathrm{bw}(\mathcal{P})$. The minimum block-width of any Resolution refutation of a CNF formula F is denoted by $\mathrm{bw}(F \vdash \bot)$.

A Resolution refutation can also be depicted as a DAG with the $\mathcal{B}_i$ as its nodes. Specifically, the $\mathcal{C}_k$ axioms (or weakenings of them) are the leaves, $\emptyset$ or $\bot$ is the root, and the internal nodes are the clauses derived via the Resolution rule. The edges in the DAG are from the left premise and right premise to the clause they were used to derive.

This view allows us to more easily define the proof systems of regular Resolution and ordered Resolution. A *regular Resolution refutation* is a Resolution refutation where any variable is resolved upon at most once along any path through the DAG of the refutation. An *ordered Resolution refutation* is a regular Resolution refutation where a single total order of the variables is respected by the variables resolved upon along any path through the DAG.

It is also useful to view regular and ordered Resolution through a lens brought to us by read-once branching programs [Kra95]. In this frame, a querier is interacting with an assignment and seeking to find the input clause that the assignment violates, which must exist since the formula is unsatisfiable. A path through the DAG of a regular Resolution proof corresponds to responses to a sequence of queries of the variables involved that ultimately finds this conflicting clause, so that we see the DAG corresponds to a read-once branching program solving the conflict clause search problem.

In particular, a node labelled with a given clause in the DAG corresponds to a node in the branching program labelled with a partial assignment which precisely violates that clause. Start at the root of the DAG. If we are at a derived node in the DAG, we then query the variable that was resolved on to derive it and continue our path to the clause with the literal that is false under the assignment to that variable to maintain our invariant. Further, if there are variables that are assigned which are no longer in this new clause, then we forget them. Because the Resolution refutation is regular, we need not be concerned about querying them again later. When we reach a leaf, we will have found an input clause that is precisely violated by the assignment. We can also construct an assignment that will take us along any path through the DAG we choose when this procedure is followed.

Therefore showing that a refutation is regular can also be framed as showing that any path through the DAG corresponds to responses to a sequence of queries in which the same variable is never re-queried, producing a read-once branching program. Likewise, to show that it is ordered, we can show that a fixed ordering of variables is always respected by the sequence of queries, producing a special case of a read-once branching program called an Ordered Binary Decision Diagram (OBDD) [LNNW95].

3 The Ref(F) formula

The formula $\mathcal{A}(F)$ from the introduction is based on (a slight variation of) the Ref(F) formula described below, which semantically describes a short Resolution refutation of F. The formula we present is slightly different from the encodings used in previous works. This is done for convenience, and does not affect the ultimate result. See Appendix A for the details of how our formalization differs from those of [AM19] and [GNP$^+$20].

3.1 Variables of Ref(F)

Fix a CNF formula F with n variables and $m = \text{poly}(n)$ clauses, $\mathcal{C}_1, \mathcal{C}_2, \ldots, \mathcal{C}_m$. Fix a maximum allowed refutation length s for the Resolution refutation refuting F described by Ref(F). We ultimately set $s = n^3$. Let $\mathcal{B}_i$ denote the ith clause in the Resolution refutation of F ($\mathcal{B}_1, \mathcal{B}_2, \ldots, \mathcal{B}_s$) described by Ref($F$). See Table 1 for the definition of the variables we will use.

We partition these variables into blocks based on which clause $\mathcal{C}_k$ or $\mathcal{B}_i$ they are primarily associated with. Specifically, for each $k \in [m]$, the variables $C_{k,q,b}$ ranging over $q \in [n]$ and $b \in \{0, 1\}$ constitute a block. We will call this the $\mathcal{C}_k$ block of variables. Additionally, for each $i \in [s]$, the variables $B_{i,q,b}$, D_i, A_i, $W_{i,k}$, and $I_{i,i_L,i_R,\ell}$ ranging over all other indices constitute a block. We call this the $\mathcal{B}_i$ block. We do this in preparation to lift bounds on block-width to bounds on size in order to maintain the desired lower bound in the case that F is unsatisfiable.

Variable	**Meaning**
$C_{k,q,b}$	Input clause $\mathcal{C}_k$ of F contains x_q^b where $x_q^1 = x_q$ and $x_q^0 = \overline{x_q}$ $k \in [m]$, $q \in [n]$, $b \in \{0,1\}$
$B_{i,q,b}$	Clause $\mathcal{B}_i$ of the refutation of F contains x_q^b $i \in [s]$, $q \in [n]$, $b \in \{0,1\}$
D_i	Clause $\mathcal{B}_i$ is disabled (we do not enforce rules on it) $i \in [s]$
A_i	Clause $\mathcal{B}_i$ is a weakening of an input axiom $i \in [s]$
$W_{i,k}$	Clause $\mathcal{B}_i$ is a weakening of input clause $\mathcal{C}_k$ $i \in [s]$, $k \in [m]$
$I_{i,i_L,i_R,\ell}$	Clause $\mathcal{B}_i$ is inferred from $\mathcal{B}_{i_L}$ and $\mathcal{B}_{i_R}$ by resolving on x_ℓ ($x_\ell \in \mathcal{B}_{i_L}$, $\overline{x_\ell} \in \mathcal{B}_{i_R}$) $i \in [s]$, $1 \leq i_L, i_R < i$, $\ell \in [n]$

Table 1: Variables of Ref(F)

3.2 Axioms of Ref(F)

We define the following axioms for Ref(F) given in Table 2 so that it must encode a valid refutation of F of length at most s (allowing weakening throughout). Thus Ref(F) can be written in CNF form with a polynomial number of clauses in n, m, and s (and thus ultimately in n). Notice that the block-width of Ref(F) is 2 since no more than two clauses are directly discussed at a time.

As the (1) and (1') axioms make clear, the $C_{k,q,b}$ variables are technically unnecessary since their values are immediately forced. In particular, the (1') axioms will not even be used in the refutation (as they have nothing to resolve with). These variables and axioms are included to help clarify what is going on in the Ref(F) formula and its refutation semantically.

4 Order that will be respected by our refutation

Our resolution of variables respects an ordering starting with all of the $\mathcal{C}_k$ (with any ordering between them) followed by the $\mathcal{B}_i$ blocks starting from 1 up to s. Essentially, in the language of branching programs and queries (for which the order is reversed since we start at the root), we are able to iteratively reduce our search for a conflicting clause from variables defining a $\mathcal{B}_i$ clause to those defining a $\mathcal{B}_j$ clause earlier in the refutation or finally to those defining an input axiom $\mathcal{C}_k$.

