Limits of CDCL Learning via Merge Resolution

Marc Vinyals
University of Auckland, New Zealand

Chunxiao Li
University of Waterloo, Canada

Noah Fleming
Memorial University of Newfoundland, Canada

Antonina Kolokolova
Memorial University of Newfoundland, Canada

Vijay Ganesh
University of Waterloo, Canada

Abstract

In their seminal work, Atserias et al. and independently Pipatsrisawat and Darwiche in 2009 showed that CDCL solvers can simulate resolution proofs with polynomial overhead. However, previous work does not address the tightness of the simulation, i.e., the question of how large this overhead needs to be. In this paper, we address this question by focusing on an important property of proofs generated by CDCL solvers that employ *standard learning schemes*, namely that the derivation of a learned clause has at least one inference where a literal appears in both premises (aka, a merge literal). Specifically, we show that proofs of this kind can simulate resolution proofs with at most a linear overhead, but there also exist formulas where such overhead is necessary or, more precisely, that there exist formulas with resolution proofs of linear length that require quadratic CDCL proofs.

2012 ACM Subject Classification Theory of computation → Proof complexity; Theory of computation → Constraint and logic programming

Keywords and phrases proof complexity, resolution, merge resolution, CDCL, learning scheme

Digital Object Identifier 10.4230/LIPIcs.SAT.2023.27

Related Version *Full Version*: `https://arxiv.org/abs/2304.09422`

Acknowledgements The authors are grateful to Yuval Filmus and a long list of participants in the program *Satisfiability: Theory, Practice, and Beyond* at the Simons Institute for the Theory of Computing for numerous discussions. We also thank the SAT 2023 reviewers for their helpful suggestions. This work was done in part while the authors were visiting the Simons Institute for the Theory of Computing.

1 Introduction

Over the last two decades, CDCL SAT solvers have had a dramatic impact on many areas of software engineering [10], security [13, 28], and AI [7]. This is due to their ability to solve very large real-world formulas that contain upwards of millions of variables and clauses [17]. Both theorists and practitioners have expended considerable effort in understanding the CDCL algorithm and the reasons for its unreasonable effectiveness in the context of practical applications. While considerable progress has been made, many questions remain unanswered.

Perhaps the most successful set of tools for understanding the CDCL algorithm come from proof complexity, and a highly influential result is the one that shows that idealized models of CDCL can polynomially simulate the resolution proof system, proved independently by Atserias, Fichte, and Thurley [2], and Pipatsrisawat and Darwiche [23], building on initial results by Beame et al. [5] and Hertel et al. [15]. (See also a recent alternative proof by

Beyersdorff and Böhm [6].) Such simulation results are very useful because they reassure us that whenever a formula has a short resolution proof then CDCL with the right choice of heuristics can reproduce it.

Recent models make assumptions that are closer to real solvers, but pay for that with a polynomial overhead in the simulation. A series of papers have focused on understanding which of the assumptions are needed for these simulations to hold, often using and/or introducing refinements of resolution along the way. For instance, the question of whether restarts are needed, while still open, has been investigated at length, and the pool resolution [26] and RTL [9] proof systems were devised to capture proofs produced by CDCL solvers that do not restart. The importance of decision heuristics has also been explored recently, with results showing that neither static [21] nor VSIDS-like [27] ordering of variables are enough to simulate resolution in full generality (unless VSIDS scores are periodically erased [16]). In the case of static ordering, the (semi-)ordered resolution proof system [21] was used to reason about such variants of CDCL solvers.

But even if we stay within the idealized model, it is not clear how efficient CDCL is in simulating resolution. The analysis of Pipatsrisawat and Darwiche gives an $\mathrm{O}(n^4)$ overhead—that is, if a formula over n variables has a resolution refutation of length L, then a CDCL proof with no more than $\mathrm{O}(n^4 L)$ steps exists. Beyersdorff and Böhm [6] improved the overhead to $\mathrm{O}(n^3)$, but we do not know what the optimal is. Furthermore, to the best of our knowledge, prior to our paper, we did not even know if the overhead can be avoided altogether.

1.1 Learning Schemes in CDCL and Connection with Merges

A common feature of CDCL solvers is the use of 1-empowering learning schemes [22, 2]: that is, they only learn clauses which enable unit propagations that were not possible before. An example of 1-empowering learning scheme is the popular First Unique Implication Point (1UIP) learning scheme [18]. To model this behavior we build upon a connection between 1-empowerment, and merges [1], i.e., resolution steps involving clauses with shared literals.

Nearly every CDCL solver nowadays uses the 1UIP learning scheme, where conflict analysis starts with a clause falsified by the current state of the solver and sequentially resolves it with clauses responsible for unit propagations leading to the conflict, until the clause becomes *asserting*, i.e., unit immediately upon backjumping.

Descriptions of early implementations of CDCL solvers [18, 20] already remark on the importance of learning an asserting clause, since that nudges the solver towards another part of the search space, and consequently early alternative learning schemes explored learning many kinds of asserting clauses. First observe that conflict analysis can be extended to produce other asserting clauses that appear after the 1UIP during conflict analysis such as intermediate UIPs and the last UIP [4]. The early solver GRASP can even learn multiple UIP clauses from a single conflict. While there is empirical evidence that it is often best to stop conflict analysis at the 1UIP [29], recent work has identified conditions where it is advantageous to continue past it [14] (see also the discussion of learning schemes therein).

Ryan [24, §2.5] also observed empirically that clause quality is negatively correlated with the length of the conflict analysis derivation and considered the opposite approach, that is, learning clauses that appear before the 1UIP during conflict analysis in addition to the 1UIP. This approach is claimed to be useful for some empirical benchmarks but, like any scheme that learns multiple clauses, slows down Boolean constraint propagation (BCP) in comparison to a scheme that learns just the 1UIP.

Later works provide a more theoretically oriented approach to understanding the strength of 1UIP and to learning clauses that appear before the 1UIP [12, 22]. In particular, and highly relevant for our discussion, Pipatsrisawat and Darwiche identified 1-empowerment as a fundamental property of asserting clauses. Furthermore they identified a connection between 1-empowering clauses and merges, and used the simplicity of checking for merges as an approximation for 1-empowerment.

An orthogonal approach is to extend the 1UIP derivation by resolving it with clauses other than those that would usually be used during conflict analysis [3]. A prominent example is clause minimization [25], where literals are eliminated from the 1UIP clause by resolving it with the appropriate input clauses, independently of their role in the conflict, so the resultant clause that is actually learned is a shorter and therefore stronger version of the 1UIP.

Furthermore, a relation between merges and unit-resolution completeness has also been observed in the context of knowledge compilation [11]. Finally, the amount of merges directly inferable from a formula (i.e., in a single resolution step) has been proposed, under the name of mergeability, as a measure to help explain the hardness of a formula based on both controlled experiments as well as analysis of real-world instances [30].

To summarize, merges are relevant in the context of CDCL learning schemes for the following reason: all practical CDCL learning schemes either produce a 1-empowering clause or extend one, and since 1-empowering clauses always contain a merge in its derivation, we have that all practical learning schemes produce a clause that contains a merge in its derivation, which is exactly the property imposed by the proof systems we introduce below.

1.2 Our Contributions

As mentioned earlier, we build upon a connection between 1-empowerment and merges [22, 2], and introduce a DAG-like version of Andrews' merge resolution which includes CDCL with an arbitrary 1-empowering learning scheme. This is because for any 1-empowering clause, at least one step in its resolution derivation must resolve two clauses that share a common literal: a *merge* step in the sense of Andrews [1]. This is precisely the condition that our merge resolution proof system enforces. Clause minimization procedures, as long as they are applied on top of 1-empowering clauses, are also modelled by merge resolution.

We prove that, on the one hand, merge resolution is able to simulate resolution only with a linear overhead. On the other hand, we show a quadratic separation between resolution and merge resolution, that is there exist formulas with resolution proofs of linear length that require merge resolution proofs of quadratic length. The practical consequence of this pair of results is that CDCL may be polynomially worse than resolution because of the properties of a standard learning scheme, but the blow-up due to these properties is not more than linear.

We also consider weaker proof systems, all of which contain 1UIP (and do so with finer granularity), but not necessarily other asserting learning schemes. A technical point of interest is that we work with proof systems that are provably not closed under restrictions, which is unusual in proof complexity. This fact forces our proof to exploit syntactic properties of the proof system, as opposed to relying on more convenient semantic properties.

2 Preliminaries

A literal is either a variable $x^1 = x$ or its negation $x^0 = \overline{x}$. A clause is a disjunction of literals, and a CNF formula is a conjunction of clauses. The support of a clause or vars(C) is the set of variables it contains. A resolution derivation from a formula F is a sequence of clauses $\eta = C_1, \ldots, C_L$ such that C_i is either an axiom in F or it is the conclusion of applying the resolution rule

$$\mathrm{Res}(A \vee x, B \vee \overline{x}, x) = \mathrm{Res}(A \vee x, B \vee \overline{x}) = A \vee B$$

on two premises C_j, C_k with $j, k < i$. The unique variable x that appears with opposite signs in the premises of a resolution inference is called the pivot. If furthermore there is a *literal* common to A and B the resolvent is called a merge. If instead of being the result of a syntactic inference we allow C_i to be any clause semantically implied by C_j and C_k, even if C_j and C_k might not be resolvable, then we say η is a semantic resolution derivation. A derivation is a refutation if its last clause is the empty clause $\bot$. We denote $\eta[a, b] = \{C_i \in \eta \mid i \in [a, b]\}$.

