



Contents lists available at ScienceDirect

Information Processing Letters

journal homepage: www.elsevier.com/locate/ipl

A simple supercritical tradeoff between size and height in resolution

Sam Buss^{a,1}, Neil Thapen^{b,2}^a Department of Mathematics, U.C. San Diego, USA^b Institute of Mathematics, Czech Academy of Sciences, Czech Republic

A B S T R A C T

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 . Our proof method uses a simple construction, based on or-ification and base d representations of integers, to reduce the number of variables. A similar result appeared in [Fleming, Pitassi and Robere, ITCS '22], for different formulas using a more complicated construction for reducing the number of variables.

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 restarts. We are not able to show this. In the other direction, we show that a simpler version of our formula, with a similar irregularity property, *does* have polynomial size pool resolution refutations and thus does not provide such a separation for CDCL.

1. Introduction

Resolution is an important propositional proof system for many applications. The survey [5] describes some of the theoretical aspects of the proof complexity of resolution; the survey [9] discusses the relation between resolution and unexpectedly successful present-day SAT (Satisfiability) solvers. Other significant lower bounds on the complexity of resolution have been established by Raz [21], Razborov [22] and Atserias-Müller [2]. The present paper augments the work of [3,23,15] by giving new supercritical size-height tradeoffs for resolution proof complexity with new and simpler constructions.³

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, cf. [3,23,15]. Similar supercritical size-height tradeoffs have been independently obtained by Fleming, Pitassi and Robere [15] using an xor-ification method of Razborov [23] that xor-ifies by reusing variables. Our proofs are simpler and more direct, and use or-ification and the base d representations of integers.

The principle $\Gamma_{a,b}$ is an induction principle related to PLS [18]. The second principle $\Gamma_{a,b,c}$ is a substitution instance of $\Gamma_{a,b}$ obtained via or-ification, 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 [23]: 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.

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 refutations 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 [26] or regWRTI [11,6–8], 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

E-mail addresses: sbuss@ucsd.edu (S. Buss), thapen@math.cas.cz (N. Thapen).

¹ Supported in part by Simons Foundation grant 578919.

² Supported by the Institute of Mathematics, Czech Academy of Sciences (RVO 67985840) and GAČR grant 23-04825S.

³ Since this paper was submitted, a stronger result about tradeoffs has appeared, where they are supercritical in the total size of the formula rather than just in the number of variables [14].

<https://doi.org/10.1016/j.ipl.2025.106589>

Received 18 December 2023; Received in revised form 25 April 2025; Accepted 14 May 2025

0020-0190/© 2025 Published by Elsevier B.V.

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, $x_1 \wedge \dots \wedge x_k \rightarrow y_1 \vee \dots \vee y_\ell$ means the clause $\neg x_1 \vee \dots \vee \neg x_k \vee y_1 \vee \dots \vee y_\ell$.

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 $x = 0, \dots, b-1$; we assume b is a power of two. The a many rows in the grid are numbered with $i = 0, \dots, a-1$; we picture row 0 as the top row, and row $a-1$ as the bottom row (see Fig. 1). 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 $f : [b] \rightarrow [b]$ mapping columns to columns, via a binary encoding. (The notation $[b]$ means $\{0, \dots, b-1\}$.) That is, for each column x there are $\log b$ many variables $f(x)_0, \dots, f(x)_{\log b-1}$ 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' .

The purpose of using a binary encoding is to describe values of $f(x)$ with as few variables as possible. This allows the formulas $\Gamma_{a,b,c}$, and $\Phi_{b,c,d}$ defined below have only small width clauses when the parameter c is small.

