Electronic Colloquium on Computational Complexity, Report No. 1 (2024)

A Simple Supercritical Tradeoff between Size and Height in Resolution

Sam Buss* Neil Thapen†

December 29, 2023

Abstract

We describe CNFs in n variables which, over a range of parameters, have small resolution refutations but are such that any small refutation must have height larger than n (even exponential in n), where the height of a refutation is the length of the longest path in it. This is called a *supercritical* tradeoff between size and height because, if we do not care about size, every CNF is refutable in height n. A similar result appeared in [Fleming, Pitassi and Robere, ITCS '22], for different formulas using a more complicated construction.

Small refutations of our formula are necessarily highly irregular, making it a plausible candidate to separate resolution from pool resolution, which amounts to separating CDCL with restarts from CDCL without. We are not able to show this, but we show that a simpler version of our formula, with a similar irregularity property, does have polynomial size pool resolution refutations.

1 Introduction

We define families of unsatisfiable CNF formulas, $\Gamma_{a,b}$, $\Gamma_{a,b,c}$ and $\Phi_{b,c,d}$ for which small resolution refutations require large height and, correspondingly, low height refutations require exponential size. For $\Phi_{b,c,d}$, these size-height tradeoffs are called "supercritical" since the height lower bounds can be larger than the number of variables, which is the natural upper bound on refutation height, c.f. [2, 17, 11]. Similar supercritical size-height tradeoff have been independently obtained by Fleming, Pitassi and Robere [11] using an xor-ification method of Razborov [17] that xor-ifies by reusing variables, but our proofs are simpler and more direct.

*Department of Mathematics, U.C. San Diego, sbuss@ucsd.edu. Supported in part by Simons Foundation grant 578919.

†Institute of Mathematics, Czech Academy of Sciences, thapen@math.cas.cz. Supported by the Czech Academy of Sciences (RVO 67985840) and GAČR grant 23-04825S.

ISSN 1433-8092

The principle $\Gamma_{a,b}$ is an induction principle related to PLS [14]. The second principle $\Gamma_{a,b,c}$ is a substitution instance of $\Gamma_{a,b}$ obtained via or-fication, that is, by replacing some of the variables in $\Gamma_{a,b}$ with disjunctions of new variables. The third principle $\Phi_{b,c,d}$ is obtained from $\Gamma_{a,b,c}$ by identifying variables. In other words, $\Phi_{b,c,d}$ can be obtained from $\Gamma_{a,b}$ by an or-ification that reuses variables. Our or-ification with reuse of variables is considerably simpler than the xor-ification of [17]: variables are reused in a pattern based on the base-d representations of integers and this does not depend on expander graphs. For a general statement of our tradeoff result, see Corollary 10 below.

The supercriticality of the height lower bounds means that, for suitable parameters, short resolution refutations of $\Phi_{b,c,d}$ must be highly *irregular*, containing paths that query the same variable many times, simply because the height of the refutation is much more than the number of distinct variables; for similar reasons, small proofs of $\Gamma_{a,b,c}$ must be highly irregular, for suitable parameters. These principles are thus potential candidates for solving an open problem about the relation between resolution and the CDCL algorithms used in SAT solving, by separating the systems of pool-resolution [20] or regWRTI [8, 4, 5, 6], which model CDCL without restarts, from unrestricted resolution. We discuss this problem in Section 5 and show that the principles $\Gamma_{a,b,c}$ *do* have polynomial size pool resolution refutations, and so do not give such a separation. However this remains open for the more complex principles $\Phi_{b,c,d}$.

We recall some standard definitions. A *literal* is a propositional variable x or its negation $\neg x$. A *clause* is a disjunction of literals, and a CNF (conjunctive normal form) formula is a set of clauses, treated as a conjunction of clauses. The *width* of a clause is the number of literals in it. We will often write clauses using a "sequent-style" notation; for example, $x_1 \wedge \cdots \wedge x_k \to y_1 \vee \cdots \vee y_\ell$ means the clause $\neg x_1 \vee \cdots \vee \neg x_k \vee y_1 \vee \cdots \vee y_\ell$.

A *resolution refutation* of a CNF F is a sequence of clauses, where each clause is either from F or is derived from earlier clauses by a *resolution* or *weakening* rule, and where the last clause is the (unsatisfiable) empty clause. The resolution rule allows deriving the clause $C \vee D$ from $C \vee x$ and $D \vee \neg x$, where x is any variable. The weakening rule allows deriving D from C whenever $D \supseteq C$. Such a refutation naturally has the structure of a directed acyclic graph, with the initial clauses of F as sources and the final empty clause as its sink. The *height* of the refutation is the number of edges in the longest path from any source to the sink (this measure is often called *depth*; we prefer to reserve *depth* for the logical depth of formulas appearing in a refutation). The *size* of the refutation is simply the number of clauses.

2 The CNF families

For $a, b > 1$, the CNF $\Gamma_{a,b}$ is a principle about an $a \times b$ rectangular grid. Columns in the grid are numbered with values $x = 0, \ldots, b-1$; we assume b is a power of two. The a many rows in the grid are numbered with $i = 0, \ldots, a-1$; we picture row 0 as the top row, and row $a-1$ as the bottom row. Each node (i, x) in the grid corresponds to a propositional variable $G_{i,x}$. If $G_{i,x}$ is true, we say the node (i, x) is given the value 1; otherwise $G_{i,x}$ is false, and we say (i, x) is given the value 0. There are also propositional variables describing a function $f : [b] \to [b]$ mapping columns to columns, via a binary encoding. (The notation $[b]$ means $\{0, \ldots, b-1\}$.) That is, for each column x there are $\log b$ many variables $f(x)_0, \ldots, f(x)_{\log b - 1}$ giving the value of $f(x)$ in binary. For a column x' we will use the notation $(f(x) = x')$ for the conjunction of $\log b$ literals which asserts that the bits of $f(x)$ match the bits of x'.