Additionally, our resolution of variables within a given $\mathcal{B}_i$ block respects the following

Axiom	**Meaning**
(1) $C_{k,q,b}$	x_q^b is in $\mathcal{C}_k$ of our fixed F where $x_q^1 = x_q$ and $x_q^0 = \overline{x_q}$ $k \in [m]$, $q \in [n]$ and $b \in \{0,1\}$ such that $x_q^b \in \mathcal{C}_k$
(1') $\overline{C_{k,q,b}}$	x_q^b is *not* in $\mathcal{C}_k$ of our fixed F $k \in [m]$, $q \in [n]$ and $b \in \{0,1\}$ such that $x_q^b \notin \mathcal{C}_k$
(2) $D_i \vee A_i \vee \bigvee_{\substack{1 \le i_L, i_R < i \\ \ell \in [n]}} I_{i,i_L,i_R,\ell}$	Either $\mathcal{B}_i$ is disabled, the weakening of an axiom, or it is derived from something $i \in [s]$
(3) $\overline{A_i} \vee \bigvee_{k \in [m]} W_{i,k}$	If $\mathcal{B}_i$ is a weakening of an input axiom, then it must be the weakening of a specific one $i \in [s]$
(4) $\overline{W_{i,k}} \vee \overline{C_{k,q,b}} \vee B_{i,q,b}$	If clause $\mathcal{B}_i$ is a weakening of input clause $\mathcal{C}_k$, then everything in $\mathcal{C}_k$ must be in $\mathcal{B}_i$ $i \in [s]$, $k \in [m]$, $q \in [n]$, $b \in \{0,1\}$
(5) $\overline{I_{i,i_L,i_R,\ell}} \vee \overline{D_{i_L}}$	If $\mathcal{B}_i$ is derived from $\mathcal{B}_{i_L}$, $\mathcal{B}_{i_R}$ by resolving on x_ℓ, then $\mathcal{B}_{i_L}$ cannot be disabled $i \in [s]$, $1 \le i_L, i_R < i$, $\ell \in [n]$
(6) $\overline{I_{i,i_L,i_R,\ell}} \vee \overline{D_{i_R}}$	If $\mathcal{B}_i$ is derived from $\mathcal{B}_{i_L}$, $\mathcal{B}_{i_R}$ by resolving on x_ℓ, then $\mathcal{B}_{i_R}$ cannot be disabled $i \in [s]$, $1 \le i_L, i_R < i$, $\ell \in [n]$
(7) $\overline{I_{i,i_L,i_R,\ell}} \vee B_{i_L,\ell,1}$	If $\mathcal{B}_i$ is derived from $\mathcal{B}_{i_L}$, $\mathcal{B}_{i_R}$ by resolving on x_ℓ, then $\mathcal{B}_{i_L}$ must contain x_ℓ $i \in [s]$, $1 \le i_L, i_R < i$, $\ell \in [n]$
(8) $\overline{I_{i,i_L,i_R,\ell}} \vee B_{i_R,\ell,0}$	If $\mathcal{B}_i$ is derived from $\mathcal{B}_{i_L}$, $\mathcal{B}_{i_R}$ by resolving on x_ℓ, then $\mathcal{B}_{i_R}$ must contain $\overline{x_\ell}$ $i \in [s]$, $1 \le i_L, i_R < i$, $\ell \in [n]$
(9) $\overline{I_{i,i_L,i_R,\ell}} \vee \overline{B_{i_L,q,b}} \vee B_{i,q,b}$	If $\mathcal{B}_i$ is derived from $\mathcal{B}_{i_L}$, $\mathcal{B}_{i_R}$ by resolving on x_ℓ, then everything in $\mathcal{B}_{i_L}$ must be in $\mathcal{B}_i$—except x_ℓ $i \in [s]$, $1 \le i_L, i_R < i$, $\ell \in [n]$, $q \in [n]$ and $b \in \{0,1\}$ except $q = \ell$ and $b = 1$
(10) $\overline{I_{i,i_L,i_R,\ell}} \vee \overline{B_{i_R,q,b}} \vee B_{i,q,b}$	If $\mathcal{B}_i$ is derived from $\mathcal{B}_{i_L}$, $\mathcal{B}_{i_R}$ by resolving on x_ℓ, then everything in $\mathcal{B}_{i_R}$ must be in $\mathcal{B}_i$—except $\overline{x_\ell}$ $i \in [s]$, $1 \le i_L, i_R < i$, $\ell \in [n]$, $q \in [n]$ and $b \in \{0,1\}$ except $q = \ell$ and $b = 0$
(11) $\overline{D_s}$	The final clause cannot be disabled
(12) $\overline{B_{s,q,b}}$	The final clause must be empty $q \in [n]$, $b \in \{0,1\}$

Table 2: Clauses of Ref(F)

order on variable types as well:

$$\begin{gathered} I_{i,i_L,i_R,\ell} \\ W_{i,k} \\ A_i \\ B_{i,q,b} \\ D_i \end{gathered}$$

We can also adjust our refutation to respect whatever order we want within the $I_{i,i_L,i_R,\ell}$, $W_{i,k}$, and $B_{i,q,b}$ variable types because we will resolve on all of the variables of a type one after the other in an arbitrary sequence. This holds true as well for the one variable type in a $\mathcal{C}_k$ block, the $C_{k,q,b}$.

By maintaining this order, we will show that our refutation is not only regular, but ordered. Further, notice that this ordering demonstrates that our refutation is what we will call block-regular and block-ordered, which essentially means that its regularity and ordering respect the blocks the variables of the formula are partitioned into:

Definition 1. *A Resolution refutation is* block-regular *if it is a regular Resolution refutation where variables from a given block are resolved upon in at most one consecutive segment along any path through the DAG of the refutation.*

Definition 2. *A Resolution refutation is* block-ordered *if it is ordered, and the ordering on its variables respects an ordering on the blocks.*

Notice that a Resolution refutation is block-ordered if and only if it is block-regular and ordered. These properties will be crucial when we seek to maintain regularity and order on the lifted version of the Ref(F) formula.

5 Upper bound on ordered refutation size for Ref(F) with F satisfiable

We will describe a polynomial-size refutation of Ref(F) in the case that F is satisfiable. Thus let the sets $P \subseteq [n]$ and $N = [n] \setminus P$ reflect a satisfying assignment by placing $q \in P$ if the satisfying assignment sets $x_q = 1$ and $q \in N$ if the satisfying assignment sets $x_q = 0$ for each $q \in [n]$. This satisfying assignment will guide our efficient refutation.