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 clause⁴

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

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 [13,10], the iteration principle [12], or the sink-of-dag principle [18]. The novel aspect of $\Gamma_{a,b}$ is that there is one common function f for all rows, instead of 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 $G_{i,x}^y$ for $i < a$, $x < b$ and $y < c$. Then $\Gamma_{a,b,c}$ is the result of replacing the vari-

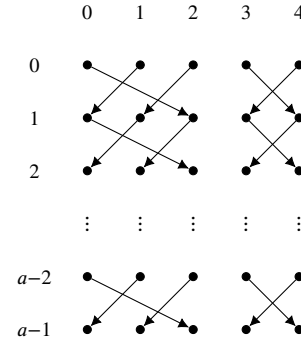


Fig. 1. The CNF formula $\Gamma_{a,b}$ is visualized as a $[a] \times [b]$ array of variables. Shown with $b = 5$. Implications (actually, reverse implications) are shown by arrows. E.g., in the figure $f(0) = 2$, $f(1) = 0$, etc. The same f is used for all rows.

able $G_{i,x}$ in $\Gamma_{a,b}$ with the disjunction $G_{i,x}^0 \vee \dots \vee G_{i,x}^{c-1}$ 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 $\neg G_{0,0}^y$
2. For each $i < a-1$, each pair $x, x' < b$ and each $y' < c$, the clause

$$(f(x) = x') \wedge \left(\bigwedge_{y < c} \neg G_{i,x}^y \right) \rightarrow \neg G_{i+1,x'}^{y'}$$

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

The principle $\Gamma_{a,b,c}$ is something like CPLS [16,24] (and our presentation in this section is modelled on [24]), 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 $G_{i,x}^0 \vee \dots \vee G_{i,x}^{c-1}$. 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 $H_{j,x}^y$ for $j < d$, $x < b$ and $y < c$. The next definition acts by replacing, for fixed values of x, y , each of the a many variables $G_{i,x}^y$ with one of the d many variables $H_{i',x}^y$ (where $0 \leq i' < d$). The subscript i' is the y th digit in the base d representation of i .

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 $i_0, \dots, i_{c-1}$ 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 $G_{i,x}^y$ with the new variable $H_{i_y,x}^y$.

After the substitution the clause corresponding to (i, x) changes from $G_{i,x}^0 \vee \dots \vee G_{i,x}^{c-1}$ to $H_{i_0,x}^0 \vee \dots \vee H_{i_{c-1},x}^{c-1}$. 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.

An important property of Definition 3 is that it is possible to set the clause $H_{i_0,x}^0 \vee \dots \vee H_{i_{c-1},x}^{c-1}$, corresponding to the node in row i and column x , to be false without forcing any other clause in the same column to be false.

We remark that the fact that the tuples of indices $i_0, \dots, i_{c-1}$ 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.

⁴ 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 $f(x) = x'$.

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

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

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 refutation of $\Gamma_{a,b,c}$. Letting $a = d^c$, the bounds on proof size come from the natural inductive refutation of $\Gamma_{a,b,c}$. 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 $\bigvee_{y < c} G_{i,x}^y$ for $x < b$ are derived.⁵ 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. of $\Gamma_{a,b,c}$, which can be rewritten as

$$f(x) = x' \rightarrow \bigvee_{y < c} G_{i,x}^y \vee \neg G_{i+1,x}^y. \quad (1)$$

The derivation is straightforward: fixing values for x and x' , the clauses (1) are resolved in turn against $\bigvee_{y < c} G_{i+1,x}^y$ for $y = 0, \dots, c - 1$.

This results in the clause $f(x) = x' \rightarrow \bigvee_{y < c} G_{i,x}^y$. Repeating for all values of x' , and resolving the resulting clauses on the literals of $f(x) = x'$ yields $\bigvee_{y < c} G_{i,x}^y$ as desired. The largest clauses are those derived by the first resolution against an initial clause (1), and have width $2c + \log b - 1$.

Finally, the clause for row $i = 0$ and column $x = 0$ contradicts the initial clauses 1 in $\Gamma_{a,b,c}$.

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 come from handling the bitwise encoding of the function f . $\square$

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 $\Phi_{b,c,d}$.

We define a Prover-Delayer game played on the variables of $\Gamma_{a,b}$. The Prover's goal is to falsify one of the axioms, that is, one of the clauses in $\Gamma_{a,b}$. The Delayer's goal is force the Prover either to make a lot of queries or to simultaneously have in memory the values of a lot of variables (see Lemma 5).

The Prover acts by querying variables; the Delayer responds with truth assignments to the queried variables. For this, we divide the variables into *blocks* and the Prover will always query an entire block at once. For each column x , all the variables $G_{i,x}$ in the column form a block. In addition, for each x , the variables $f(x)_0, \dots, f(x)_{\log b - 1}$ form a block. So there are $2b$ blocks in total, two for each column. 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 the block's variables and this is added to the Prover's memory, or
- Forget all variables in a block, erasing them from memory.