Definition 1. *The unsatisfiable* CNF $\Gamma_{a,b}$ *consists of*

1. *The singleton clause* $\neg G_{0,0}$
2. *For each* $i < a - 1$ *and each pair* $x, x' < b$ *the clause*[1]

$$(f(x) = x') \wedge \neg G_{i,x} \to \neg G_{i+1,x'}$$

3. *For each* $x < b$ *the singleton clause* $G_{a-1,x}$.

This expresses that the node at $(0, 0)$ at the top left is 0; that if (i, x) is 0 then $(i{+}1, f(x))$ is 0; and that every node on the bottom row is 1. This is clearly a contradiction, and is very close to what has been called the house-sitting contradiction [10, 7], the iteration principle [9], or the sink-of-dag principle [14]. The novel aspect of $\Gamma_{a,b}$ is that there is one common function f for all rows, instead of having different functions for f at each row, as one might expect.

The definition of $\Gamma_{a,b,c}$ uses a third parameter $c \geq 2$ and variables $G^y_{i,x}$ for all $i < a$, $x < b$ and $y < c$. Then $\Gamma_{a,b,c}$ is the result of replacing the variable $G_{i,x}$ in $\Gamma_{a,b}$ with the disjunction $G^0_{i,x} \vee \cdots \vee G^{c-1}_{i,x}$ and expanding the result as a CNF:

Definition 2. *The unsatisfiable* CNF $\Gamma_{a,b,c}$ *consists of*

1. *For each* $y < c$, *the singleton clause* $\neg G^y_{0,0}$

[1] Here we are using sequent notation for clauses. The sequent displayed here means precisely the clause $\neg(f(x) = x') \vee G_{i,x} \vee \neg G_{i+1,x'}$ where $\neg(f(x) = x')$ denotes the disjunction of the negations of the $\log b$ many literals whose conjunction expresses that $f(x) = x'$.

2. *For each $i < a-1$, each pair $x, x' < b$ and each $y' < c$, the clause*

$$(f(x) = x') \wedge (\bigwedge\nolimits_{y<c} \neg G^y_{i,x}) \to \neg G^{y'}_{i+1,x'}$$

3. *For each $x < b$ the clause $\bigvee_{y<c} G^y_{a-1,x}$.*

The principle $\Gamma_{a,b,c}$ is something like CPLS [12, 18] (and our presentation in this section is modelled on [18]), except that CPLS contains extra functions which allow the analogs of items 2. and 3. to be written with small width, which is something we do not need here.

The node (i, x) now corresponds to $G^0_{i,x} \vee \cdots \vee G^{c-1}_{i,x}$. In this sense each column x of the grid in $\Gamma_{a,b,c}$ contains ac distinct G-variables.

The construction of the final CNF $\Phi_{b,c,d}$ uses a substitution which reduces this number of variables substantially, and thereby allows for supercritical proof height lower bounds. Let $d \geq 2$ be a new parameter and suppose $a = d^c$. We introduce new variables all $H^y_{j,x}$ for $j < d$, $x < b$ and $y < c$.

Definition 3. *Suppose $a = d^c$. The unsatisfiable* CNF $\Phi_{b,c,d}$ *is $\Gamma_{a,b,c}$ after the following substitution. For each $i < a$, let $i_0, \ldots, i_{c-1}$ be the digits of i when written in base d with, for definiteness, the least significant digit first. Then $\Phi_{b,c,d}$ is formed from $\Gamma_{a,b,c}$ by replacing every occurrence of each $G^y_{i,x}$, with the new variable $H^y_{i_y,x}$.*

After the substitution the clause corresponding to (i, x) changes from $G^0_{i,x} \vee \cdots \vee G^{c-1}_{i,x}$ to $H^0_{i_0,x} \vee \cdots \vee H^{c-1}_{i_{c-1},x}$. So whereas in $\Gamma_{a,b,c}$ each column contains ac many G-variables, in $\Phi_{b,c,d}$ each column contains only dc many H-variables. For appropriate choices of the parameters, this can be an exponential reduction in the number of variables.

We remark that the fact that the tuples of indices $i_0, \ldots, i_{c-1}$ of the H-variables appear in each column in lexicographic order is not actually used in our tradeoff arguments below, as they would still work if these tuples were reordered arbitrarily. In fact we anticipate that a version of these CNFs in which the tuples are randomly permuted may be useful for some lower bounds.

Proposition 4. *The formula $\Phi_{b,c,d}$ has*

- *$bcd + b \log b$ variables*
- *$c + (d^c - 1)b^2c + b \leq d^c b^2 c$ clauses, of width at most $c + \log b + 1$*
- *a resolution refutation of size $O(d^c b^2 c)$ and width $c + \log b + 1$.*

Proof. The numbers of variables and clauses of $\Phi_{b,c,d}$ are immediate from the definition. Since $\Phi_{b,c,d}$ is a substitution instance of $\Gamma_{a,b,c}$, the upper bound on proof size will be proved by bounding the size of a proof of $\Gamma_{a,b,c}$.

Letting $a = d^c$, the bounds on proof size come from the natural inductive refutation of $\Gamma_{a,b,c}$ where $a = d^c$. The refutation of $\Gamma_{a,b,c}$ works with each row i in turn, starting at the bottom row $i = a-1$. For each i, the b many clauses $\bigvee_{y<c} G^y_{i,x}$ for $x < b$ are derived.[2] For the base case, $i = a-1$, this is precisely an initial clause 3. of $\Gamma_{a,b,c}$..

The clauses for row i are derived from the clauses for row $i+1$ using the initial clauses 2. The argument is straightforward and splits into b many subcases depending on the values of $f(x)_j$ specifying the value of $f(x)$ in binary. Finally, the clause for row $i = 0$ and column $x = 0$ contradicts the initial clauses 1.