We assume that every clause in a derivation is annotated with the premises it is obtained from, which allows us to treat the proof as a DAG where vertices are clauses and edges point from premises to conclusions. When this DAG is a tree we call a derivation tree-like, and when it is a centipede (i.e., a maximally unbalanced tree) we call it input.

A derivation is unit if in every inference at least one of the premises is a unit clause consisting of a single literal. Since neither input nor unit resolution are complete proof systems, we write $F \vdash_i C$ (respectively $F \vdash_1 C$) to indicate that there exists an input (resp. unit) resolution derivation of C from F.

A clause C syntactically depends on an axiom A with respect to a derivation η if there is a path from A to C in the DAG representation of η. This does not imply that A is required to derive C, since a different derivation might not use A.

A restriction—or more formally a variable substitution, since we allow mapping variables to other variables—is a mapping $\rho\colon X \to X \cup \{0, 1\}$, successively extended to literals, clauses, formulas, and derivations, simplifying where needed. We write $\rho(x) = *$ as a shorthand for $\rho(x) = x$. It is well-known that if η is a resolution derivation from F and ρ is a restriction, then $\eta\restriction_\rho$ is a semantic resolution derivation from $F\restriction_\rho$.

It is convenient to leave satisfied clauses in place in a derivation that is the result of applying a restriction to another derivation so that we can use the same indices to refer to both derivations. To do that we use the symbol 1 and treat it as a clause that always evaluates to true, is not supported on any set, does not depend on any clause, and cannot be syntactically resolved with any clause.

A semantic derivation can be turned into a syntactic derivation by ignoring unnecessary clauses. Formally, if η is a semantic resolution derivation, we define its syntactic equivalent $s(\eta)$ as the syntactic resolution derivation obtained by replacing each clause of $C \in \eta$ by a clause $s(C)$ as follows. If C is an axiom then $s(C) = C$. Otherwise let A and B be the parents of C. If $s(A) \vDash C$ we set $s(C) = s(A)$, analogously with $s(B)$. Otherwise we set $s(C) = \mathrm{Res}(s(A), s(B))$. It is not hard to see that for each $C_i \in \eta$, $s(C_i) \vDash C_i$.

2.1 CDCL

We need to define a few standard concepts from CDCL proofs. An in-depth treatment can be found in the Handbook of Satisfiability [8]. Let F be a CNF formula, to which we refer as a clause database. A trail τ is a sequence of tuples $(x_{j_i} = b, C_i)$ where C_i is either a clause in F or the special symbol d representing a decision. We denote by $\alpha_{<i}$ the assignment $\{x_{j_i} = b \mid i' < i\}$. A trail is valid with respect to F if for every position i that is not a decision we have that $C_i\restriction_{\alpha_{<i}} = x^b_{j_i}$, in which case we say that C_i propagates $x^b_{j_i}$, and if for every decision i there is no clause $C \in F$ such that $C\restriction_{\alpha_{<i}} = x^b$.

We denote by $\mathrm{dl}(i) = \mathrm{dl}(i-1) + [\![C_i = d]\!]$ the decision level at position i, that is the number of decisions up to i. Here $[\![C_i = d]\!]$ is the indicator of the event that the ith variable on the trail was set by a decision. We mark the position of the last decision in a trail by i^*.

A clause C is asserting if it is unit at the last decision in the trail, that is $C\restriction_{\alpha_{<i^*}} = x^b$. It is 1-empowering if C is implied by F and can lead to new unit propagations after being added to F, that is if there exists a literal $\ell \in C$ such that for some $A \in \{\bot, \ell\}$, it holds that $F \wedge \overline{C \setminus \ell} \nvdash_1 A$. If a clause is not 1-empowering then we say it is absorbed by F.

Given a clause $D_{|\tau|}$ falsified by a trail τ, the conflict derivation is an input derivation $D_{|\tau|}, D_{|\tau|-1}, \ldots, D_k$ where $D_{i-1} = \mathrm{Res}(D_i, C_i, x_{j_i})$ if $C_i \neq d$ and $x_{j_i} \in D_i$, and $D_{i-1} = D_i$ otherwise. The first (i.e., with the largest index) asserting clause in the derivation is called the 1UIP. Note that D_{i^*} is always asserting (because D_{i^*} is falsified by $\alpha_{\leq i^*}$ but not by $\alpha_{<i^*}$), therefore we can assume that the 1UIP always has index at least i^*.

We call a sequence of input derivations *input-structured* if the last clause of each derivation can be used as an axiom in successive derivations. The last clause of each but the last derivation is called a lemma. A CDCL derivation is an input-structured sequence of conflict derivations, where learned clauses are lemmas. This definition is similar to that of Resolution Trees with Input Lemmas [9], with the difference that the sequence only needs to be ordered, without imposing any further tree-structure on the global proof.

The following Lemmas highlight the practical relevance of merges by relating them to 1UIP, asserting, and 1-empowering clauses.

▶ **Lemma 1** ([22, Proposition 2]). *If a clause is asserting, then it is 1-empowering.*[1]

▶ **Lemma 2** ([2, Lemma 8]). *If $A \vee x$ and $B \vee \overline{x}$ are absorbed but $A \vee B$ is 1-empowering, then $A \vee B$ is a merge. In particular, if a clause is 1-empowering, then it contains a merge in its derivation.*

▶ **Lemma 3.** *The 1UIP clause is a merge.*

Proof. Let $D_i = \mathrm{Res}(C_{i+1}, D_{i+1})$ be the 1UIP. Every clause in F that is not already satisfied by $\alpha_{<i^*}$, and in particular C_i for $i > i^*$ and $D_{|\tau|}$, contains at least two literals at the last decision level, otherwise it would have propagated earlier. This also applies to clauses $D_{i+1}, \ldots, D_{|\tau|}$, since they are not asserting.

We accounted for 4 literals at the last decision level present in the premises of D_i, of which 2 are not present in the conclusion because they are the pivots. In order for D_i to contain only one literal at the last decision level, the remaining two literals must be equal. ◀

3 Merge Resolution

Andrews' definition of merge resolution [1] considers tree-like proofs with the additional restriction that in every inference at least one premise is an axiom or a merge. He also observes that such derivations can be made input-structured.

▶ **Observation 4** ([1]). *A tree-like merge resolution derivation can be decomposed into an input-structured sequence where all the lemmas are merges.*

This observation is key when working with such derivations, as is apparent in Section 4, to the point that we define our proof systems in terms of the *input-structured* framework. Every resolution proof can be thought of as being input-structured if we consider it as a sequence of unit-length input resolutions and every clause as a lemma; it is when we impose restrictions on which clauses are permitted as lemmas that we obtain different proof systems.

Andrews' main result is that the merge restriction does not affect tree-like resolution.

[1] The original result does not prove 1-consistency, but the proof is analogous.

▶ **Lemma 5** ([1, Lemma 5]). *If there is a tree-like resolution derivation of C of length L where at most the root is a merge, then there is an input resolution derivation of some $C' \subseteq C$ of length at most L.*

▶ **Theorem 6** ([1, Theorem 1]). *If there is a tree-like resolution derivation of C of length L, then there is a tree-like merge resolution derivation of some $C' \subseteq C$ of length at most L.*

If we lift the tree-like restriction from the input-structured view of merge resolution proofs we obtain a proof system between tree- and DAG-like resolution where clauses can be reused (i.e., have outdegree larger than 1) if and only if they are merges or, in other words, lemmas in the input-structured decomposition. As a consequence of Lemma 3 this proof system already includes CDCL refutations produced by solvers that use the 1UIP learning scheme. In order to model all asserting learning schemes we allow reusing clauses that contain a merge not only when they are inferred but anywhere in their derivation.

▶ **Definition 7.** *A merge resolution derivation is an input-structured sequence of input resolution derivations where all derivations but the last contain a merge.*

It follows from Lemmas 1 and 2 that refutations produced by solvers that use any asserting learning scheme are in merge resolution form.

We immediately have from the simulation of resolution by CDCL [23, 2] that merge resolution polynomially simulates standard resolution. In Section 4 we make this simulation more precise and prove that the simulation overhead can be made linear, and in Section 5 that the simulation is optimal because there exist formulas that have resolution refutations of linear length but require merge resolution refutations of quadratic length.