The outline of the refutation of Ref(F) is as follows. For each i (starting with 1 and ending with s) we seek to derive

$$D_i \vee \bigvee_{q \in P} B_{i,q,1} \vee \bigvee_{q \in N} B_{i,q,0},$$

which says that if clause $\mathcal{B}_i$ is not disabled, then it contains a literal which is satisfied by the satisfying assignment of F. This is to be expected since Resolution is sound, all of the input axioms of F are satisfied by the assignment, and $\mathcal{B}_i$ must have been ultimately derived from these if it is not disabled.

It is thus also clear where our ultimate contradiction arises. The final clause in the refutation is not disabled and must be the empty clause, and thus cannot contain a literal which is satisfied under the given assignment. Indeed, once we derive

$$D_s \vee \bigvee_{q \in P} B_{s,q,1} \vee \bigvee_{q \in N} B_{s,q,0},$$

we can resolve it with the appropriate (12) $\overline{B_{s,q,b}}$ (all $q \in [n]$ with $b = 1$ if $q \in P$ and $b = 0$ if $q \in N$) and finally the (11) $\overline{D_s}$ axiom to derive the empty clause and finish the refutation of $\mathrm{Ref}(F)$.

Notice that this contributes no more than $1+n$ derived clauses to the size of our refutation, and all clauses involved have block-width 1. Also notice that our ordering has been respected because we are ending with resolving variables from the $\mathcal{B}_s$ block, and we first resolved the $B_{s,q,b}$ and finally D_s.

To fill in the details of this strategy, it is natural to describe the reduction inductively. Assume that we have already derived

$$D_j \vee \bigvee_{q \in P} B_{j,q,1} \vee \bigvee_{q \in N} B_{j,q,0}$$

for all $j < i$ and now seek to derive

$$D_i \vee \bigvee_{q \in P} B_{i,q,1} \vee \bigvee_{q \in N} B_{i,q,0}.$$

To verify that these steps of the refutation remain ordered, we will see that in the process of deriving the i-th clause above we will only resolve on the following types of variables in the following order after using our derivation of a $j < i$ clause:

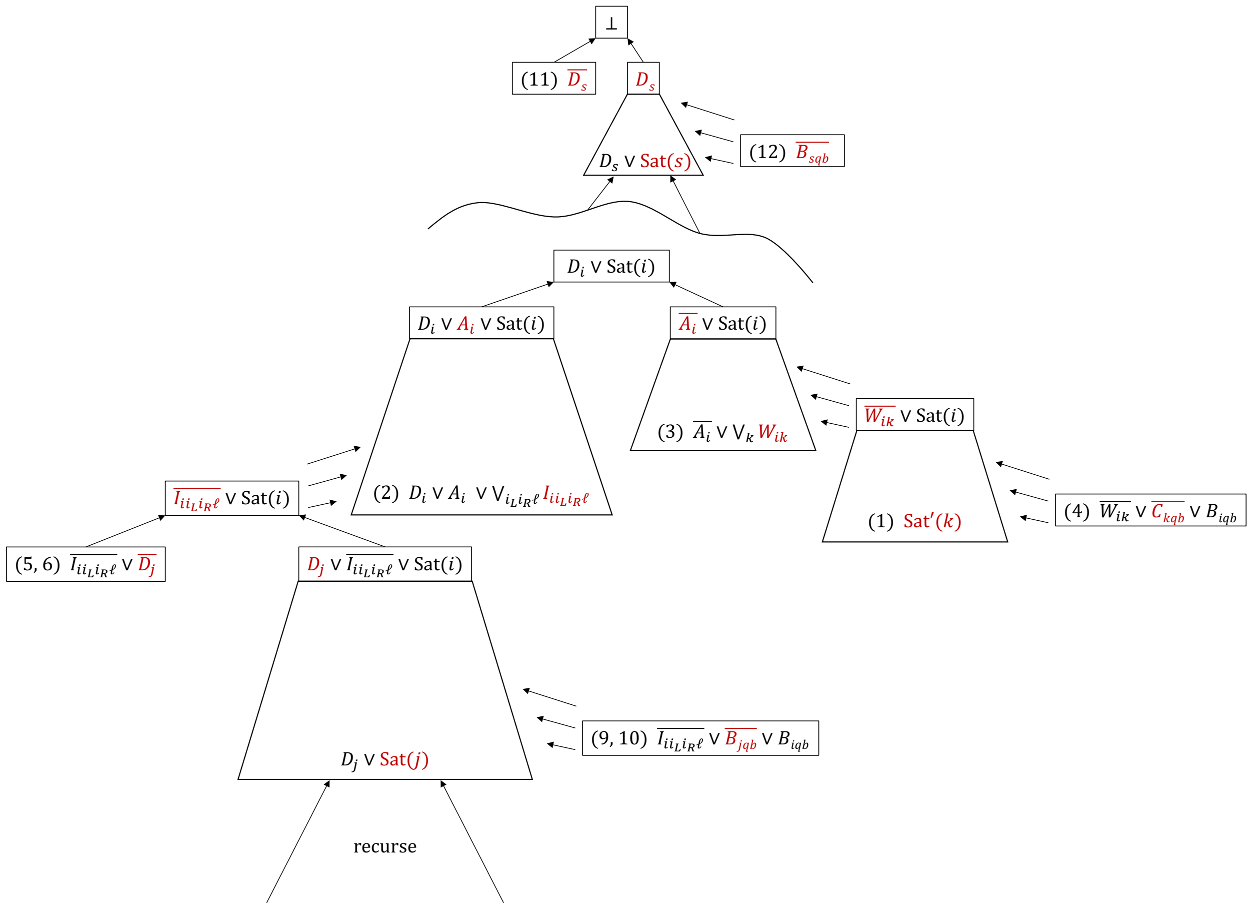
$$\begin{gathered} B_{j,q,b} \\ D_j \\ I_{i,i_L,i_R,\ell} \\ A_i \end{gathered}$$

or we will respect the following ordering:

$$\begin{gathered} C_{k,q,b} \\ W_{i,k} \\ A_i \end{gathered}$$

In either case, we respect our total order, though where the variables belonging to each $\mathcal{B}_j$ block are resolved on is split between the end of their own j-th step and the beginning of the i-th step that follows.

We will proceed in our reduction case-wise. It may be helpful to refer to Fig. 1 below, especially to see how the order is respected.