The Delayer is free to give different answers if a forgotten block is queried again.

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 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.

The Prover 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. Suppose the Prover's strategy has block-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 in α of length k to be a sequence $x_0, \dots, x_k$ such that the partial assignment α sets $f(x_j) = x_{j+1}$ for each $j < k$. We allow $k = 0$; that is, any x_0 is an f -path of length 0. 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, with $0 \notin \text{im } f$.
- 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 $x_0, \dots, x_k$ is an f -path in α of length k and either
 - (i) $x_0 = 0$ and $s = 0$.
 - (ii) G -block x_0 is set and has a 0 in row s .
 Then $s' := s + k < t$ and if G -block x_k is set then it has a 0 in row s' .

Condition A3(i) is present since even if G -block 0 is not set, it must implicitly have a 0 in row 0 to avoid later violating axiom 1 of $\Gamma_{a,b}$, namely $\neg G_{0,0}$.

Conditions A1–A3 are trivially valid when the game starts, and forgetting a piece of information preserves them. For other moves, the Delayer responds to queries as follows.

The Prover queries $f(x)$. Call column x' *good* if $x' \neq 0$, the G -block x' is not set, and x' is not on any f -path of non-zero length. By the limit on block-width at most $2w + 1$ columns x' are not good, namely one for each G -block in α and two for each f -block in α , plus column 0. Therefore some good x' must exist. The Delayer chooses any good x' and replies that $f(x) = x'$. This preserves A2 trivially, and A1 and A3 because x' is good and t has increased by one.

The Prover queries G -block x . The Prover replies so as to preserve condition A3. Namely, if there is an f -path from some column x' to x of length k in α and a value s such that either (i) $x' = 0$ and $s = 0$ or (ii) the G -block x' is set and has a 0 in row s , then the Delayer replies with the assignment that puts a 0 in row $s' := s + k$ in G -block x and sets the rest of the rows of G -block x to 1. (This is possible since $s' \leq t < a$ by A3.) If there is no such x' and s , the Delayer replies with the assignment setting all variables in the G -block x to 1. In light of A1–A3, noting especially that f is injective, these rules for Delayer are unambiguous. Condition A1 is trivially preserved and conditions A2 and A3 are preserved by construction. $\square$

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

⁵ Similar refutations are given for CPLS in [24].

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, and then to formally win he must falsify an axiom, which requires querying at least one G -block since every axiom contains a G -variable. $\square$

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 $H_{i_0,x}^0 \dots H_{i_{c-1},x}^{c-1}$, 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. $\square$

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:

A4. Suppose s is a 0-row 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 the Delayer sets a new G -block with a 0 row is when it is on a path which already contains either a G -block with a 0 row or contains column 0, and paths have maximum length w . $\square$

4. Size versus height

Theorem 9. *Any resolution refutation of $\Phi_{b,c,d}$ of size less than $2^{b^{1/4}}$ 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 and depth lower bounds are very close to those in [15]. In particular, like [15], 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 $2^{k \log n} = n^k$. $\square$

Proof of Theorem 9. Let Π be a refutation of $\Phi := \Phi_{b,c,d}$ of size less than $2^{b^{1/4}}$. 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). Given that the first is true, for the second condition we then want our $|S_1|$ random choices of x' all to fall outside S_1 , which happens with probability $(1 - |S_1|/b)^{|S_1|} \geq (e^{-2|S_1|/b})^{|S_1|} \geq e^{-8b^{2\delta-1}} = e^{-8b^{-1/3}}$ 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 $z_1, \dots, z_w$ from variables in f -blocks for distinct columns $x_1, \dots, x_w$. 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 at least $p/2$, and the probability that no literal in C is satisfied is at most $(1 - p/2)^w < e^{-\frac{1}{2}pw} = e^{-\frac{1}{8}b^{1/3}}$. The refutation Π contains at most $2^{b^{1/4}}$ 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 $H_{j,x}^y$ 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 $\Pi \upharpoonright \rho$ of $\Phi \upharpoonright \rho$ 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.

We say that 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 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.