The size and width bounds on the refutation of $\Gamma_{a,b,c}$ are evident by inspection. In particular, the $\log b$ terms in the width comes from handling the bitwise encoding of the function f. □

3 Width versus height

We will first show that every small-width refutation of the simple principle $\Gamma_{a,b}$ must have large height. Then we will observe that this is still true, with the same width and height parameters, for the full principles $\Gamma_{a,b,c}$ and $\Phi_{b,c,d}$. In the next section, an easy random restriction argument will turn this into a size-height tradeoff for $\Phi_{b,c,d}$.

Consider the following Prover-Delayer game played on the variables of $\Gamma_{a,b}$. We divide the variables into *blocks*. For each column x, all the variables $G_{i,x}$ in the column form a block, and all the variables $f(x)_0, \ldots, f(x)_{\log b-1}$ form a second block. So there are $2b$ blocks in total. We only consider assignments that assign whole blocks, and for this reason we may talk about assigning values to $f(x)$ rather than to individual bits $f(x)_j$.

At the beginning of each turn in the game, the Prover's memory contains an assignment to all variables in some set of blocks (this is empty at the start of the game). The Prover can either

- Query a new block, in which case the Delayer reveals an assignment to block's variables and this is added to the Prover's memory, or
- Forget all variables in a block, erasing them from memory.

The Prover wins when the partial assignment in memory falsifies an axiom of $\Gamma_{a,b}$. The Delayer's goal is to force the Prover to make as many queries as possible before winning. The Delayer is free to give different answers each time a block is queried.

We define the *block-width* of a Prover-strategy to be the maximum number of blocks the Prover has in memory at once. Thus the Prover has a simple strategy of block-width 3, which will win in about $2a$ turns. This is

[2] Similar refutations are given for CPLS in [18].

to first query the G-block in column 0, to which the Delayer must reply with an assignment that sets node $(0, 0)$ to 0, to avoid falsifying axiom 1. The Prover can then propagate this 0 down the grid by querying the f-block for the column the current lowest 0 is in, then querying the G-block for the column f points to. He does not need to remember the answers to old queries. He also has a strategy of block-width $b+1$, which requires only $b+1$ queries. This is to query all G-blocks. If no 0 appears, this violates axiom 1; a 0 in the bottom row violates axiom 3; otherwise he queries the f-block for the column in which the lowest 0 appears, and the answer must violate axiom 2.

Lemma 5. *There is a strategy for the Delayer which forces the Prover either to use block-width at least $b/2$, or to make at least $a-1$ queries to f-blocks.*

Proof. We will make the simplifying assumption that the Prover's first query is to G-block 0 (that is, the G-block in column 0), and that this block is never forgotten. This increases the block-width of the Prover's strategy by at most 1. So suppose the Prover's strategy has width $w < b/2$. We will use this assumption to lower bound the number of queries to f-blocks.

We write α for the partial assignment currently in the Prover's memory and t for the number of times that the Prover has queried a value of f so far. We define an *f-path* of length $k \geq 1$ to be a sequence $x_0, \ldots, x_k$ such that $f(x_j) = x_{j+1}$ in α for each $j < k$. The Delayer tries to maintain the following invariants of α, as long as $t \leq a - 1$.

A1. f is a partial injection with no cycles.

A2. For every x, either G-block x is not set by α, or it is set so that at most one row s is 0 and every other row is 1.

A3. Suppose G-block x_0 is set with a 0 in row s. Then $s \leq t$. If furthermore there is an f-path $x_0, \ldots, x_k$ in α, then $s + k \leq t$.

A4. Suppose $x_0, \ldots, x_k$ is a path in α and G-blocks x_0 and x_k are both set and have 0s respectively in rows s and s'. Then $s' = s + k$.

Forgetting a piece of information preserves the invariants. The Delayer responds to queries as follows.

The Prover queries $f(x)$. Call column x' *good* if G-block x' is not set and x' is not on any f-path. By the limit on the block-width, at most $2w$ columns x' are not good, namely one for each G-block set in α and two for each f-block. Therefore at least one good x' must exist. The Delayer chooses any good x' and replies that $f(x) = x'$. This preserves A2 trivially, and A1 and A4 because x' is good. It preserves A3 because t has increased by one.

The Prover queries G-block x. The Prover's first query has this form, with $x = 0$, and in this case the Adversary replies that the top row is 0

and all other rows are 1. Otherwise, if x is on a path which already has a set G-block with a 0 row in it, the Adversary replies with an assignment which puts a 0 in the appropriate row in G-block x to satisfy A4, and a 1 in every other row in the block. By A3 such an assignment exists as long as $t \leq a-1$. This does not affect A1 and preserves A2, A3 and A4 by construction. If there is no such column on the path, the Adversary replies with an assignment setting all variables in the column to 1. □

Theorem 6. *Any resolution refutation of $\Gamma_{a,b}$ of width strictly below $b/2$ must have height at least a.*

Proof. Suppose such a refutation π exists. We make π into a strategy for the Prover in the standard way. That is, we work backwards from the final (empty) clause of π; each resolution step turns into a query of the block in which the resolved variable appears; and the Prover forgets a block whenever we reach a clause in which no variables from that block appear. Thus the block-width of the strategy is at most the width of π, and the game cannot last for more queries than the height of π. But the Delayer-strategy outlined in Lemma 5 forces the Prover to make at least a queries — precisely, he must make at least $a-1$ queries to f-blocks, in addition to which he must query at least one G-block to violate an axiom and win the game. □

Having shown the height-width tradeoff for the simple formula $\Gamma_{a,b}$, we now show that it holds for the full formula $\Phi_{b,c,d}$.

Theorem 7. *Any resolution refutation of $\Phi_{b,c,d}$ of width strictly below $b/2$ must have height at least d^c.*

Proof. Letting $a = d^c$, the proof is essentially the same as that of the previous theorem. We define a Prover-Delayer game played on the variables of $\Phi_{b,c,d}$. As before, we divide the variables into $2b$ blocks with, in each column, one block for all the f-variables and another block for all the H-variables. The Prover can either query a full block, or forget one or more full blocks, and wins when the current assignment falsifies an axiom of $\Phi_{b,c,d}$. We claim that Lemma 5 above also holds for this game, that is, that there is a strategy for the Delayer which forces the Prover either to use block-width at least $b/2$, or to make at least $a-1$ queries to f-blocks.