$\perp$
(11) $\overline{D_s}$
D_s
(12) $\overline{B_{sqb}}$
$D_s \vee \mathrm{Sat}(s)$
$D_i \vee \mathrm{Sat}(i)$
$D_i \vee A_i \vee \mathrm{Sat}(i)$
$\overline{A_i} \vee \mathrm{Sat}(i)$
(3) $\overline{A_i} \vee \bigvee_k W_{ik}$
$\overline{W_{ik}} \vee \mathrm{Sat}(i)$
(1) $\mathrm{Sat}'(k)$
(4) $\overline{W_{ik}} \vee \overline{C_{kqb}} \vee B_{iqb}$
(2) $D_i \vee A_i \vee \bigvee_{i_L i_R \ell} I_{ii_L i_R \ell}$
$\overline{I_{ii_L i_R \ell}} \vee \mathrm{Sat}(i)$
(5, 6) $\overline{I_{ii_L i_R \ell}} \vee \overline{D_j}$
$D_j \vee \overline{I_{ii_L i_R \ell}} \vee \mathrm{Sat}(i)$
(9, 10) $\overline{I_{ii_L i_R \ell}} \vee \overline{B_{jqb}} \vee B_{iqb}$
$D_j \vee \mathrm{Sat}(j)$
recurse

Figure 1: This diagram is a schematic representation of the DAG of the refutation we describe, in an orientation that corresponds naturally to the querying view of the refutation. Note that it presents a top-down view of the refutation, while the main text describes the refutation in a bottom-up manner. Thus the natural way of viewing the order is reversed. Sat(i) represents $\bigvee_{q\in P} B_{i,q,1} \vee \bigvee_{q\in N} B_{i,q,0}$, and Sat$'(k)$ represents $\bigvee_{q\in P} C_{k,q,1} \vee \bigvee_{q\in N} C_{k,q,0}$. Axioms are numbered as in Table 2. The variables being resolved upon (queried, when viewed top-down) in each step are in red. Three arrows coming from a block represents resolving in sequence versions of the clause produced by varying over the appropriate indices. The index $j = i_L$ or i_R depending on whether $\ell \in N$ or P respectively, leading the appropriate of the pair of axioms indicated to be invoked. Notice that $D_j \vee \mathrm{Sat}(j)$ is used by all $i > j$, but only once in any single path through the DAG. For the base case $i = 1$, only the axiom side exists, thereby terminating our recursion.

5.1 Case 1: Derived

Our goal in this first subsection is to derive

$$\overline{I_{i,i_L,i_R,\ell}} \vee \bigvee_{q\in P} B_{i,q,1} \vee \bigvee_{q\in N} B_{i,q,0},$$

which corresponds to the case that $\mathcal{B}_i$ was derived from $\mathcal{B}_{i_L}$ and $\mathcal{B}_{i_R}$ by resolving on x_ℓ for some $1 \le i_L, i_R < i$, $\ell \in [n]$. This clause says that if $\mathcal{B}_i$ was derived from $\mathcal{B}_{i_L}$ and $\mathcal{B}_{i_R}$ by resolving on x_ℓ, then it must contain a literal which is satisfied by the assignment. To decide how to derive this, we check whether $\ell \in P$ or $\ell \in N$. This is because the literal x_ℓ or $\overline{x_\ell}$ may have been causing $\mathcal{B}_{i_L}$ or $\mathcal{B}_{i_R}$ to be satisfied by the assignment respectively, but not both at once, so $\mathcal{B}_i$ will inherit its satisfaction from whichever cannot have been satisfied by the assignment to x_ℓ.

Thus if $\ell \in N$, then we use our inductive hypothesis for $i_L < i$:

$$D_{i_L} \vee \bigvee_{q\in P} B_{i_L,q,1} \vee \bigvee_{q\in N} B_{i_L,q,0}.$$

Since $\ell \in N$, we can transform each of these $B_{i_L,q,b}$'s into $B_{i,q,b}$'s by resolving with each

$$(9) \quad \overline{I_{i,i_L,i_R,\ell}} \vee \overline{B_{i_L,q,b}} \vee B_{i,q,b}$$

for all $q \in [n]$ with $b = 1$ if $q \in P$ and $b = 0$ if $q \in N$ in sequence—without needing to call on the $q = \ell$ and $b = 1$ case, which we do not have for i_L. We then resolve on D_{i_L}:

$$\frac{(5) \quad \overline{I_{i,i_L,i_R,\ell}} \vee \overline{D_{i_L}} \qquad D_{i_L} \vee \overline{I_{i,i_L,i_R,\ell}} \vee \bigvee_{q\in P} B_{i,q,1} \vee \bigvee_{q\in N} B_{i,q,0}}{\overline{I_{i,i_L,i_R,\ell}} \vee \bigvee_{q\in P} B_{i,q,1} \vee \bigvee_{q\in N} B_{i,q,0}}$$

obtaining our goal.

On the other hand, if $\ell \in P$, then we proceed identically but with i_R instead of i_L in order to derive the same clause, which we can again safely do by this protocol because we are not asked to use the non-existent axiom

$$\overline{I_{i,i_L,i_R,\ell}} \vee \overline{B_{i_R,\ell,0}} \vee B_{i,\ell,0}.$$

We repeat this process for all $1 \le i_L, i_R < i$, $\ell \in [n]$, finishing with the clauses

$$\overline{I_{i,i_L,i_R,\ell}} \vee \bigvee_{q\in P} B_{i,q,1} \vee \bigvee_{q\in N} B_{i,q,0}$$

for all possible combinations of these indices. Notice that this process has contributed no more than $s^2 n \cdot (1+n)$ derived clauses to the size of our refutation, and all clauses involved have block-width at most 2. Also notice that we have respected our desired order within this case because after using the inductive hypothesis for either $j = i_L$ or $i_R < i$, we resolve on the $B_{j,q,b}$ and then D_j.

5.2 Case 2: Weakened Axiom

Now we will address the case that $\mathcal{B}_i$ is the weakening of an input axiom of F and derive

$$\overline{A_i} \vee \bigvee_{q \in P} B_{i,q,1} \vee \bigvee_{q \in N} B_{i,q,0},$$

which says that if $\mathcal{B}_i$ is a weakening of an axiom, then it must contain a literal which is satisfied by the assignment. In this case, we do not need to appeal to our inductive hypothesis because this follows straightforwardly from P and N reflecting an assignment that satisfies all of the input clauses of F.

First, notice that since P, N reflect the satisfying assignment,

$$\bigvee_{q \in P} C_{k,q,1} \vee \bigvee_{q \in N} C_{k,q,0}$$

is a weakening of one of the (1) $C_{k,q,b}$ axioms describing what is in $\mathcal{C}_k \in F$, since it must contain some literal which is satisfied by the assignment.

Then we can resolve this with each appropriate

$$(4) \quad \overline{W_{i,k}} \vee \overline{C_{k,q,b}} \vee B_{i,q,b}$$

(for all $q \in [n]$ with $b = 1$ if $q \in P$ and $b = 0$ if $q \in N$) in sequence to transform each of these $C_{k,q,b}$'s into $B_{i,q,b}$'s, ultimately obtaining

$$\overline{W_{i,k}} \vee \bigvee_{q \in P} B_{i,q,1} \vee \bigvee_{q \in N} B_{i,q,0},$$

which says that if $\mathcal{B}_i$ is the weakening of input axiom $\mathcal{C}_k$, then it must contain a literal which is satisfied by the assignment.