4 Simulation

As an auxiliary tool to simulate resolution in merge resolution we define the input-resolution closure of a set G, denoted $\mathrm{Cl_i}(G) = \{C \mid \exists C' \subseteq C, G \vdash_i C'\}$, as the set of clauses derivable from G via input resolution plus weakening. It is well-known that, since input resolution derivations can be assumed to be regular—using each variable at a pivot at most once—without loss of generality, we can also assume them to have length at most linear in the number of variables.

▶ **Observation 8.** *If G is a CNF formula over n variables and $C \in \mathrm{Cl_i}(G)$ then there is a regular input resolution derivation of some $C' \subseteq C$ from G of length at most n.*

Combining Theorem 6 with the idea that in order to simulate a resolution derivation we do not need to generate each clause, but only do enough work so that in the following steps we can pretend that we had derived it [23, 2], we can prove that merge resolution simulates resolution with at most a multiplicative linear overhead in the number of variables.

▶ **Theorem 9.** *If F is a CNF formula over n variables that has a resolution refutation of length L then it has a merge resolution refutation of length $\mathrm{O}(nL)$.*

Proof. Let $\pi = (C_1, \ldots, C_L)$ be a resolution refutation. We construct a sequence of sets $F = G_0 \subseteq \cdots \subseteq G_L$ with the following properties.

1. $G_t \setminus F$ is the set of lemmas in a merge resolution derivation from F of length at most $(2n+1)t$.
2. $\pi[1,t] \subseteq \mathrm{Cl_i}(G_t)$.

This is enough to prove the theorem: since $\bot \in \mathrm{Cl_i}(G_L)$ we can obtain $\bot$ from G_L in length n, so the total length of the refutation is $(2n+1)L+n$.

We build the sets by induction, starting with $G_0 = F$. Assume we have built G_t and let $C = C_{t+1}$. If $C \in \mathrm{Cl_i}(G_t)$ we set $G_{t+1} = G_t$ and we are done. Otherwise $C = \mathrm{Res}(A, B)$ with $A, B \in \pi[1, t]$, and by induction we have $A, B \in \mathrm{Cl_i}(G_t)$, therefore by Observation 8 there are input resolution derivations of $A' \subseteq A$ and $B' \subseteq B$ of length at most n. Since neither $A' \vDash C$ nor $B' \vDash C$, A' and B' can be resolved and therefore there is a tree-like derivation η of $C' \subseteq C$ from G_t of length at most $2n+1$. By Theorem 6 there is a tree-like merge resolution derivation η' of $C'' \subseteq C$ from G_t of length at most $2n+1$. By Observation 4 the derivation η' can be decomposed into a sequence of input derivations of total length at most $2n+1$. Let E be the lemmas in that sequence and set $G_{t+1} = G_t \cup E$. We have that $C \in \mathrm{Cl_i}(F \cup E) \subseteq \mathrm{Cl_i}(G_{t+1})$, and that we can obtain E from G_t in at most $2n+1$ steps. Thus G_{t+1} has all the required properties. ◀

5 Separation

We prove the following separation between standard and merge resolution.

▶ **Theorem 10.** *There exists a family of formulas F_n over $\mathrm{O}(n \log n)$ variables and $\mathrm{O}(n \log n)$ clauses that have resolution refutations of length $\mathrm{O}(n \log n)$ but every merge resolution refutation requires length $\Omega(n^2 \log n)$.*

5.1 Formula

Let ℓ, m, n be positive integers. We have variables x_i for $i \in [m\ell - 1]$ and $w_{j,k}$ for $j \in [\ell]$ and $k \in [n]$. For convenience we define $x_0 = 1$ and $x_{m\ell} = 0$, which are not variables. Let $X = \{x_i \mid i \in [m\ell - 1]\}$, $W_j = \{w_{j,k} \mid k \in [n]\}$ and $W = \bigcup_{j \in [\ell]} W_j$. For each $j \in [\ell]$ we build the following gadget:

$$w_{j,k} = w_{j,k+1} \qquad \text{for } k \in [n-1] \tag{1}$$

Each equality is expanded into the two clauses $B_{j,k,1} = w_{j,k} \vee \overline{w_{j,k+1}}$ and $B_{j,k,0} = \overline{w_{j,k}} \vee w_{j,k+1}$, and we collectively call them $\mathcal{W} = \{B_{j,k,b} \mid j \in [\ell], k \in [n-1], b \in \{0,1\}\}$. Observe that the j-th gadget implies $w_{j,1} = w_{j,n}$. Additionally we build the following gadget:

$$(w_{1,1} = w_{1,n}) \to x_1 \tag{2}$$
$$(w_{\hat{\imath},1} = w_{\hat{\imath},n}) \to (x_{i-1} \to x_i) \qquad \text{for } i \in [2, m\ell - 1] \tag{3}$$
$$(w_{\ell,1} = w_{\ell,n}) \to \overline{x_{m\ell-1}} \tag{4}$$

where $\hat{\imath} \in [\ell]$ denotes the canonical form of $i \pmod{\ell}$. Each constraint is expanded into the two clauses $A_{i,1} = w_{\hat{\imath},1} \vee w_{\hat{\imath},n} \vee \overline{x_{i-1}} \vee x_i$ and $A_{i,0} = \overline{w_{\hat{\imath},1}} \vee \overline{w_{\hat{\imath},n}} \vee \overline{x_{i-1}} \vee x_i$, and we collectively call them $\mathcal{X} = \{A_{i,b} \mid i \in [m\ell], b \in \{0,1\}\}$. The formula $\mathcal{X} \cup \mathcal{W}$ is called $F_{\ell,m,n}$.

5.2 Upper Bound

It is not hard to see that there is a resolution refutation of $F_{\ell,m,n}$ of length $\mathrm{O}(\ell \cdot (m+n))$. Indeed, we first derive the two clauses representing $w_{j,1} = w_{j,n}$ for each $j \in [\ell]$, which requires

$\mathrm{O}(n\ell)$ steps:

$$\dfrac{\dfrac{\dfrac{\dfrac{w_{j,1} \lor \overline{w_{j,2}} \qquad w_{j,2} \lor \overline{w_{j,3}}}{w_{j,1} \lor \overline{w_{j,3}}}}{\vdots}}{w_{j,1} \lor \overline{w_{j,n-1}}} \qquad w_{j,n-1} \lor \overline{w_{j,n}}}{w_{j,1} \lor \overline{w_{j,n}}} \tag{5}$$

Then we resolve each of the $\mathcal{X}$ axioms with one of these clauses, appropriately chosen so that we obtain pairs of clauses of the form $w_i^b \lor \overline{x_{i-1}} \lor x_i$ for $i \in [m\ell]$, and resolve each pair to obtain the chain of implications $x_1, \ldots, x_i \to x_{i+1}, \ldots, \overline{x_{n\ell-1}}$ in $\mathrm{O}(m\ell)$ steps.

$$\dfrac{\dfrac{w_{\hat{\imath},1} \lor \overline{w_{\hat{\imath},n}} \qquad w_{\hat{\imath},1} \lor w_{\hat{\imath},n} \lor \overline{x_{i-1}} \lor x_i}{w_{\hat{\imath},1} \lor \overline{x_{i-1}} \lor x_i} \qquad \dfrac{\overline{w_{\hat{\imath},1}} \lor w_{\hat{\imath},n} \qquad \overline{w_{\hat{\imath},1}} \lor \overline{w_{\hat{\imath},n}} \lor \overline{x_{i-1}} \lor x_i}{\overline{w_{\hat{\imath},1}} \lor \overline{x_{i-1}} \lor x_i}}{\overline{x_{i-1}} \lor x_i} \tag{6}$$

Since we have derived a chain of implications x_1, $x_1 \to x_2$, $\ldots$, $x_{m\ell-1} \to x_{m\ell-1}$, $\overline{x_{m\ell-1}}$ we can complete the refutation in $\mathrm{O}(m\ell)$ more steps. Let us record our discussion.

▶ **Lemma 11.** *$F_{\ell,m,n}$ has a resolution refutation of length $\mathrm{O}(\ell \cdot (m+n))$.*

Before we prove the lower bound let us discuss informally what are the natural ways to refute this formula in merge resolution, so that we understand which behaviours we need to rule out.

If we try to reproduce the previous resolution refutation, since we cannot reuse the clauses representing $w_{j,1} = w_{j,n}$ because they are not merges, we have to rederive them each time we need them, which means that it takes $\mathrm{O}(mn\ell)$ steps to derive the chain of implications $x_1, \ldots, x_i \to x_{i+1}, \ldots, \overline{x_{n\ell-1}}$. We call this approach *refutation 1*. This refutation has merges (over $w_{\hat{\imath},1}$, x_{i-1}, and x_i) when we produce $w_{\hat{\imath},1}^b \lor \overline{x_{i-1}} \lor x_i$, and (over x_{i-1} and x_i) when we produce $\overline{x_{i-1}} \lor x_i$, but since we never reuse these clauses the refutation is in fact tree-like.