The strategy is the same as the strategy described in the proof of Lemma 5, except that there the Delayer set G-blocks, in which each node in the column corresponds to a single variable $G_{i,x}$, while now she must set H-blocks, in which each node corresponds to c variables $H^0_{i_0,x} \dots H^{c-1}_{i_{c-1},x}$, behaving as a disjunction, and where each variable appears in several rows in the same block. However, there were only two kinds of assignment the Delayer had to make to a G-block in the proof of Lemma 5: either set every node to 1; or set the node in some given row s to 0, and set every other node to 1. To imitate this strategy, it is enough to have $a+1$ possible assignments to

each H-block x: one which satisfies the disjunction at every node in the column; and, for each $s < a$, one which falsifies the disjunction at row s, and satisfies the disjunction at every other node. By construction of the formula $\Phi_{b,c,d}$, this is easy. In the first case, we just set every variable in the block to 1. In the second case, we set every variable in row s to 0, and set every other variable in the block to 1. This works, because every row other than s contains some variable that does not appear in row s.

The theorem then follows by the same proof as Theorem 6. □

Lemma 5 gave a lower bound on the number of queries to f-blocks; thus our bound on proof height is in actuality a lower bound on the maximum number of resolutions on f-variables along a path in the proof. It is possible to get a lower bound on H-variables as well as f-variables:

Theorem 8. *Any resolution refutation of $\Phi_{b,c,d}$ of width $w < b/2$ must have a path in which there are at least d^c many resolutions on f-variables and d^c/w many resolutions on H-variables.*

Proof sketch. We may extend Lemma 5 to also lower-bound the number of queries to G-variables made by the Prover. For this, the Delayer's strategy is unchanged, but the Delayer maintains one more invariant:

A5. Suppose s is the maximum row with a 0 in any G-block. Then the number of queries made so far to G-blocks is at least s/w.

This is maintained because the only time (other than at the first query) the Delayer sets a new G-block with a 0 row is when it is on a path which already contains a G-block with a 0 row, and paths have maximum length w. □

4 Size versus height

Theorem 9. *Any resolution refutation of $\Phi_{b,c,d}$ of size less than $2^{b^{\frac{1}{4}}}$ must have height at least $a = d^c$.*

Before proving the theorem we describe the supercritical tradeoffs that follow from it, using the estimates from Proposition 4.

For a simple example, given a parameter m, set $b = d = m$ and $c = 3$. Then $\Phi_{b,c,d}$ has $\Theta(m^2)$ variables, $O(m^5)$ clauses, and a refutation of size $O(m^5)$; but any refutation of size less than $2^{m^{1/4}}$ must have height at least m^3. In particular, measured by the number of variables, a polynomial-sized proof exists but any subexponential-sized proof must have superlinear height.

The next corollary constructs a more general family of examples, showing that our tradeoffs are in broadly the same regime as those in [11]. In particular, we can force the height to be exponential in the number of variables, although at the cost of the number of clauses also being exponential.

Corollary 10. *For n, k with $1 \leq k < n/(\log n)^2$, there is a CNF with $\Theta(n)$ variables, n^{k+2} clauses and a refutation of size $O(n^{k+2})$, for which any refutation of size less than $2^{(n/k)^{1/8}}$ must have height at least n^k.*

Proof. The CNF is $\Phi_{b,c,d}$ with $c = k \log n$, $b = n/c$, and $d = 2$ (we are ignoring issues with rounding, and that b should strictly be a power of 2). Observe $\log n \leq b \leq n/\log n$. Referring to Proposition 4, we have $bcd + b \log b$ variables, where $bcd = 2n$ and $b \log b \leq n$. We upper bound the number of clauses and refutation size using $d^c b^2 c = 2^{k \log n}(n/c)^2 c = n^{k+2}/c$. For the tradeoff, we have $n/k = (bk \log n)/k \leq b^2$. Thus, by the theorem, any refutation of size less than $2^{(n/k)^{1/8}}$ has height at least $2^{k \log n}$. □

Proof of Theorem 9. Let Π be a refutation of $\Phi := \Phi_{b,c,d}$ of size less than $2^{b^{\frac{1}{4}}}$. We will define a random restriction which, with high probability, makes Π "narrow" in the following sense: no clause will contain variables from $b/4$ different f-blocks, or from $b/4$ different H-blocks. This will be enough to then apply a width lower bound argument like the one in the proof of Theorem 7.

Fix $w = b/4$. Set $\delta = 1/3$ and let $p = b^\delta/b = b^{-2/3}$. We do the restriction in two, almost independent, stages, to deal with the two different kinds of variables.

Stage 1. Independently for each column x, with probability p put x into a set S_1. For each $x \in S_1$ choose a random column x'. Set $f(x)$ and $f(x')$ both to x'. Set all variables in H-blocks x and x' to 1.

For this construction we want that $|S_1| \leq 2b^\delta$ and that S_1 does not contain 0 or any of the chosen columns x'. The first condition is true with exponentially high probability (in b), and given that the first is true, the second is true with probability at least $(1-p)^{2b^\delta} \geq (e^{-2p})^{2b^\delta} = e^{-4b^{2\delta-1}} = e^{-4b^{-1/3}}$ which asymptotically approaches 1.

If either condition fails, we abandon the construction. Otherwise, we first observe that we have a restriction which does not falsify any axiom of $\Phi_{b,c,d}$. Now let C be any clause containing literals $z_1, \ldots, z_w$ from variables in f-blocks for distinct columns $x_1, \ldots, x_w$. Each $f(x_i)$ is set to some random value with probability p. Hence the probability that z_i is satisfied (that is, is set true) is $p/2$, and the probability that no literal in C is satisfied is at most $(1-p/2)^w < e^{-\frac{1}{2}pw} = e^{-\frac{1}{8}b^{\frac{1}{3}}}$. The refutation Π contains at most $2^{b^{\frac{1}{4}}}$ clauses so, by the union bound, with high probability if we apply this restriction to Π we get a refutation of the restricted Φ in which no clause mentions variables from w or more f-blocks.