We do the same process for all $k \in [m]$, and then resolve them each with

$$(3) \quad \overline{A_i} \vee \bigvee_{k \in [m]} W_{i,k}$$

in sequence to obtain

$$\overline{A_i} \vee \bigvee_{q \in P} B_{i,q,1} \vee \bigvee_{q \in N} B_{i,q,0},$$

as desired. Notice that this case has contributed no more than $m \cdot (n+1)$ derived clauses to the size of our refutation, and all clauses involved have block-width at most 2. Also notice that we have respected our desired order within this case because we resolve on the $C_{k,q,b}$ followed by the $W_{i,k}$.

5.3 Combining the cases

Since $\mathcal{B}_i$ must either be disabled, the weakening of an axiom, or derived and we have just shown that the latter two cases imply that it must contain a literal which is satisfied by the assignment, our final result is straightforward to derive from here.

First we resolve

$$(2) \quad D_i \vee A_i \vee \bigvee_{\substack{1 \le i_L, i_R < i \\ \ell \in [n]}} I_{i,i_L,i_R,\ell}$$

with each

$$\overline{I_{i,i_L,i_R,\ell}} \vee \bigvee_{q \in P} B_{i,q,1} \vee \bigvee_{q \in N} B_{i,q,0}$$

for all $1 \le i_R, i_L < i$, $\ell \in [n]$ in sequence to obtain

$$D_i \vee A_i \vee \bigvee_{q \in P} B_{i,q,1} \vee \bigvee_{q \in N} B_{i,q,0}.$$

Now we resolve on A_i to derive our goal:

$$\frac{D_i \vee A_i \vee \bigvee_{q\in P} B_{i,q,1} \vee \bigvee_{q \in N} B_{i,q,0} \qquad \overline{A_i} \vee \bigvee_{q\in P} B_{i,q,1} \vee \bigvee_{q \in N} B_{i,q,0}}{D_i \vee \bigvee_{q\in P} B_{i,q,1} \vee \bigvee_{q \in N} B_{i,q,0}}$$

Notice that this combination has contributed no more than $s^2 n + 1$ derived clauses to the size of our refutation, and all clauses involved have block-width 1.

Now we must consider how to deal with the base case where $i = 1$. But we have already shown this! Since there are no $i_L, i_R < i$, we simply have the axiom

$$D_1 \vee A_1,$$

so we skip the first part of the process and simply use the axiom case, which did not rely on the inductive hypothesis.

Let us consider our overall size and block-width. Summing over all $1 \le i \le s$ as well as the extra steps at the end to get to the empty clause from the $i = s$ result, the size of our refutation is bounded by a constant times the number of derived clauses

$$s \cdot (s^2 n \cdot (1+n) + m \cdot (n+1) + s^2 n + 1) + 1 + n \in n^{O(1)}$$

for $s = n^3$ and $m = \text{poly}(n)$. Our block-width was also bounded by $2 \in O(1)$ throughout.

Also notice that we have respected our desired order throughout the refutation, along both paths which merged when we resolved on A_i. The Derived path came from a previous $j < i$ step and respected the following order:

$$\begin{gathered} B_{j,q,b} \\ D_j \\ I_{i,i_L,i_R,\ell} \\ A_i \end{gathered}$$

while the Weakened Axiom path respected the order:

$$C_{k,q,b}$$
$$W_{i,k}$$
$$A_i$$

Therefore we have maintained order throughout our refutation as desired.

6 Maintenance of regularity and order in the refutation of Lift(Ref(F))

Taking into account Appendix A, which shows that the refutation of our encoding of Ref(F) can be slightly adjusted to refute the encoding given in [AM19], we inherit their upper bound in the case that F is unsatisfiable and thus have shown Theorem 1.2. However, in order to use the same formula to do so as for $\mathbf{S} = \mathbf{NS}, \mathbf{PC}, \mathbf{SA}$ as well, one issue remains.

In [GNP+20], the lower bound in the case that F is unsatisfiable is obtained by lifting a lower bound on block-width to one on size, and thus uses $\mathcal{A}(F) = \text{Lift}(\text{Ref}(F))$. For a CNF G, Lift(G) is defined by creating two variables x_i^0 and x_i^1 for each variable x_i of G as well as a selector variable s_B for each block and using the gadget $g(x_i^0, x_i^1, s_B) = x_i^{s_B}$ for x_i in block B to substitute for x_i in the original formula. It turns out that if G has $O(1)$ block-width (like Ref(F) does), then Lift(G) can be constructed in polynomial time from G. See [GNP+20] for the full details.

So far, after inheriting part (ii), we have shown part (i) of the following lemma, which is an extension Lemma 2.1 in [GNP+20].

Lemma 6.1. *There is a polynomial-time algorithm that on the input of a $O(1)$ block-width CNF formula F outputs a $O(1)$ block-width CNF formula Ref(F) such that*

(i) If F is satisfiable, then Ref(F) admits a size-$n^{O(1)}$ $O(1)$ block-width $\textbf{Res}_{\textbf{ord}}$ refutation that is also block-ordered.

*(ii) If F is unsatisfiable, then Ref(F) requires a **Res** refutation of block width at least $n^{\Omega(1)}$.*

Now we must ensure that our block-ordered, polynomial-size, and $O(1)$-block-width refutation of Ref(F) (which Appendix A shows can be slightly adjusted to refute the encoding given by [GNP+20]) implies that there is also a regular and even ordered refutation for Lift(Ref(F)) of polynomial size in the F satisfiable case, while in the unsatifiable case we get exponential size. We will do this by proving the following lemma, which is an extension of Lemma 2.3 in [GNP+20], which shows this lift with general Resolution given a general Resolution refutation. In order for the lift to maintain regularity and order, we need block-regularity and block-order instead of simply regularity and order in the original refutation.

Lemma 6.2. *There is a polynomial-time algorithm that on the input of a $O(1)$ block-width CNF formula G outputs a CNF formula Lift(G) such that*

$$2^{\Omega(bw(G \vdash \bot))} \leq \textbf{Res}_{\textbf{reg}}(\textit{Lift}(G)) \leq 2^{O(bw(\mathcal{P}))} \cdot ||\mathcal{P}||$$

where $\mathcal{P}$ is any block-regular Resolution refutation of G. Additionally, Lift(G) is such that

$$2^{\Omega(bw(G \vdash \bot))} \leq \boldsymbol{Res_{ord}}(\mathit{Lift}(G)) \leq 2^{O(bw(\mathcal{P}))} \cdot ||\mathcal{P}||$$

where $\mathcal{P}$ is any block-ordered Resolution refutation of G.