An alternative approach, which we call *refutation 2*, is to start working with the $\mathcal{X}$ axioms instead. In this proof we clump together all of the repeated constraints of the form $w_{j,1} \neq w_{j,n}$ for every $j \in [\ell]$, and then resolve them out in one go. In other words, we first derive some clausal encoding of the sequence of constraints

$$D_i = \Bigg(\bigvee_{\hat{\imath} \in [\min(i,\ell)]} w_{\hat{\imath},1} \neq w_{\hat{\imath},n} \Bigg) \lor x_i \qquad \text{for } i \in [m\ell] \ , \tag{7}$$

where D_i can be obtained from D_{i-1} and the pair of $\mathcal{X}$ axioms $A_{i,b}$, then resolve away the inequalities from $D_{m\ell} = \bigvee_{j \in [\ell]} w_{j,1} \neq w_{j,n}$ using the $\mathcal{W}$ axioms. However, representing any of the constraints D_i for $i \geq \ell$ requires 2^ℓ clauses, which is significantly larger than $mn\ell$ and even superpolynomial for large enough ℓ, so this refutation is not efficient either. Note that this refutation has merges (over W variables) each time that we derive D_i with $i \geq \ell$.

A third and somewhat contrived way to build a refutation is to derive the pair of clauses representing $w_{j,1} = w_{j,n}$ using a derivation whose last step is a merge, so that they can be reused. Each of these clauses can be derived individually in $\mathrm{O}(mn\ell)$ steps, for a total of $\mathrm{O}(mn\ell^2)$ steps, by slightly adapting refutation 1, substituting each derivation of $x_i \to x_{i+1}$ by a derivation of $w_{j,1} \lor \overline{w_{j,n}} \lor \overline{x_i} \lor x_{i+1}$ whenever $i \equiv j \pmod{\ell}$ so that at the end we obtain $w_{j,1} \lor \overline{w_{j,n}}$ instead of the empty clause. Such a substitution clause can be obtained, e.g., by resolving $w_{j,1} \lor w_{j,2} \lor \overline{x_i} \lor x_{i+1}$ with $\overline{w_{j,2}} \lor \overline{w_{j,n}} \lor \overline{x_i} \lor x_{i+1}$ as follows

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{w_{j,2}\lor\overline{w_{j,3}} \quad w_{j,3}\lor\overline{w_{j,4}}}{w_{j,2}\lor\overline{w_{j,4}}}}{\vdots}}{w_{j,2}\lor\overline{w_{j,n-1}}} \quad w_{j,n-1}\lor\overline{w_{j,n}}}{w_{j,2}\lor\overline{w_{j,n}}} \quad w_{i,1}\lor w_{i,n}\lor\overline{x_{i-1}}\lor x_i}{w_{i,1}\lor w_{i,2}\lor\overline{x_{i-1}}\lor x_i} \quad \dfrac{w_{i,1}\lor\overline{w_{i,2}} \quad \overline{w_{i,1}}\lor\overline{w_{i,n}}\lor\overline{x_{i-1}}\lor x_i}{\overline{w_{i,2}}\lor\overline{w_{i,n}}\lor\overline{x_{i-1}}\lor x_i}}{w_{i,1}\lor\overline{w_{i,n}}\lor\overline{x_{i-1}}\lor x_i} \tag{8}$$

After deriving $w_{j,1} = w_{j,n}$ as merges we follow the next steps of refutation 1 and complete the refutation in $\mathrm{O}(m\ell)$ steps. We call this *refutation 3*.

Observe that the minimum length of deriving the clauses representing $w_{j,1} = w_{j,n}$ is only $\mathrm{O}(n)$, even in merge resolution, so if we only used the information that refutation 3 contains these clauses we would only be able to bound its length by $\Omega(\ell \cdot (m+n))$. Therefore when we compute the hardness of deriving a clause we need to take into account not only its semantics but how it was obtained syntactically.

5.3 Lower Bound

Before we begin proving our lower bound in earnest we make two useful observations.

▶ **Lemma 12.** *Let η be a resolution derivation that only depends on $\mathcal{W}$ axioms. Then η does not contain any merges, and all clauses are supported on W.*

Proof. We prove by induction that every clause in η is of the form $w_{j,k} \lor \overline{w_{j,k'}}$ with $k \neq k'$. This is true for the axioms. By induction hypothesis, a generic resolution step over $w_{j,k}$ is of the form

$$\dfrac{w_{j,k} \lor \overline{w_{j,k'}} \qquad \overline{w_{j,k}} \lor w_{j,k''}}{w_{j,k''} \lor \overline{w_{j,k'}}} \tag{9}$$

and in particular is not a merge. ◀

▶ **Lemma 13.** *Let η be a resolution derivation of a clause C supported on W variables that uses an $\mathcal{X}$ axiom. Then η uses at least one $A_{i,b}$ axiom for each $i \in [m\ell]$.*

Proof. We prove the contrapositive and assume that there is an axiom $A_{i,b}$ that is used, and either both $A_{i+1,0}$ and $A_{i+1,1}$ are not used, or both $A_{i-1,0}$ and $A_{i-1,1}$ are not. In the first case the literal x_i appears in every clause in the path from $A_{i,b}$ to C, contradicting that C is supported on W variables. Analogously with literal $\overline{x_{i-1}}$ in the second case. ◀

At a high level, our first step towards proving the lower bound is to rule out that refutations like refutation 2 can be small, and to do that we show that wide clauses allow for very little progress. This is a common theme in proof complexity, and the standard tool is to apply a random restriction to a short refutation in order to obtain a narrow refutation. However, merge resolution is not closed under restrictions, as we prove later in Corollary 24, and because of this we need to argue separately about which merges are preserved.

We observed that derivation fragments where no X variable appears do not contain any merges, but we cannot claim that clauses where no X variable appears are not merges. However, refutation 3 suggests that deriving a clause supported on W variables that depends on $\mathcal{X}$ axioms should still be hard, so we restrict our attention to the part of the refutation before any such clause is derived.

We then show that the resulting refutation needs to use all X variables, and picks up a pair of W variables each time that a new X variable appears. If we were to introduce the X variables in order, and since we ruled out wide clauses, every time that we use ℓ many X variables we also need to eliminate proportionately as many W variables before we move onto the next interval of X variables. We would expect to eliminate variables $\Omega(m\ell)$ times, and each elimination requires $\Omega(n)$ steps. Of course the refutation might not use X variables in order, but we can still break the proof into parts, each corresponding roughly to an interval of ℓ many X variables.

Let us begin implementing our plan by defining the class of restrictions that we use and which need to respect the structure of the formula. A restriction is an autarky [19] with respect to a set of clauses G if it satisfies every clause that it touches; in other words for every clause $C \in G$ either $C\restriction_\rho = 1$ or $C\restriction_\rho = C$. A restriction is k-respecting if it is an autarky with respect to $\mathcal{W}$ axioms, we have $F_{\ell,m,n}\restriction_\rho \cong F_{k,m,n}$ up to variable renaming, and every X variable is mapped to an X variable. Our definition of a narrow clause is also tailored to the formula at hand, and counts the number of different W-blocks that a clause C mentions. Formally $\mu(C) = |\{j \in [\ell] \mid \exists x_{j,k} \in \text{vars(C)}\}|$.

▶ **Lemma 14.** *Let π be a resolution refutation of $F_{\ell,m,n}$ of length $L = \mathrm{o}((4/3)^{\ell/8})$. There exists an $\ell/4$-respecting restriction ρ such that every clause in $\pi\restriction_\rho$ has $\mu(C) \le \ell/8$.*

Proof. We use the probabilistic method. Consider the following distribution $\mathcal{J}$ over $\{0,1,*\}^\ell$: each coordinate is chosen independently with $\Pr[J_i = 0] = \Pr[J_i = 1] = 1/4$, $\Pr[J_i = *] = 1/2$. Given a random variable $J \sim \mathcal{J}$ sampled according to this distribution, we derive a random restriction ρ as follows: $\rho(w_{j,i}) = J_j$, $\rho(x_i) = *$ if $J_{\hat{\imath}} = *$, and $\rho(x_i) = \rho(x_{i-1})$ otherwise (where $\rho(x_0) = 1$).

Observe that $F_{\ell,m,n}\restriction_\rho \cong F_{|J^{-1}(*)|,m,n}$ up to variable renaming, and by a Chernoff bound we have $\Pr[|J^{-1}(*)| < \ell/4] \le e^{-\ell/16}$.

We also have, for every clause $C \in \pi$ with $\mu(C) > \ell/8$, that

$$\Pr[C\restriction_\rho \neq 1] \le (3/4)^{\mu(C)} \le (3/4)^{\ell/8} \quad . \tag{10}$$

Therefore by a union bound the probability that $|J^{-1}(*)| < \ell/4$ or that any clause has $\mu(C\restriction_\rho) > \ell/8$ is bounded away from 1 and we conclude that there exists a restriction ρ that satisfies the conclusion of the lemma. ◀

Note that $s(\pi\restriction_\rho)$ is a resolution refutation of $F_{n,\ell,m}\restriction_\rho$, but not necessarily a merge resolution refutation, therefore we lose control over which clauses may be reused[2]. Nevertheless, we can identify a fragment of $s(\pi\restriction_\rho)$ where we still have enough information.