$\Pi \upharpoonright \rho$ 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 because 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 $\Pi \upharpoonright \rho$, and hence Π , has large height, at least $a = d^c$. $\square$

As in [23,15], 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 [20]. $\square$

5. CDCL without restarts

The CDCL algorithm [17] 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 [4,19] 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 [26] 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 [26], 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 [1,25] 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 [6–8].

We observe that it follows from Theorem 9 that, for suitable choices of parameters (such as $b = d = m$ and $c = 3$, discussed just after Theorem 9), $\Phi_{b,c,d}$ 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 are unable to show a similar result 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 [11]. We recall some definitions from [11]. 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 ([26,11]). 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 $G_{i,x}$ to stand for the disjunction $\bigvee_{z < c} G_{i,x}^z$. The axioms become

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

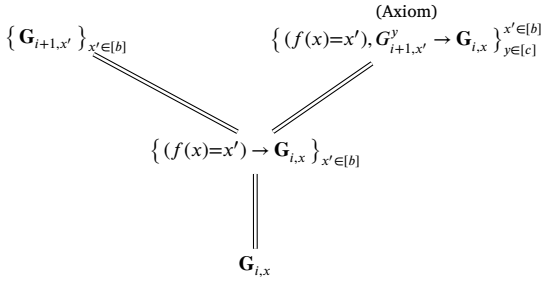


Fig. 2. The structure of the tree $A_{i,x}$. This is an abstracted diagram where we do not show individual resolution steps. Clauses in $\Gamma_{a,b,c}$ are labelled as axioms. The other leaf clauses are available as learned clauses.

3. $G_{a-1,x}$ for $x < b$.

When these are rewritten as clauses, they yield exactly the same clauses as in Definition 2.

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 $G_{i,x}$ 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 $G_{i,x}$ assuming we already have $\{G_{i+1,x'} : x' \in [b]\}$. The structure of $A_{i,x}$ is shown in Fig. 2. For each $x' \in [b]$, it introduces all axioms $\{(f(x)=x'), G_{i+1,x'}^y \rightarrow G_{i,x}\}_{y \in [c]}$ and resolves these with $G_{i+1,x'}$ on the variables $G_{i+1,x'}^y$ to derive $(f(x)=x') \rightarrow G_{i,x}$. Thus we have $\{(f(x)=x') \rightarrow G_{i,x}\}_{x' \in [b]}$ and we can derive $G_{i,x}$ 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 $G_{i,x}$ 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 $T_{a-1}, \dots, T_1$, where T_i is a derivation of $G_{i-1,0}$ 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 $G_{0,0}$, followed by resolution steps with all instances of axiom 1, that is, with all clauses in $\{\neg G_{0,0}^y\}_{y \in [c]}$, giving the empty clause. The structure of T_i , for $i < a-1$, is shown in Fig. 3 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 $T_{i+1}, A_{i,1}, \dots, A_{i,b-1}$; we do not need them, since we are given the clauses $G_{a-1,0}, \dots, G_{a-1,b-1}$ directly as axioms. We have not described the subtrees of the form $B_{i,x}^y$ 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 $G_{i,0}^0, \dots, G_{i,x}^0 \rightarrow G_{i-1,0}$. 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 $G_{i,x}^0, \dots, G_{i,x}^{c-1}$. The details of these resolutions are not shown, but preserve that T_{i+1} and the trees $A_{i,x}$ appear as left subtrees.

Tree T_i contains T_{i+1} as a leftmost subtree in the post-order, and inductively T_{i+1} contains every clause in $G_{i+1,x}$ for $x \in [b]$. Thus we are allowed to reuse these clauses anywhere else in T_i , 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 $G_{i+1,x'}^y$). 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 -