Proof. Recall from Section 2 that regular Resolution refutations can be characterized as read-once branching programs (which can be generalized to query strategies for general Resolution). It will be useful to use the querying framework for this proof.

We use the same construction of Lift(G) as [GNP+20], and thus inherit their lower bound from the lower bound for general Resolution in both cases. We also use the same construction of a query strategy for Lift(G) from that for G as [GNP+20] does, so we also inherit their upper bound as long as we can ensure regularity and order in each case.

Following their construction, given a regular query strategy for G, we construct one for Lift(G) as follows. Whenever the strategy for G queries x_i in block B, if s_B is not yet queried then we query it and then query the appropriate $x_i^{s_B}$ based on the value of s_B. We then proceed as if the strategy for G received the result for x_i that we received for $x_i^{s_B}$. Additionally, whenever the strategy for G forgets some subset of variables, we forget all variables associated with them unless they are needed to determine the appropriate $x_i^{s_B}$ to use for a variable x_i that we still remember (i.e., we remember all of the s_B variables for blocks that contain variables the strategy for G is still remembering). It is easy to see that this strategy solves the conflict cause search problem for Lift(G), because if the strategy for G gets to a conflict clause C we will get to the one for Lift(G) that was created by substituting the $g(x_i^0, x_i^1, s_B)$ gadgets into C. It is also easy to see that we get a $O(2^{bw(\mathcal{P})})$ blowup compared to the original size, because we only ever need to have $2^{bw(\mathcal{P})}$ versions of a node in the original strategy for all possible assignments to the at most $bw(\mathcal{P})$ s_B variables we ever need to know at once.

Now, let $\mathcal{P}$ be a block-regular Resolution refutation of G. Notice that the Resolution refutation of Lift(G), or query strategy for Lift(G), remains regular for two reasons. First, because the original refutation was regular, the new refutation is regular with respect to the x_i^0, x_i^1 variables. Second, because the original refutation was block-regular, we can safely forget a selector s_B variable when the protocol indicates without concern of requerying it later. This is because if after querying some of the variables associated with a block we forget all of them (which must occur after querying a variable from a different block), we know we will not query another variable from the same block later due to the original proof's block-regularity and so will not query s_B again either. Thus regularity is maintained.

It is easy to see how this also maintains order if we let $\mathcal{P}$ be a block-ordered Resolution refutation of G. Since we use the same construction of a query strategy again, from what we have just shown, we have maintained regularity in our new proof since a block-ordered a refutation must be block-regular as well. Notice that the Resolution refutation of Lift(G) remains ordered as well for two reasons. First, because the original refutation was ordered, the new refutation is ordered with respect to the x_i^0, x_i^1 variables (with arbitrary ordering between x_i^0 and x_i^1 for a given i in our new order). Second, the refutation is also ordered with the s_B included because if we insert each s_B before all of the other variables associated with its block in the ordering, this order will be respected. This is because in our new protocol we always query s_B before we query anything else from the block. Thus order is maintained. □

We now seek to also prove our result using the formula that was used to show Proposition 1.1 for **CP**. In [GKMP20], a slightly different kind of lifting is used to show this result for **CP**. Instead of one selector variable for each block specifying which of two copies of the variables to use, a $O(\log n)$-bit pointer specifies which of the $O(n)$ copies of the variables in the block to use. [GKMP20] shows this variation can be also constructed in polynomial time. In a similar way as Lemma 6.2, we can extend Theorems 4 and 8 from [GKMP20] to show

Lemma 6.3. *There is a polynomial-time algorithm that on the input of a $O(1)$ block-width CNF formula G outputs a CNF formula $Lift_{\mathbf{CP}}(G)$ such that*

$$2^{\Omega(bw(G \vdash \bot))} \leq \boldsymbol{Res_{reg}}(Lift_{\mathbf{CP}}(G)) \leq n^{O(bw(\mathcal{P}))} \cdot ||\mathcal{P}||$$

where $\mathcal{P}$ is any block-regular Resolution refutation of G. Additionally, Lift(G) is such that

$$2^{\Omega(bw(G \vdash \bot))} \leq \boldsymbol{Res_{ord}}(Lift_{\mathbf{CP}}(G)) \leq n^{O(bw(\mathcal{P}))} \cdot ||\mathcal{P}||$$

where $\mathcal{P}$ is any block-ordered Resolution refutation of G.

Proof (sketch). Like before, we inherit the lower bound from [GKMP20]. To establish the upper bound, we use the same query strategy as in the proof of Lemma 6.2 with one adjustment. Instead of querying the single selector variable when the original strategy queries a variable from a new block, we query all of the pointer variables to determine which version of the variable to use. This then causes the size to blow up by a factor of $2^{O(\log n \cdot \mathrm{bw}(\mathcal{P}))} = n^{O(\mathrm{bw}(\mathcal{P}))}$ instead of a factor of $2^{O(1 \cdot \mathrm{bw}(\mathcal{P}))}$. Regularity and order are maintained (under the appropriate assumptions about $\mathcal{P}$) for the same reasons as in Lemma 6.2. □

Notice that the upper bound remains polynomial for constant $\mathrm{bw}(\mathcal{P})$, so this gives us the bounds we need for this version of lifting as well.

7 Conclusion

Combining Lemmas 6.1 and 6.2, we have shown that using $\mathcal{A}(F) = \mathrm{Lift}(\mathrm{Ref}(F))$ suffices to establish Theorem 1.2. Alternatively, we can use the slight variation $\mathcal{A}(F) = \mathrm{Lift}(\mathrm{TreeRef}(F))$ so that all of $\mathbf{Res}, \mathbf{NS}, \mathbf{PC}, \mathbf{SA}, \mathbf{Res_{reg}}, \mathbf{Res_{ord}}$ can be established using the same formula. TreeRef(F) simply has extra axioms ensuring the described refutation of F is tree-like, which are only needed for the upper bound in the **NS** case. By using the same formula as for $\mathbf{Res}, \mathbf{NS}, \mathbf{PC}, \mathbf{SA}, \mathbf{Res_{reg}}$, we have also established Theorem 1.3 except for **CP**. Finally, Lemma 6.3 allows us to extend this result to **CP** as well.