▶ **Lemma 15.** *Let π be a merge resolution refutation of $F_{n,\ell,m}$ and ρ be the restriction from Lemma 14. There exists an integer t such that $\psi = s(\pi[1,t]\restriction_\rho)$ is a resolution derivation of a clause supported on W variables that depends on an $\mathcal{X}$ axiom and where no clause supported on W variables is reused.*

[2] Recall that $s(\pi)$ is the syntactic equivalent of π.

Proof. Let $C_t \in \pi$ be the first clause that depends on an $\mathcal{X}$ axiom and such that $D_t = s(C_t\restriction_\rho)$ is supported on W, which exists because $\bot$ is one such clause.

By definition of t, we have that every ancestor $D_k \in \psi$ of D_t that is supported on W variables corresponds to a clause C_k in π that only depends on $\mathcal{W}$ axioms, hence by Lemma 12 C_k is not a merge. By definition of merge resolution C_k is not reused, and by construction of $s(\cdot)$ neither is D_k.

It remains to prove that D_t depends on an $\mathcal{X}$ axiom. Since C_t depends on an $\mathcal{X}$ axiom, at least one of its predecessors C_p and C_q also does, say C_p. By definition of t, $D_p = s(C_p\restriction_\rho)$ is not supported on W, and hence by Lemma 12 either D_p depends on an $\mathcal{X}$ axiom or $D_p = 1$. Analogously, if C_q also depends on an $\mathcal{X}$ axiom then so does $D_q = s(C_j\restriction_\rho)$ (or it is 1) and we are done. Otherwise C_q is of the form $w_{j,k} \lor \overline{w_{j,k'}}$ and is either satisfied by ρ or left untouched. In both cases we have that $D_q \not\models C_t\restriction_\rho$ (trivially in the first case and because D_q contains the pivot while C_t does not in the second), hence D_t depends on D_p. ◀

Note that C_t may be semantically implied by the $\mathcal{W}$ axioms, and have a short derivation as in refutation 3, therefore we are forced to use syntactic arguments to argue that deriving C_t *using an $\mathcal{X}$ axiom* takes many resolution steps.

The next step is to break ψ into m (possibly intersecting) parts, each corresponding roughly to the part of ψ that uses $\mathcal{X}$ axioms with variables in an interval of length ℓ (by Lemma 13 we can assume that ψ contains axioms from every interval). To do this we use the following family of restrictions defined for $i \in [n]$:

$$\sigma_i(x_{i'}) = \begin{cases} 1 & \text{if } i' \le i\ell \\ * & \text{if } i\ell < i' \le (i+1)\ell \\ 0 & \text{if } (i+1)\ell < i' \end{cases} \qquad\qquad \sigma_i(w_{i',j}) = * \tag{11}$$

Let $X_i = X \cap \sigma_i^{-1}(*)$ and note that $F_{\ell,m,n}\restriction_{\sigma_i} \cong F_{\ell,1,n}$.

Clauses in ψ with many X variables could be tricky to classify, but intuitively it should be enough to look at the smallest positive literal and the largest negative literal, since these are the hardest to eliminate. Therefore we define $r(C)$ to be the following operation on a clause: literals over W variables are left untouched, all positive X literals but the smallest are removed, and all negative X literals but the largest are removed. Formally,

$$r\Bigg(\bigvee_{i\in A} x_i \lor \bigvee_{i\in B} \overline{x_i} \lor \bigvee_{(i,j)\in C} w_{i,j}^{b_{i,j}}\Bigg) = x_{\min A} \lor \overline{x_{\max B}} \lor \bigvee_{(i,j)\in C} w_{i,j}^{b_{i,j}} \tag{12}$$

where $x_{\min A}$ (resp. $\overline{x_{\max B}}$) is omitted if A (resp. B) is empty.

We need the following property of $r(C)$.

▶ **Lemma 16.** *If $C\restriction_{\sigma_i} \neq 1$ and* $\mathrm{vars(r(C))} \cap X_i = \emptyset$ *then $C\restriction_{\sigma_i}$ is supported over W variables.*

Proof. The hypothesis that $\mathrm{vars(r(C))} \cap X_i = \emptyset$ implies that the smallest positive X literal in C is either not larger than $i\ell$ or larger than $(i+1)\ell$, but the hypothesis that $C\restriction_{\sigma_i} \neq 1$ rules out the first case. Therefore all positive X literals are falsified by σ_i. Analogously the largest negative X literal is not larger than $i\ell$ and all negative X literals are also falsified. ◀

Now we are ready to formally define how to divide ψ.

▶ **Definition 17.** *The i-th part of ψ is the sequence ψ_i of all clauses $C \in \psi$ such that C is either*

1. *an $\mathcal{X}$ axiom not satisfied by σ_i; or*

2. *the conclusion of an inference with pivot in X_i; or*
3. *the conclusion of an inference with pivot in W that depends on an $\mathcal{X}$ axiom if $r(C)$ contains a variable in X_i; or*
4. *a clause that does not depend on $\mathcal{X}$ axioms if the* only *immediate successor of C is in ψ_i.*

For convenience we use the same indexing for ψ_i and ψ, so elements of ψ_i are not necessarily consecutive. Note that ψ_i needs not be a valid derivation.

This is the point in the proof where we use crucially that the original derivation is in merge resolution form: because clauses that do not depend on $\mathcal{X}$ axioms are not merges, they have only one successor and the definition is well-formed.

As an example, if ψ were refutation 1, whose only clause supported over W variables that depends on $\mathcal{X}$ axioms is the final empty clause and therefore satisfies the guarantees given by the conclusion of Lemma 15, then we would divide it as follows. Axioms $A_{1,b}, \ldots, A_{\ell,b}$ are all part of ψ_1 because of Item 1. The result of resolving axioms $A_{i,b}$ with the clauses representing $w_{i,1} = w_{i,n}$ in order to obtain the implications $x_i \to x_{i+1}$ for $i \in [1, \ell]$ are also part of ψ_1 because of Item 3. This implies that the $\mathcal{W}$ axioms and intermediate clauses used to derive $w_{i,1} = w_{i,n}$ are also part of ψ_1 because of Item 4. Derivations of x_{i+1} from x_i and $x_i \to x_{i+1}$ for $i \in [1, \ell]$ are part of ψ_1 because of Item 2. Similarly ψ_2 will contain the analogous clauses with $i \in [\ell + 1, 2\ell]$ and so on. Note that each part ψ_i contains derivations of $w_{j,1} = w_{j,n}$ for $j \in [\ell]$, but they refer to different copies of the same derivation.

Ideally we would like to argue that parts ψ_i are pairwise disjoint and of size $\Omega(n\ell)$, which would allow us to bound $|\psi| = \sum_{i\in[m]} |\psi_i| = \Omega(mn\ell)$. This is not quite true, but nevertheless clauses do not appear in too many different parts and we have the following bound.

▶ **Lemma 18.** *Let ψ and $\{\psi_i \mid i \in [m]\}$ be as discussed above. Then $2|\psi| \geq \sum_{i\in[m]} |\psi_i|$.*

Proof. Axioms may appear in at most two different ψ_i, and clauses obtained after resolving with an X pivot in only one. The only other clauses that depend on an $\mathcal{X}$ axiom and may appear in different ψ_i are obtained after resolving with a W pivot, but since $r(C)$ only contains two X variables, such clause only may appear in two different ψ_i. Finally, clauses that do not depend on an $\mathcal{X}$ axiom appear in the same ψ_i as one clause of the previous types, and therefore at most two different parts. ◀

To conclude the proof we need to argue that each ψ_i has size $\Omega(n\ell)$. The intuitive reason is that ψ_i must use one $\mathcal{X}$ axiom for each $j \in [(i\ell, (i+1)\ell]$, which introduces a pair of W variables from each W_j block, but since no clause contains more than $\ell/8$ such variables, we need to use enough $\mathcal{W}$ axioms to remove the aforementioned W variables. Formally we first need to extract a valid derivation from ψ_i as we do in the next lemma.

▶ **Definition 19.** *For each $i \in [m]$ let t_i be the smallest integer such that clause C_{t_i} depends on an $\mathcal{X}$ axiom (with respect to ψ), $C_{t_i}\restriction_{\sigma_i}$ is supported on W variables, and $C_{t_i} \in \psi_i$. If no such clause exists then we set $t_i = \infty$.*

▶ **Lemma 20.** *For each $i \in [m]$, t_i is finite and $s(\psi_i[1, t_i]\restriction_{\sigma_i})$ is a valid resolution derivation.*

Proof. We prove by induction that for all $k \leq \min(t_i, t)$, if the clause $C_k \in \psi$ depends on an $\mathcal{X}$ axiom and is not satisfied by σ_i, then there exists a clause $C_{k'} \in \psi_i$ with $k' \leq k$ that implies C_k modulo σ_i, that is $C_{k'}\restriction_{\sigma_i} \vDash C_k\restriction_{\sigma_i}$, and depends on an $\mathcal{X}$ axiom (over ψ).