Stage 2. Independently for each column x, with probability p put x into a set S_2. Then for each $x \in S_2$, randomly divide the interval $[0, c)$ into two "halves", one of size $\lceil c/2 \rceil$ and the other of size $\lfloor c/2 \rfloor$. For each index y in the first half, set every variable of the form $H^y_{j,x}$ in H-block x to 1. Set all remaining variables in H-block x to 0. Set $f(x) = x$.

We want $|S_2| \leq 2b^\delta$ and that S_2 does not contain 0, any column from S_1, or any of the columns x' from stage 1, and we abandon the construction if any of these conditions fail. By a similar calculation to before, with probability close to 1 we do not abandon it.

As before this restriction (unless it was aborted) does not falsify any axioms. For any H-literal z, the probability that z is set is p and, given that it is set, the probability that it is satisfied is at least $1/3$ (it is $1/3$ only if z is negative and $c = 3$). Therefore, if a clause involves H-variables from w or more columns, it is satisfied with probability at least $(1 - \frac{p}{3})^w = (1 - b^{-2/3}/3)^{b/4}$. Hence, by a similar calculation as above, with high probability, applying this restriction to Π yields a refutation of the restricted Φ in which no clause mentions H-variables from w or more different columns.

Now let ρ be the restriction given by combining the two stages. We have a refutation $\Pi \restriction \rho$ of $\Phi \restriction \rho$ in which each clause mentions f-variables from at most $b/4$ columns and H-variables from at most $b/4$ columns. We now repeat the proof of Theorem 7, with a few small changes.

Say that the restriction ρ *affects* a column x if $x \in S_1$, $x \in S_2$, or x was the value assigned to some f-block x' for $x' \in S_1$. Then ρ affects less than a constant fraction of columns, and for every column x it affects, it either sets at least one variable to 1 in every row of H-block x, or it does not set any variables in the block at all. The Delayer now avoids in her strategy every column x affected by ρ, and because ρ does not set any row to 0, she does not care that f may have collisions and cycles on these columns, since this will never falsify any axiom.

$\Pi \restriction \rho$ gives rise to a Prover strategy in which the Prover knows at most $b/4$ f-blocks and $b/4$ H-blocks. So if the Prover queries $f(x)$, and we want to count the number of "good" columns, at most $b/4$ columns are not good because the Prover knows about that H-block; at most $b/2$ are not good because they are on a path known by the Prover; and only a small fraction, much less than $b/4$, are not good becuase they are affected by ρ and the Delayer is not allowed to touch them. Hence a good column still exists. The rest of the proof is unchanged, and shows that $\Pi \restriction \rho$, and hence Π, has large height, namely height at least $a = d^c$. □

As in [17, 11], we can also use this construction to show a double-exponential lower bound on treelike proof size, over proofs of small width.

Theorem 11. *Any treelike resolution refutation of $\Phi_{b,c,d}$ of width strictly less than $b/4$ must have size at least 2^{d^c}.*

Proof sketch. In the proof of Theorem 7, when the Prover queries $f(x)$, at most $2w$ columns are not good, where w is the block-width of the Prover's strategy. If $w < b/4$ then more than half of the columns are good. The query arose from a resolution on some variable $f(x)_j$ among the $\log b$ variables representing the bits of $f(x)$. So there must be a good column x' for

which this bit is 1, and a good column x'' for which this bit is 0, and the Adversary could potentially reply to the query with either x' or x''. This is enough to give the lower bound on treelike size by adapting the proof of the Impagliazzo-Pudlák game [16]. □

5 CDCL without restarts

The CDCL algorithm [13] is at the core of most modern SAT solvers. Given a set of clauses, it grows a partial assignment, called the *trail*, until it contradicts a clause in the set; from this it learns a new clause implied by the current set, forgets some recently assigned values from the trail, and begins growing it again. Eventually it either builds an assignment which satisfies every clause, or learns the empty clause and declares the initial set to be unsatisfiable. An important additional heuristic, which seems in practice to be necessary for effective SAT solving, is to frequently *restart*, meaning, throw away the current trail and begin again, keeping just the learned clauses.

It is known [3, 15] that, for an unsatisfiable CNF F, the length of the shortest run of CDCL on F is polynomially related to the size of the shortest resolution refutation of F, provided that CDCL is allowed unlimited restarts. This is open for CDCL without restarts. In particular it is possible that disallowing restarts means that CDCL computations must be exponentially longer than resolution refutations, on some CNFs. The *pool resolution* proof system was defined in [20] as a restricted version of resolution which captures CDCL without restarts, in such a way that we could resolve the question above by either showing that pool resolution simulates resolution, or showing superpolynomial bounds in pool resolution for a CNF with short resolution refutations.

A resolution refutation is *regular* if, on every path through the refutation from an axiom to the final clause, no variable is resolved on twice. Pool resolution simulates regular resolution [20], so natural candidates for superpolynomially separating resolution from pool resolution are CNFs which are already known to separate resolution from regular resolution. There are not many such CNFs known. Three examples, from [1, 19] are the guarded graph tautologies, the Stone tautologies, and the guarded pebbling tautologies. All three have been shown to have polynomial size pool resolution refutations [4, 5, 6].

We observe that it follows from Theorem 9 that, for suitable choices of parameters, $\Phi_{b,c,d}$ has no subexponential-size regular resolution refutations. This is simply because a regular refutation cannot have height greater than the number of variables. (Strictly, for this to be true we should not count weakening steps when we measure height; but the height lower bounds in this paper do not count weakenings.) Indeed, any small refutation must be highly irregular, with variables reused many times. Setting $a = d^c$, even for

the formula $\Gamma_{a,b,c}$ any small refutation must be highly irregular, as it must have paths that resolve on f-variables d^c times, and this is greater than the number of f-variables.