Altogether, we have shown that regular and ordered Resolution are automatable in polynomial time if and only if $\mathsf{P} = \mathsf{NP}$. They are also not automatable in subexponential time unless ETH fails. Additionally, it is not possible to approximate minimum proof length in each system within subexponential error unless $\mathsf{P} = \mathsf{NP}$. Further, the problem of distinguishing between the existence of a polynomial-size ordered Resolution refutation of F and at least exponential-size general Resolution, Nullstellensatz, Polynomial Calculus, and Sherali-Adams proofs being required to refute F is NP-complete. The problem of distinguishing between the existence of a polynomial-size ordered Resolution refutation of F and at least exponential-size Cutting Planes proof being required to refute F is likewise NP-complete.

A major open problem in this area that remains is whether Proposition 1.1 can be extended to include the semi-algebraic proof system Sum of Squares (**SoS**) as well. The technique used in [GNP^{+}20] to extend the result to other algebraic and semi-algebraic systems by showing the exponential lower bound using a reduction from the Pigeonhole Principle will not work for **SoS**, since it does have low-degree proofs of the Pigeonhole Principle. The method [GKMP20] used for **CP** will not work for **SoS** either because even its lifted version of the Pigeonhole Principle is too easy for **SoS**. Thus a new approach would be required to extend these results to **SoS**.

Acknowledgements

I would like to thank my advisor for this project, Paul Beame, for his invaluable guidance.

References

[ABMP01] Michael Alekhnovich, Samuel R. Buss, Shlomo Moran, and Toniann Pitassi. Minimum propositional proof length is NP-hard to linearly approximate. *J. Symb. Log.*, 66(1):171–191, 2001.

[AM19] Albert Atserias and Moritz Müller. Automating resolution is NP-hard. In *60th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2019, Baltimore, Maryland, USA, November 9-12, 2019*, pages 498–509. IEEE Computer Society, 2019.

[AR08] Michael Alekhnovich and Alexander A. Razborov. Resolution is not automatizable unless W[P] is tractable. *SIAM J. Comput.*, 38(4):1347–1363, 2008.

[Ats20] Albert Atserias. Talk: Automating resolution is NP-hard, January 22, 2020. Banff International Research Station (BIRS) Proof Complexity Workshop, 20w5144, Banff, Canada, http://www.birs.ca/events/2020/5-day-workshops/20w5144/videos/watch/202001220955-Atserias.html.

[BDP09] Fahiem Bacchus, Shannon Dalmao, and Toniann Pitassi. Solving #sat and bayesian inference with backtracking search. *J. Artif. Intell. Res.*, 34:391–442, 2009.

[BP96] P. Beame and T. Pitassi. Simplified and improved resolution lower bounds. In *Proceedings 37th Annual Symposium on Foundations of Computer Science*, pages 274–282, Burlington, VT, October 1996. IEEE.

[DP60] M. Davis and H. Putnam. A computing procedure for quantification theory. *Communications of the ACM*, 7:201–215, 1960.

[EGG08] Kord Eickmeyer, Martin Grohe, and Magdalena Grüber. Approximation of natural W[P]-complete minimisation problems is hard. In *Proceedings of the 23rd Annual IEEE Conference on Computational Complexity, CCC 2008, 23-26 June 2008, College Park, Maryland, USA*, pages 8–18. IEEE Computer Society, 2008.

[GKMP20] Mika Göös, Sajin Koroth, Ian Mertz, and Toniann Pitassi. Automating cutting planes is NP-hard. In *Proccedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing, STOC 2020, Chicago, IL, USA, June 22-26, 2020*, pages 68–77. ACM, 2020.

[GNP$^+$20] Mika Göös, Jakob Nordström, Toniann Pitassi, Robert Robere, Dmitry Sokolov, and Susanna F. de Rezende. Automating algebraic proof systems is NP-hard. *Electronic Colloquium on Computational Complexity (ECCC)*, 27:64, 2020.

[Iwa97] Kazuo Iwama. Complexity of finding short resolution proofs. In Igor Prívara and Peter Ruzicka, editors, *Mathematical Foundations of Computer Science 1997, 22nd International Symposium, MFCS'97, Bratislava, Slovakia, August 25-29, 1997, Proceedings*, volume 1295 of *Lecture Notes in Computer Science*, pages 309–318. Springer, 1997.

[Kra95] Jan Krajícek. *Bounded arithmetic, propositional logic, and complexity theory*, volume 60 of *Encyclopedia of mathematics and its applications*. Cambridge University Press, 1995.

[LNNW95] László Lovász, Moni Naor, Ilan Newman, and Avi Wigderson. Search problems in the decision tree model. *SIAM J. Discret. Math.*, 8(1):119–132, 1995.

[MPW19] Ian Mertz, Toniann Pitassi, and Yuanhao Wei. Short proofs are hard to find. In Christel Baier, Ioannis Chatzigiannakis, Paola Flocchini, and Stefano Leonardi, editors, *46th International Colloquium on Automata, Languages, and Programming, ICALP 2019, July 9-12, 2019, Patras, Greece*, volume 132 of *LIPIcs*, pages 84:1–84:16. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019.

[Urq95] A. Urquhart. The complexity of propositional proofs. *Bulletin of Symbolic Logic*, 1(4):425–467, December 1995.

A Essential equivalence of our formalization of Ref(F) to previous ones

In the main text, we have used a slightly different variable and axiom encoding for the semantic meaning of Ref(F) than [AM19] or [GNP$^+$20]. This was done in order to more naturally describe the refutation of F and make it easier to see what was happening in the refutation. In this appendix, we describe these differences and outline how our ordered Resolution refutation above could be modified for the encoding of Ref(F) of [AM19], and modified a bit further again to address the encoding of Ref(F) given by [GNP$^+$20], which also differs slightly from that of [AM19], so that we can utilize their lower bounds in the case that F is unsatisfiable.

A.1 Adjustments for the [AM19] encoding

There are a few differences between our encoding and that of [AM19]. See their paper for their full formalization of Ref(F) (actually what they refer to as RRef(F)), but the differences from ours in terms of variables and axioms can be summarized as follows.

Differences between our variables

1. They do not have $C_{k,q,b}$ variables. Their values from the (1) $C_{k,q,b}$ axioms have already been substituted into the (4) $\overline{W_{i,k}} \vee \overline{C_{k,q,b}} \vee B_{i,q,b}$ axioms.
2. Instead of D_i, they have a variable that means the opposite (i.e. that a clause is active, instead of that it is disabled). This is immaterial, so we will continue using D_i in what follows.
3. They do not have A_i variables. Instead, there is are $W_{i,k}$ for $k = 0$ to set the axiom from which $\mathcal{B}_i$ was weakened to null. Thus A_i has the same meaning as $\overline{W_{i,0}}$, so we will continue to use A_i in the place of $\overline{W_{i,0}}$.
4. Instead of one $I_{i,i_L,i_R,\ell}$ variable to express that clause $\mathcal{B}_i$ is inferred from $\mathcal{B}_{i_L}$ and $\mathcal{B}_{i_R}$ by resolving on x_ℓ, there are three separate variables L_{i,i_L}, R_{i,i_R}, and $V_{i,\ell}$ to indicate the left premise, the right premise, and the variable that is resolved upon to derive $\mathcal{B}_i$ respectively. Thus $I_{i,i_L,i_R,\ell}$ has the same meaning as $L_{i,i_L} \wedge R_{i,i_R} \wedge V_{i,\ell}$. Further, these can also be set to null by extra variables associated with the index $i_L = 0$, $i_R = 0$, or $\ell = 0$ being true respectively.