When C_k is a non-satisfied $\mathcal{X}$ axiom we can simply take $C_{k'} = C_k$ by Item 1 of Definition 17. Otherwise let C_p and C_q be the premises of C_k in ψ and we consider a few cases.

Case 1: the pivot is an X variable. Then both premises depend on an $\mathcal{X}$ axiom (by Lemma 12).

Case 1.1: the pivot is an X_i variable. Then we can take $C_{k'} = C_k$ by Item 2 of Definition 17.

Case 1.2: the pivot is an $X_{i'}$ variable with $i' \neq i$. Then the pivot is assigned by σ_i and exactly one of the premises, say C_p, is non-satisfied. By the induction hypothesis we can take $C_{k'} = C_{p'}$.

Case 2: the pivot is a W variable. If either premise were satisfied, and since σ_i only assigns X variables, the satisfied literal would carry over to C_k, hence neither premise is satisfied.

Case 2.1: exactly one premise depends on an $\mathcal{X}$ axiom, say C_p. Then $C_{p'}$ is present in ψ_i, and by Item 4 of Definition 17 the other premise C_q is present in ψ_i if and only if the conclusion C_k is.

Case 2.2: both premises depend on an $\mathcal{X}$ axiom. Then both $C_{p'}$ and $C_{q'}$ are present in ψ_i.

Therefore in both subcases it is enough to prove that $C_k \in \psi_i$, since then we can take $C_{k'} = C_k$ and we have that $C_k\restriction_{\sigma_i}$ follows from a valid semantic resolution step. Assume for the sake of contradiction that $C_k \notin \psi_i$. Then for Item 3 of Definition 17 not to apply it must be that $r(C_k)$ does not contain any variable from X_i. By Lemma 16 $C_k\restriction_{\sigma_i}$ is a clause supported on W variables, which by definition of C_{t_i} implies that $k = t_i$. However, since the pivot is a W variable, $C_{p'}\restriction_{\sigma_i}$ is also supported on W variables and, together with the fact that $C_{p'}$ depends on an $\mathcal{X}$ axiom, this contradicts that C_{t_i} is the first such clause.

This finishes the induction argument and proves that $\psi_i[1, t_i]\restriction_{\sigma_i}$ is a valid semantic derivation, from where it follows that $s(\psi_i[1, t_i]\restriction_{\sigma_i})$ is a valid syntactic derivation. It also follows from the induction hypothesis that $t_i \leq t$: since C_t is left untouched by σ_i and depends on an $\mathcal{X}$ axiom, there exists a clause $C_{t'} \in \psi_i$ that depends on an $\mathcal{X}$ axiom and such that $C_{t'}\restriction_{\sigma_i} \vDash C_t\restriction_{\sigma_i} = C_t$, which is supported on W variables. ◀

Having established that each ψ_i is a valid derivation, we show that they are large in the following two lemmas.

▶ **Lemma 21.** *For each $i \in [m]$ the clause $C_{t_i}\restriction_{\sigma_i}$ depends on an $\mathcal{X}$ axiom with respect to derivation $s(\psi_i[1, t_i]\restriction_{\sigma_i})$.*

Proof. We prove by induction that for every clause $D_k \in s(\psi_i[1, t_i]\restriction_{\sigma_i})$, if C_k depends on an $\mathcal{X}$ axiom (over ψ) then so does D_k (over $s(\psi_i[1, t_i]\restriction_{\sigma_i})$). This is immediate when D_k is an axiom.

Otherwise fix C_k, $E_k = C_k\restriction_{\sigma_i}$, and $D_k = s(E_k)$, and let $E_p = C_p\restriction_{\sigma_i}$ and $E_q = C_q\restriction_{\sigma_i}$ be the premises of E_k in the semantic derivation $\psi_i[1, t_i]\restriction_{\sigma_i}$. When both C_p and C_q depend on an X axiom, then by hypothesis so do D_p and D_q and we are done because at least one of them is used to syntactically derive D_k. Otherwise one premise, say C_p, depends on an X axiom and the other premise, say C_q, does not. In that case, because σ_i only affects X variables, all the axioms used in the derivation of C_q are left untouched by σ_i, therefore we have that $D_q = E_q = C_q$, which contains the pivot used to derive C_k and therefore E_q alone does not imply E_k. In other words, the other premise E_p is semantically needed to derive E_k, and thus $D_p = s(E_p)$ is syntactically used to derive D_k. ◀

▶ **Lemma 22.** *Let η be a resolution derivation from $F_{\ell,1,n}$ of a clause C supported on W variables that depends on an $\mathcal{X}$ axiom. Then $|\eta| \geq (n-2)(\ell - \mu(C))/2$.*

Proof. By Lemma 13 we can assume that η uses at least one $A_{j,b}$ axiom for each $j \in [\ell]$.

Let $J = \{j \in [\ell] \mid \exists w_{j,k} \in \text{vars(C)}\}$ be the set of W blocks mentioned by C. We show that for each $j \in \overline{J} = [\ell] \setminus J$ at least $(n-2)/2$ axioms over variables in W_j appear in η, which makes for at least $(n-2)|\overline{J}|/2 = (n-2)(\ell - \mu(C))/2$ axioms.

Fix $j \in \overline{J}$ and assume for the sake of contradiction that less than $(n-2)/2$ axioms over variables in W_j appear in η. Then there exists $k \in [2, n-1]$ such that variable $w_{j,k}$ does not appear in η. Rename variables as follows: $w_{j,k'} \mapsto y_{k'}$ for $k' < k$, and $w_{j,k'} \mapsto \overline{y_{k'-n}}$ for $k' > k$. Then we can prove by induction, analogously to the proof of Lemma 12, that every clause derived from axiom $A_{j,b}$ is of the form $y_{k'} \vee \overline{y_{k''}} \vee D$ where D are literals supported outside W_j. Since that includes C, it contradicts our assumption that $j \notin J$. ◀

To conclude the proof of Theorem 10 we simply need to put the pieces together.

Proof of Theorem 10. We take as the formula family $F_{\ell=48 \log n, n, n}$, for which a resolution refutation of length $\mathrm{O}(n \log n)$ exists by Lemma 11.

To prove a lower bound we assume that a merge resolution refutation π of length $L \leq n^3 = 2^{16\ell} = \mathrm{o}((4/3)^{8\ell})$ exists; otherwise the lower bound trivially holds. We apply the restriction given by Lemma 14 to π and we use Lemma 15 to obtain a resolution derivation ψ of a clause supported on W variables that uses an $\mathcal{X}$ axiom. We then break ψ into m parts ψ_i, each of size at least $n\ell/16$ as follows from Lemmas 20, 21, and 22. Finally by Lemma 18 we have $|\pi| \geq |\psi| \geq mn\ell/32 = \Omega(n^2 \log n)$. ◀

5.4 Structural Consequences

Theorem 10 immediately gives us two unusual structural properties of merge resolution. One is that proof length may decrease when introducing a weakening rule.

▶ **Corollary 23.** *There exists a family of formulas over $\mathrm{O}(n \log n)$ variables and $\mathrm{O}(n \log n)$ clauses that have merge resolution with weakening refutations of length $\mathrm{O}(n \log n)$ but every merge resolution refutation requires length $\Omega(n^2 \log n)$.*

Proof. Consider the formula $F_n \wedge \overline{z}$, where F_n is the formula given by Theorem 10 and z is a new variable. If we weaken every clause $C \in F_n$ to $C \vee z$ then we can derive $F \vee z \vdash z$ in $\mathrm{O}(n \log n)$ merge resolution steps because each inference is a merge. However, if we cannot do weakening, then $\overline{z}$ cannot be resolved with any clause in F_n and the lower bound of Theorem 10 applies. ◀

The second property is that merge resolution is not a *natural* proof systems in the sense of [5] because proof length may increase after a restriction.

▶ **Corollary 24.** *There exists a restriction ρ and a family of formulas over $\mathrm{O}(n \log n)$ variables and $\mathrm{O}(n \log n)$ clauses that have merge resolution refutations of length $\mathrm{O}(n \log n)$ but every merge resolution refutation of $F_n{\restriction_\rho}$ requires length $\Omega(n^2 \log n)$.*

Proof. Consider the formula $G_n = (F_n \vee z) \wedge \overline{z}$, where F_n is the formula given by Theorem 10, $F \vee z = \{C \vee z \mid C \in F\}$, and z is a new variable. As in the proof of Corollary 23 there is a merge resolution derivation of z of length $\mathrm{O}(n \log n)$ steps, while $G_n{\restriction_\rho} = F_n$. ◀

6 Further Proof Systems

In order to model CDCL with 1UIP more closely and possibly obtain a stronger separation we look at restricted versions of merge resolution, which in this section we refer to as Resolution with Merge Ancestors (RMA) to disambiguate it from the rest. The diagram in Figure 1 can help keeping track of these.