We show that nevertheless, like the three CNF families mentioned above, $\Gamma_{a,b,c}$ has polynomial size pool resolution refutations. We have not been able to show a similar results for $\Phi_{b,c,d}$ and it remains a possible candidate for the separation.

We will not use the original definition of pool resolution but will work with an equivalent system called regRTL (standing for *regular resolution trees with lemmas*). There are also generalizations of these concepts that allow certain types of weakenings or "w-resolution" inferences, but we will not need them here – see [8]. We recall some definitions from [8]. A resolution proof is now represented as an (ordered) tree T with nodes labelled with clauses. The root node of T is at the bottom, and is labelled with the empty clause. The *post-order* ordering $<_T$ of the clauses in T is defined as follows: if u is a clause in T, v and w are clauses in the left and right subtrees (respectively) above U, then $v <_T w <_T u$; intuitively, clauses earlier in the post-order represent clauses learnt earlier in the CDCL computation.

Definition 12. ([20, 8]) *A* pool resolution refutation*, or* regRTL refutation*, of a set of clauses F is a resolution proof tree T such that: (a) each leaf is labeled with either a clause of F or a clause C (called a "lemma") that appears earlier in the tree in the $<_T$ ordering; (b) each internal node is obtained by resolution from its two children; (c) the proof tree is regular, in that no branch in T uses the same resolution variable twice; (d) the root is labelled with the empty clause.*

Theorem 13. *The formulas $\Gamma_{a,b,c}$ have polynomial size* regRTL *refutations.*

Proof. We first restate the axioms of $\Gamma_{a,b,c}$ in a slightly different notation where in particular, to improve clarity of the figures below, we introduce a symbol $\mathbf{G}_{i,x}$ to stand for the disjunction $\bigvee_{z<c} G^z_{i,x}$. The axioms become

1. $\neg G^y_{0,0}$ for $y < c$
2. $(f(x) = x') \wedge G^y_{i+1,x'} \rightarrow \mathbf{G}_{i,x}$ for $i < a-1$, $x, x' < b$ and $y < c$
3. $\mathbf{G}_{a-1,x}$ for $x < b$.

Broadly, the refutation works its way up from the bottom row of the grid of $\Gamma_{a,b,c}$, row $a-1$, to the top row, row 0. For each row i it derives all clauses $\mathbf{G}_{i,x}$ for $x < b$.

For $i = a-1$, these are just axiom 3. For $i < a-1$, we will make use of a treelike subproof $A_{i,x}$ which derives $\mathbf{G}_{i,x}$ assuming we already have $\{\mathbf{G}_{i+1,x'} : x' \in [b]\}$. The structure of $A_{i,x}$ is shown in Figure 1. For each $x' \in [b]$, it introduces all axioms $\{(f(x){=}x'), G^y_{i+1,x'} \rightarrow \mathbf{G}_{i,x}\}_{y\in[c]}$ and resolves these

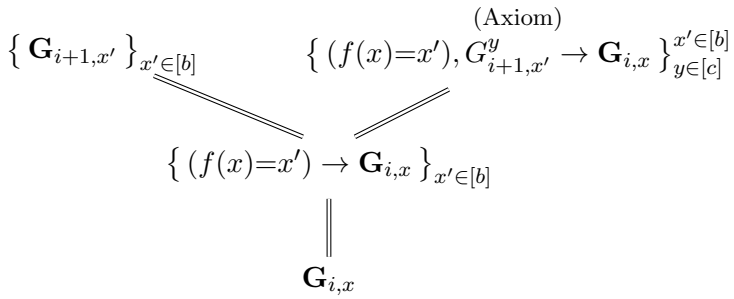

Figure 1: The structure of the tree $A_{i,x}$. This is an abstracted diagram where we do not show individual resolution steps. Clauses in $\Gamma_{a,b,c}$ are labelled as axioms. The other leaf clauses are available as learned clauses.

with $\mathbf{G}_{i+1,x'}$ on the variables $G^y_{i+1,x'}$ to derive $(f(x){=}x') \to \mathbf{G}_{i,x}$. Thus we have $\{(f(x){=}x') \to \mathbf{G}_{i,x}\}_{x' \in [b]}$ and we can derive $\mathbf{G}_{i,x}$ from these by resolving away all the variables $(f(x))_j$ of $f(x)$, since the antecedents run through all possible patterns of signs on these variables.

It would be easy to make the trees $A_{i,x}$ into a resolution refutation of $\Gamma_{a,b,c}$, if we did not care about it being treelike or satisfying the regularity condition; essentially this is what is described in Proposition 4. But this naive refutation is not treelike, as it reuses the clauses $\mathbf{G}_{i,x}$ in an uncontrolled way, and is not regular, as paths through it typically will resolve several times on the same f-variables. Instead we will introduce a scaffolding of clauses, which do not derive anything very useful themselves, but which give us the structure of a tree with appropriate places to put the trees $A_{i,x}$ as subtrees in a way that satisfies the rules of regRTL.

We inductively define a sequence of trees $T_{a-1}, \ldots, T_1$, where T_i is a derivation of $\mathbf{G}_{i-1,0}$ and contains T_{i+1} as a subtree. Our final refutation of $\Gamma_{a,b,c}$ consists of T_1, which as just stated derives $\mathbf{G}_{0,0}$, followed by resolution steps with all instances of axiom 1, that is, with all clauses in $\{\neg G^y_{0,0}\}_{y \in [c]}$, giving the empty clause. The structure of T_i, for $i < a-1$, is shown in Figure 2 and we now go through its important properties.

The base-case tree T_{a-1} has the same structure as in the figure, except that we delete the subtrees on the left marked as $T_{i+1}, A_{i,1}, \ldots, A_{i,b-1}$; we do not need them, since we are given the clauses $\mathbf{G}_{a-1,0}, \ldots, \mathbf{G}_{a-1,b-1}$ directly as axioms. We have not described the subtrees of the form $B^y_{i,x}$ in the figure yet — we postpone this to the end of the proof.