[AM19] has $2s^2 + 3sn + sm + 5s$ variables in total, while we use $\frac{1}{3}s^3n - \frac{1}{2}s^2n + \frac{13}{6}sn + sm + 2s + 2mn$.

Differences between our axioms

1. (1) and (1') are of course missing.

2. Instead of (2), they have:

$$D_i \vee \bigvee_{i_L \in \{0\} \cup [s]} L_{i,i_L},$$

$$D_i \vee \bigvee_{i_R \in \{0\} \cup [s]} R_{i,i_R},$$

   and

$$D_i \vee \bigvee_{\ell \in \{0\} \cup [n]} V_{i,\ell}.$$

   They also have

$$D_i \vee \overline{L_{i,i_L}}$$

   for $i_L \geq i$ and

$$D_i \vee \overline{R_{i,i_R}}$$

   for $i_R \geq i$, which we can resolve with the first two to get

$$D_i \vee \bigvee_{0 \leq i_L < i} L_{i,i_L}$$

   and

$$D_i \vee \bigvee_{0 \leq i_R < i} R_{i,i_R}.$$

   Finally, they have

$$D_i \vee A_i \vee \overline{L_{i,0}},$$
$$D_i \vee A_i \vee \overline{R_{i,0}},$$

   and

$$D_i \vee A_i \vee \overline{V_{i,0}}$$

   to ensure that if if a clause is not disabled or the weakening of an axiom, it is derived from something. We can resolve these with the appropriate clauses just mentioned to derive

$$D_i \vee A_i \vee \bigvee_{1 \leq i_L < i} L_{i,i_L},$$

$$D_i \vee A_i \vee \bigvee_{1 \leq i_R < i} R_{i,i_R},$$

   and

$$D_i \vee A_i \vee \bigvee_{\ell \in [n]} V_{i,\ell}$$

   which are simply (2) split into three parts.

3. Their versions of (3) through (10) have D_i added to the OR (in addition to the variations described below). (7) and (9) have D_{i_L} added. (8) and (10) have D_{i_R} added. (12) has D_s added.

4. In (4), they have the values of the $C_{k,q,b}$ already substituted in as mentioned before. Clearly, these differences due to the lack of $C_{k,q,b}$ variables simply allow us to do a little less work in the Weakened Axiom Case.
5. In (5), they have $\overline{L_{i,i_L}}$ in the OR instead of $\overline{I_{i,i_L,i_R,\ell}}$.
6. In (6), they have $\overline{R_{i,i_R}}$ in the OR instead of $\overline{I_{i,i_L,i_R,\ell}}$.
7. In (7) and (9), they have $\overline{L_{i,i_L}}$ and $\overline{V_{i,\ell}}$ in the OR instead of $\overline{I_{i,i_L,i_R,\ell}}$.
8. In (8) and (10), they have $\overline{R_{i,i_R}}$ and $\overline{V_{i,\ell}}$ in the OR instead of $\overline{I_{i,i_L,i_R,\ell}}$.
9. They have several additional axioms that go unused in our refutation.

[AM19] has $2s^3 + 4s^2n^2 - 2s^2n + 5s^2 + sn^2 + sm^2 + 2sn + sm + 9s + 2n + 1$ with as many as $2smn$ additional axioms in total (depending on how many x_i are in the input axioms), while we use $\frac{4}{3}s^4n^2 + \frac{2}{3}s^4n - 2s^3n^2 - s^3n + \frac{2}{3}s^2n^2 + \frac{1}{3}s^2n + 2snm + 2s + 2nm + 2n + 1$.

Adjusted order It is not too difficult to see that while resulting in a slightly messier refutation and order, we can adjust our refutation to address these differences and still maintain a block-order. We will respect the same order between blocks, and respect this order on variable types within a block:

$$\begin{gathered} R_{i,i_R} \text{ for } i_R \geq i \\ R_{i,0} \\ R_{i,i_R} \text{ for } i_R < i \\ L_{i,i_L} \text{ for } i_L \geq i \\ L_{i,0} \\ L_{i,i_L} \text{ for } i_L < i \\ V_{i,0} \\ V_{i,\ell} \text{ for } \ell \neq 0 \\ W_{i,k} \\ A_i \\ B_{i,q,b} \\ D_i \end{gathered}$$

Again, the specific order within variable types is immaterial because we would resolve on all of the variables of a type one after the other in an arbitrary sequence. It is easy to see that polynomial size, constant block-width, and the order described can be maintained throughout the adjusted refutation.

A.2 Adjustments for the [GNP+20] encoding

The difference in the encoding of Ref(F) in [GNP+20] is that instead of indicator variables for whether $\mathcal{B}_i$ is derived from $\mathcal{B}_{i_L}$, $\mathcal{B}_{i_R}$ by resolving on x_ℓ and whether it is the weakening of $\mathcal{C}_k \in F$, there are $2\log s + \log n$ and $\log m$ variables respectively encoding in binary what $\mathcal{B}_i$ is derived from or weakened from (which relationship is enforced depends on A_i). All of these are considered to be in the $\mathcal{B}_i$ block of variables. Once we set $s = n^3$ and $m = \text{poly}(n)$, we see that after querying A_i we simply must query $O(\log n)$ instead of $\text{poly}(n)$ variables to deduce what $\mathcal{B}_i$ is derived from (if $A_i = 0$) or weakened from (if $A_i = 1$), so this change only decreases the size of our refutation and leaves the block-width unchanged. Finally, in our order we can simply the replace the $I_{i,i_L,i_R,\ell}$ variable type with the $O(\log n)$ variables that tell us what $\mathcal{B}_i$ is derived from in any order amongst themselves (we can also separate these out in the manner of [AM19] and associate a subset of these variables with the L_{i,i_L}, R_{i,i_R}, and $V_{i,\ell}$ variables in the order above) and similarly replace the $W_{i,k}$ variable type with the $O(\log n)$ variables that tell us what $\mathcal{B}_i$ is weakened from, again in any order amongst themselves.