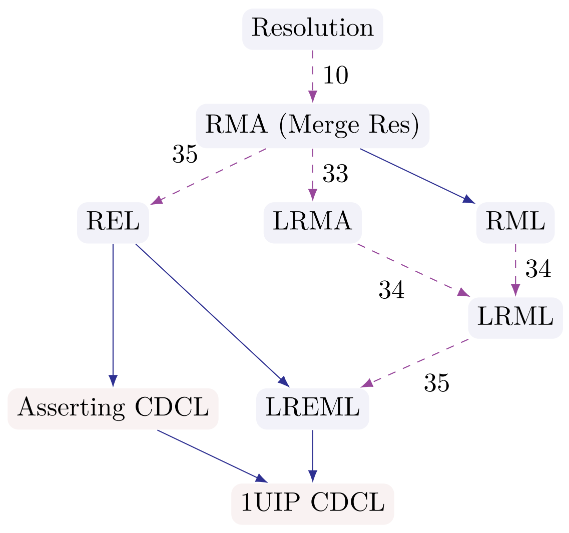

Figure 1 Relations between proof systems. A solid arrow $A \longrightarrow B$ indicates that A simulates B with no overhead. A dashed arrow $A \dashrightarrow B$ indicates that A simulates B with no overhead, but B requires linear overhead to simulate A. Statements proving separations are referenced.

▶ **Definition 25.** *A Resolution with Merge Ancestors (RMA) derivation is an input-structured sequence of input resolution derivations where all derivations but the last contain a merge.*

Note that by Lemma 5 it does not matter if we require the sequence of derivations of an RMA derivation to be input derivations or if we allow general trees. In fact, our lower bound results hold for a more general proof system where we only ask that every clause with outdegree larger than 1 has an ancestor that is a merge. Such proof system does not have a simple input structure, but can rather be thought of as a sequence of tree-like resolution derivations whose roots are merges, followed by a standard resolution derivation using the roots of the previous derivations as axioms.

To make the connection back to CDCL, we can define a proof system called Resolution with Empowering Lemmas that captures CDCL refutations produced by solvers that use any asserting learning scheme or 1-empowering learning scheme.

▶ **Definition 26.** *Let $C_1, \ldots, C_{L-1}$ be the lemmas of an input-structured sequence of input derivations. The sequence is a Resolution with Empowering Lemmas (REL) derivation of a formula F if C_i is 1-empowering with respect to $F \cup \{C_j : j < i\}$ for all $i \in [1, L-1]$.*

It follows from Lemma 2 that such refutations are in RMA form.

▶ **Observation 27.** *A REL derivation is a RMA derivation.*

It might seem more natural to work with the REL proof system rather than its merge-based counterparts, since REL is defined exactly through the 1-empowering property. However, while the merge property is easy to check because it is local to the derivation at hand, we can only determine if a clause is 1-empowering by looking at the full history of the derivation, in particular what the previous lemmas are. This makes REL too cumbersome to analyse. Furthermore, refutations produced by CDCL applying a clause minimization scheme on top of an asserting clause might not be in REL form, but they are still in RMA form.

We also discussed in Section 3 we that 1UIP CDCL solvers produce derivations where lemmas themselves are merges. We call this proof system Resolution with Merge Lemmas, or RML for short.

▶ **Definition 28.** *A Resolution with Merge Lemmas (RML) derivation is an input-structured sequence of input resolution derivations where all lemmas are merges.*

We can be even more restrictive and observe that input derivations produced by a CDCL solver that we describe next is that once a variable is resolved, it does not appear later in the derivation.

▶ **Definition 29.** *A resolution derivation η is strongly regular if for every resolution step i, the pivot variable x_i is not amongst the variables of any clause $C_i \in \eta[i, L]$. A sequence of derivations is locally regular if every derivation in the sequence is strongly regular. A LRML derivation (resp. LRMA) is a locally regular RML derivation (resp. RMA).*

Finally we can consider derivations that have empowering, merge lemmas and are locally regular. These still include 1UIP proofs.

▶ **Definition 30.** *A LREML derivation is a derivation that is both LRML and REL.*

All of the proof systems we defined are quadratically separated from resolution simply because they are weaker than RMA. At the same time, all of these proof systems still simulate standard resolution up to linear overhead, as we show next.

Going back to the proof of Theorem 9, we first observe that the resulting RMA refutation is in fact an RML refutation.

Recall that the proof idea is to maintain a set of clauses G_t such that all clauses in the proof up to time t can be derived from G_t by input derivation. Then, given a new clause C_{t+1} that can be obtained from G_t with a derivation η, we transform η into a merge resolution derivation using Theorem 6, and we add its lemmas to G_{t+1}. Since Theorem 6 produces RML derivations, so is the final derivation we construct.

To make the simulation work also for LREML we need the following lemma.

▶ **Lemma 31** ([23]). *If F absorbs $A \vee x$ and $B \vee \overline{x}$, then $F \vdash_i C' \subseteq A \vee B$.*

The simulation itself follows the general structure of Theorem 9, except that we need some additional work to construct G_{t+1} from G_t.

▶ **Theorem 32.** *If F is a CNF formula over n variables that has a resolution refutation of length L then it has a LREML refutation of length $\mathrm{O}(nL)$.*

Proof. Let $\pi = (C_1, \ldots, C_L)$ be a resolution refutation. As we already showed it is enough to construct a sequence of sets $F = G_0 \subseteq \cdots \subseteq G_L$ such that $G_t \setminus F$ is the set of lemmas in a LREML derivation from F of length at most $(2n+1)t$, and $\pi[1,t] \subseteq \mathrm{Cl_i}(G_t)$. Assume we have built G_t and let $C = C_{t+1}$. If $C \in \mathrm{Cl_i}(G_t)$ we set $G_{t+1} = G_t$ and we are done. Otherwise we showed that there are input resolution derivations of $A' \subseteq A$ and $B' \subseteq B$ from G_t of length at most n, which we can assume are strongly regular, and that A' and B' can be resolved together.

At this point we deviate from the proof of Theorem 9. We inductively build an intermediate sequence of sets $G_t = G_t^0 \subseteq \ldots \subseteq G_t^k$ with the following properties.

1. G_t^j is the set of lemmas in a LREML derivation from G_t of length at most p.
2. A' and B' can be derived from G_t^j in at most $2n - p$ resolution steps.

The base case $G_t^0 = G_t$ is trivial. For the inductive case, let us first assume that either A' or B' is 1-empowering, say A'. Let E be the first 1-empowering clause in the derivation of A'. By Lemma 2 E is a merge, therefore we are allowed to take $G_t^{j+1} = G_t^j \cup \{E\}$. Furthermore, if A' and E could be derived from G_t^j in r and s steps, then A' can be derived from G_t^{j+1} in $r - s$ steps, simply by omitting the first s steps in the derivation.

Otherwise, by Lemma 31 we reached a set G_t^j such that $C \in \mathrm{Cl_i}(G_t^j)$. In this case we choose $k = j$ and $G_{t+1} = G_t^k$. Since G_{t+1} can be obtained in $p \leq 2n$ steps from G_t, it satisfies the required properties. This concludes both the inner and outer inductions. ◀

One consequence of the proof systems we introduce being polynomially equivalent to resolution is that they are conjectured to be incomparable with respect to the related RTL and pool resolution proof systems, since these are conjectured to be exponentially weaker than resolution. This would not be too unexpected given the different purposes of the proof systems: RTL and pool resolution were introduced to study restarts, and include proofs produced by CDCL without restarts but any kind of learning, while the purpose of merge-based proof systems is to study learning, and include proofs produced by CDCL with our without restarts but only asserting learning.

We can separate the different proof systems that we introduced using a few variations of $F_{\ell,m,n}$ where we add a constant number of redundant clauses for each $i \in [\ell]$. We present the results that we obtain next, and defer the proofs to the appendix.

▶ **Proposition 33.** *There exists a family of formulas over* $\mathrm{O}(n \log n)$ *variables and* $\mathrm{O}(n \log n)$ *clauses that have RMA refutations of length* $\mathrm{O}(n \log n)$ *but every LRMA refutation requires length* $\Omega(n^2 \log n)$*.*

▶ **Proposition 34.** *There exists a family of formulas over* $\mathrm{O}(n \log n)$ *variables and* $\mathrm{O}(n \log n)$ *clauses that have RML and LRMA and refutations of length* $\mathrm{O}(n \log n)$ *but every LRML refutation requires length* $\Omega(n^2 \log n)$*.*

▶ **Proposition 35.** *There exists a family of formulas over* $\mathrm{O}(n \log n)$ *variables and* $\mathrm{O}(n \log n)$ *clauses that have LRML refutations of length* $\mathrm{O}(n \log n)$ *but every REL refutation requires length* $\Omega(n^2 \log n)$*.*

7 Concluding Remarks

In this paper, we address the question of the tightness of simulation of resolution proofs by CDCL solvers. Specifically, we show that RMA, among other flavours of DAG-like merge resolution, simulates standard resolution with at most a linear multiplicative overhead. However, contrary to what we see in the tree-like case, this overhead is necessary. While the proof systems we introduce help us explain one source of overhead in the simulation of resolution by CDCL, it is not clear if they capture it exactly. In other words, an interesting future direction would be to explore whether it is possible for CDCL to simulate some flavour of merge resolution with less overhead than what is required to simulate standard resolution.