Tree T_i has as a backbone a sequence of clauses of the form $G^0_{i,0}, \ldots, G^0_{i,x} \to \mathbf{G}_{i-1,0}$. For each $x < b-1$ this clause has exactly $c+1$ parents, as shown in the figure, and is derived from them by c resolution steps over the variables $G^0_{i,x}, \ldots, G^{c-1}_{i,x}$. The details of these resolutions are not shown, but preserve that T_{i+1} and the trees $A_{i,x}$ appear as left subtrees.

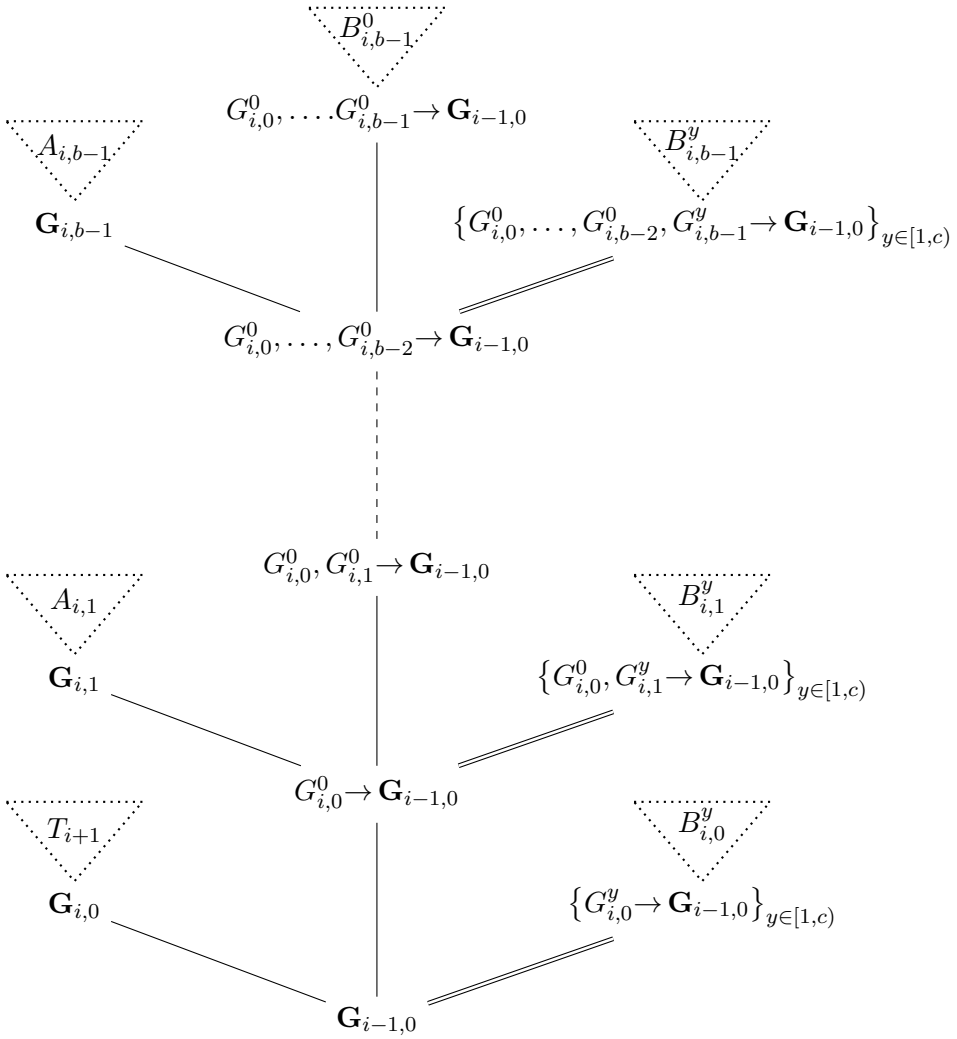

Figure 2: The structure of the tree T_i, for $i < a - 1$.

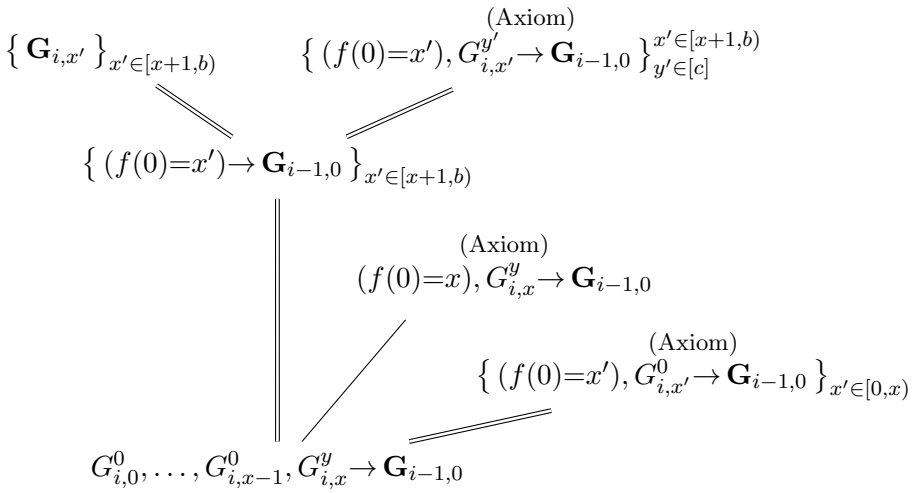

Figure 3: The structure of the tree $B^y_{i,x}$.

Tree T_i contains T_{i+1} as a leftmost subtree in the post-order, and inductively T_{i+1} contains every clause in $\mathbf{G}_{i+1,x}$ for $x \in [b]$. Thus we are allowed to reuse these clauses anywhere else in T_{i+1}, and in fact we use them in the subtrees of the form $A_{i,x}$ shown in the figure.