References

1 Peter B. Andrews. Resolution with merging. *J. ACM*, 15(3):367–381, 1968.

2 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, January 2011. Preliminary version in *SAT '09*.

3 Gilles Audemard, Lucas Bordeaux, Youssef Hamadi, Saïd Jabbour, and Lakhdar Sais. A generalized framework for conflict analysis. In Hans Kleine Büning and Xishun Zhao, editors, *Theory and Applications of Satisfiability Testing - SAT 2008, 11th International Conference, SAT 2008, Guangzhou, China, May 12-15, 2008. Proceedings*, volume 4996 of *Lecture Notes in Computer Science*, pages 21–27. Springer, 2008. `doi:10.1007/978-3-540-79719-7\_3`.

4 Roberto J. Bayardo Jr. and Robert Schrag. Using CSP look-back techniques to solve real-world SAT instances. In *Proceedings of the 14th National Conference on Artificial Intelligence (AAAI '97)*, pages 203–208, July 1997.

5 Paul Beame, Henry Kautz, and Ashish Sabharwal. Towards understanding and harnessing the potential of clause learning. *Journal of Artificial Intelligence Research*, 22:319–351, December 2004. Preliminary version in *IJCAI '03*.

6 Olaf Beyersdorff and Benjamin Böhm. Understanding the relative strength of QBF CDCL solvers and QBF resolution. In James R. Lee, editor, *12th Innovations in Theoretical Computer Science Conference, ITCS 2021, January 6-8, 2021, Virtual Conference*, volume 185 of *LIPIcs*, pages 12:1–12:20. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021. `doi:10.4230/LIPIcs.ITCS.2021.12`.

7 Avrim L Blum and Merrick L Furst. Fast planning through planning graph analysis. *Artificial intelligence*, 90(1-2):281–300, 1997.

8 Sam Buss and Jakob Nordström. Proof complexity and SAT solving. In Armin Biere, Marijn Heule, Hans van Maaren, and Toby Walsh, editors, *Handbook of Satisfiability*, volume 336 of *Frontiers in Artificial Intelligence and Applications*, chapter 7, pages 233–350. IOS Press, 2nd edition, 2021. `doi:10.3233/FAIA200990`.

9 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(4:13), December 2008.

10 Cristian Cadar, Vijay Ganesh, Peter M Pawlowski, David L Dill, and Dawson R Engler. EXE: Automatically Generating Inputs of Death. *ACM Transactions on Information and System Security (TISSEC)*, 12(2):1–38, 2008.

11 Alvaro del Val. Tractable databases: How to make propositional unit resolution complete through compilation. In Jon Doyle, Erik Sandewall, and Pietro Torasso, editors, *Proceedings of the 4th International Conference on Principles of Knowledge Representation and Reasoning (KR'94). Bonn, Germany, May 24-27, 1994*, pages 551–561. Morgan Kaufmann, 1994.

12 Nachum Dershowitz, Ziyad Hanna, and Alexander Nadel. Towards a better understanding of the functionality of a conflict-driven SAT solver. In João Marques-Silva and Karem A. Sakallah, editors, *Theory and Applications of Satisfiability Testing - SAT 2007, 10th International Conference, Lisbon, Portugal, May 28-31, 2007, Proceedings*, volume 4501 of *Lecture Notes in Computer Science*, pages 287–293. Springer, 2007. `doi:10.1007/978-3-540-72788-0\_27`.

13 Julian Dolby, Mandana Vaziri, and Frank Tip. Finding Bugs Efficiently With a SAT Solver. In *Proceedings of the 6th joint meeting of the European Software Engineering Conference and the ACM SIGSOFT International Symposium on Foundations of Software Engineering*, pages 195–204, 2007. `doi:10.1145/1287624.1287653`.

14 Nick Feng and Fahiem Bacchus. Clause size reduction with all-uip learning. In Luca Pulina and Martina Seidl, editors, *Theory and Applications of Satisfiability Testing - SAT 2020 - 23rd International Conference, Alghero, Italy, July 3-10, 2020, Proceedings*, volume 12178 of *Lecture Notes in Computer Science*, pages 28–45. Springer, 2020. `doi:10.1007/978-3-030-51825-7\_3`.

15 Philipp Hertel, Fahiem Bacchus, Toniann Pitassi, and Allen Van Gelder. Clause learning can effectively P-simulate general propositional resolution. In *Proceedings of the 23rd National Conference on Artificial Intelligence (AAAI '08)*, pages 283–290, July 2008.

16 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: 23rd International Conference, Alghero, Italy, July 3–10, 2020, Proceedings 23, pages 233–249. Springer, 2020.

17 João Marques-Silva, Inês Lynce, and Sharad Malik. Conflict-driven clause learning SAT solvers. In Armin Biere, Marijn Heule, Hans van Maaren, and Toby Walsh, editors, *Handbook of Satisfiability - Second Edition*, volume 336 of *Frontiers in Artificial Intelligence and Applications*, pages 133–182. IOS Press, 2021. `doi:10.3233/FAIA200987`.

18 João P. Marques-Silva and Karem A. Sakallah. GRASP: A search algorithm for propositional satisfiability. *IEEE Transactions on Computers*, 48(5):506–521, May 1999. Preliminary version in *ICCAD '96*.

19 Burkhard Monien and Ewald Speckenmeyer. Solving satisfiability in less than 2^n steps. *Discret. Appl. Math.*, 10(3):287–295, 1985. `doi:10.1016/0166-218X(85)90050-2`.

20 Matthew W. Moskewicz, Conor F. Madigan, Ying Zhao, Lintao Zhang, and Sharad Malik. Chaff: Engineering an efficient SAT solver. In *Proceedings of the 38th Design Automation Conference (DAC '01)*, pages 530–535, June 2001.

21 Nathan Mull, Shuo Pang, and Alexander A. Razborov. On CDCL-based proof systems with the ordered decision strategy. In *Proceedings of the 23rd International Conference on Theory and Applications of Satisfiability Testing (SAT '20)*, volume 12178 of *Lecture Notes in Computer Science*, pages 149–165. Springer, July 2020.

22 Knot Pipatsrisawat and Adnan Darwiche. A new clause learning scheme for efficient unsatisfiability proofs. In Dieter Fox and Carla P. Gomes, editors, *Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence, AAAI 2008, Chicago, Illinois, USA, July 13-17, 2008*, pages 1481–1484. AAAI Press, 2008. URL: `http://www.aaai.org/Library/AAAI/2008/aaai08-243.php`.

23 Knot Pipatsrisawat and Adnan Darwiche. On the power of clause-learning SAT solvers as resolution engines. *Artificial Intelligence*, 175(2):512–525, February 2011. Preliminary version in *CP '09*.

24 Lawrence Ryan. Efficient algorithms for clause-learning SAT solvers. Master's thesis, Simon Fraser University, 2004.

25 Niklas Sörensson and Armin Biere. Minimizing learned clauses. In *Proceedings of the 12th International Conference on Theory and Applications of Satisfiability Testing (SAT '09)*, volume 5584 of *Lecture Notes in Computer Science*, pages 237–243. Springer, July 2009.

26 Allen Van Gelder. Pool resolution and its relation to regular resolution and DPLL with clause learning. In *Proceedings of the 12th International Conference on Logic for Programming, Artificial Intelligence, and Reasoning (LPAR '05)*, volume 3835 of *Lecture Notes in Computer Science*, pages 580–594. Springer, 2005.

27 Marc Vinyals. Hard examples for common variable decision heuristics. In *Proceedings of the 34th AAAI Conference on Artificial Intelligence (AAAI '20)*, pages 1652–1659, February 2020.

28 Yichen Xie and Alexander Aiken. Saturn: A SAT-Based Tool for Bug Detection. In *Proceedings of the 17th International Conference on Computer Aided Verification, CAV 2005*, pages 139–143, 2005. `doi:10.1007/11513988\_13`.

29 Lintao Zhang, Conor F. Madigan, Matthew W. Moskewicz, and Sharad Malik. Efficient conflict driven learning in Boolean satisfiability solver. In *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design (ICCAD '01)*, pages 279–285, November 2001.

30 Edward Zulkoski, Ruben Martins, Christoph M. Wintersteiger, Jia Hui Liang, Krzysztof Czarnecki, and Vijay Ganesh. The effect of structural measures and merges on SAT solver performance. In John N. Hooker, editor, *Principles and Practice of Constraint Programming - 24th International Conference, CP 2018, Lille, France, August 27-31, 2018, Proceedings*, volume 11008 of *Lecture Notes in Computer Science*, pages 436–452. Springer, 2018. `doi:10.1007/978-3-319-98334-9\_29`.