The trees $A_{i,x}$ only include resolutions on f-variables and on G-variables from row $i+1$ (that is, of the form $G^y_{i+1,x'}$). Inductively, T_{i+1} only includes resolutions on f-variables and on G-variables from rows $j \geq i+1$. Otherwise, all resolutions in the body of T_i are on G-variables from row i, with no variable resolved twice. Thus we satisfy the regularity condition on the parts of T_i described so far.

It remains to handle the "loose ends" left in T_i, that is, the clauses of the form $G^0_{i,0}, \ldots, G^0_{i,x-1}, G^y_{i,x} \to \mathbf{G}_{i-1,0}$ on the right and at the top of the tree. We will describe trees $B^y_{i,x}$ that derive these. Note that $B^y_{i,x}$ is allowed to use the clauses $\{\mathbf{G}_{i,x'}\}_{x' \in [b]}$ since these were derived in an earlier part of the proof in the post-order, at the conclusions of the trees $A_{i,x}$ and T_{i+1}. Note also that on the path from the root of T_i to the conclusion of $B^y_{i,x}$, the only resolutions that occur are on variables of the form $G^{y'}_{i,x'}$ with $x' \leq x$, so we will avoid resolving on these inside $B^y_{i,x}$.

The structure of $B^y_{i,x}$ is shown in Figure 3. It is a variant of the tree $A_{i-1,0}$, with the main difference that we avoid resolving away as many G-variables. First, for each $x' > x$, from $\mathbf{G}_{i,x'}$ and the axioms $\{(f(0){=}x'), G^{y'}_{i,x'} \to \mathbf{G}_{i-1,0}\}_{y' \in [c]}$ we derive $(f(0){=}x') \to \mathbf{G}_{i-1,0}$ by resolving on the variables $G^{y'}_{i,x'}$. Then we resolve on the variables $(f(0))_j$ to combine the clauses $(f(0){=}x') \to \mathbf{G}_{i-1,0}$ for $x' > x$, the axioms $(f(0){=}x'), G^0_{i,x'} \to \mathbf{G}_{i-1,0}$ for

$x' < x$, and the single axiom $(f(0){=}x), G^y_{i,x} \to \mathbf{G}_{i-1,0}$ (which is where y appears), to get the desired result. Thus we satisfy the regularity condition.

This completes the description of the RegRTL refutation. It is polynomial size by construction. □

References

[1] Michael Alekhnovich, Jan Johannsen, Toniann Pitassi, and Alasdair Urquhart. An exponential separation between regular and general resolution. *Theory of Computing*, 3(5):81–102, 2007.

[2] Paul Beame, Chris Beck, and Russell Impagliazzo. Time-space tradeoffs in resolution: Superpolynomial lower bounds for superlinear space. *SIAM Journal on Computing*, 43(4):1612–1645, 2016.

[3] Paul Beame, Henry A. Kautz, and Ashish Sabharwal. Towards understanding and harnessing the potential of clause learning. *Journal of Artificial Intelligence Research*, 22:319–351, 2004.

[4] Maria Luisa Bonet and Samuel R. Buss. An improved separation of regular resolution from pool resolution and clause learning. In *Proc. 15th International Conference on Theory and Applications of Satisfiability Testing – SAT 2012*, Lecture Notes in Computer Science #7317, pages 45–57, 2012.

[5] Maria Luisa Bonet, Samuel R. Buss, and Jan Johannsen. Improved separations of regular resolution from clause learning proof systems. *Journal of Artificial Intelligence Research*, 49:669–703, 2014.

[6] Sam Buss and Leszek Kołodziejczyk. Small stone in pool. *Logical Methods of Computer Science*, 10(2):Paper 2, 2014.

[7] Samuel R. Buss. Lower bounds on Nullstellensatz proofs via designs. In P. Beame and S. Buss, editors, *Proof Complexity and Feasible Arithmetics*, pages 59–71. American Mathematical Society, 1998.

[8] 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(4:13):1–18, 2008.

[9] Samuel R. Buss and Jan Krajíček. An application of Boolean complexity to separation problems in bounded arithmetic. *Proc. London Math. Society*, 69:1–21, 1994.

[10] Matt Clegg, Jeff Edmonds, and Russell Impagliazzo. Using the Groebner basis algorithm to find proofs of unsatisfiability. In *Proceedings of the Twenty-eighth Annual ACM Symposium on the Theory of Computing*, pages 174–183, 1996.

[11] Noah Fleming, Toniann Pitassi, and Robert Robere. Extremely deep proofs. In *Proc. 13th Innovations in Theoretical Computer Science Conference, ITCS*, LIPIcs 215, pages 70:1–23, 2022.

[12] Jan Krajíček, Alan Skelley, and Neil Thapen. NP search problems in low fragments of bounded arithmetic. *Journal of Symbolic Logic*, 72(2):649–672, 2007.

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

[14] Christos H. Papadimitriou. On the complexity of the parity argument and other inefficient proofs of existence. *Journal of Computer and System Sciences*, 48(3):498–532, 1994.

[15] Knot Pipatsrisawat and Adnan Darwiche. On the power of clause-learning SAT solvers as resolution engines. *Artificial Intelligence*, 172(2):512–525, 2011.

[16] Pavel Pudlák and Russell Impagliazzo. A lower bound for DLL algorithms for k-SAT (preliminary version). In *Proc. 11th AM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 128–136, 2000.

[17] Alexander A. Razborov. A new kind of tradeoffs in propositional proof complexity. *J. ACM*, 62(3):16:1–14, 2016.

[18] Neil Thapen. A trade-off between length and width in resolution. *Theory of Computing*, 12(5):1–14, 2016.

[19] Alasdair Urquhart. A near-optimal separation of regular and general resolution. *SIAM Journal on Computing*, 40(1):107–121, 2011.

[20] Allen Van Gelder. Pool resolution and its relation to regular resolution and DPLL with clause learning. In *Logic for Programming, Artificial Intelligence, and Reasoning (LPAR 2005)*, Lecture Notes in Computer Science 3835, pages 580–594. Springer-Verlag, 2005.

ECCC ISSN 1433-8092

https://eccc.weizmann.ac.il