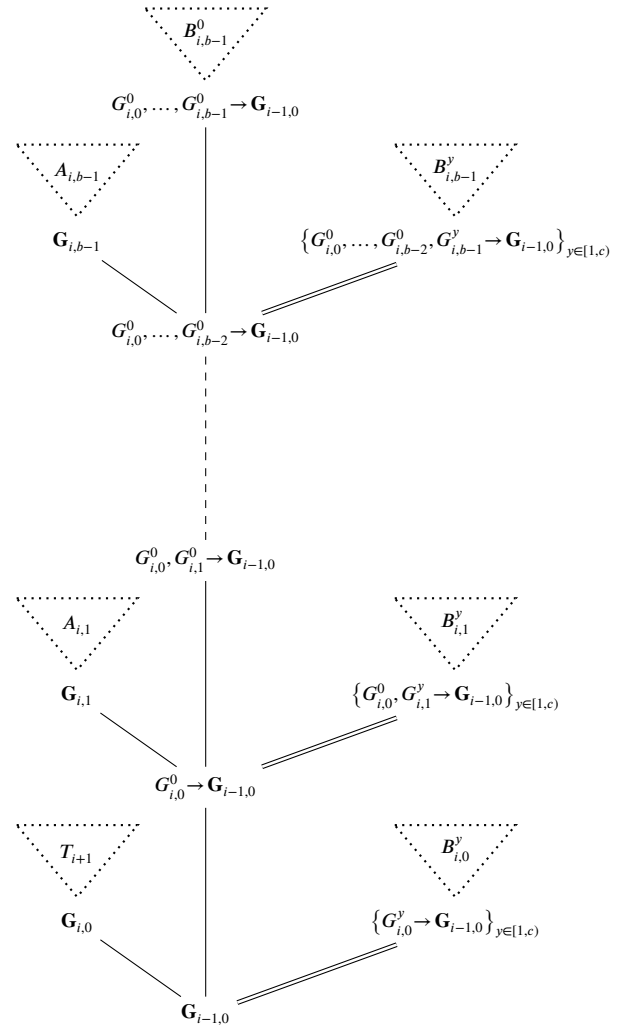


Fig. 3. The structure of the tree T_i , for $i < a-1$.

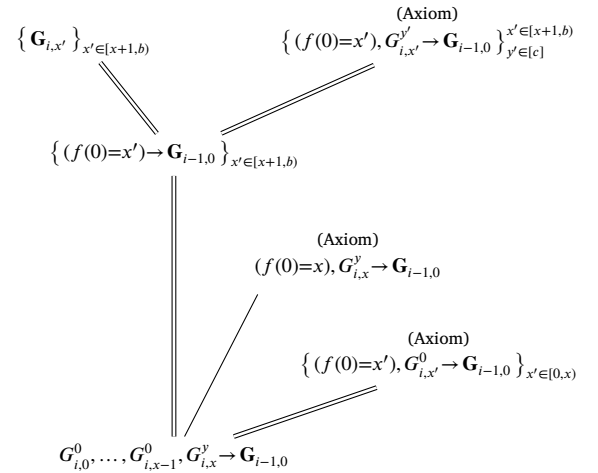


Fig. 4. The structure of the tree $B_{i,x}^y$.

variables from row i , with no variable resolved twice. Thus we satisfy the regularity condition on the parts of T_i described so far.⁶

⁶ This is the part of the argument that fails if we try to prove Theorem 13 for $\Phi_{b,c,d}$ instead of $\Gamma_{a,b,c}$.

It remains to handle the “loose ends” left in T_i , that is, the clauses of the form $G_{i,0}^0, \dots, G_{i,x-1}^0, G_{i,x}^y \rightarrow G_{i-1,0}$ on the right and at the top of the tree. We will describe trees $B_{i,x}^y$ that derive these. Note that $B_{i,x}^y$ is allowed to use the clauses $\{G_{i,x'}^y\}_{x' \in [b]}$ 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 $B_{i,x}^y$, the only resolutions that occur are on variables of the form $G_{i,x'}^y$ with $x' \leq x$, so we will avoid resolving on these inside $B_{i,x}^y$.

The structure of $B_{i,x}^y$ is shown in Fig. 4. 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 $G_{i,x'}^y$ and the axioms $\{(f(0)=x'), G_{i,x'}^y \rightarrow G_{i-1,0}\}_{y' \in [c]}$ we derive $(f(0)=x') \rightarrow G_{i-1,0}$ by resolving on the variables $G_{i,x'}^y$. Then we resolve on the variables $(f(0))_j$ to combine the clauses $(f(0)=x') \rightarrow G_{i-1,0}$ for $x' > x$, the axioms $(f(0)=x'), G_{i,x'}^0 \rightarrow G_{i-1,0}$ for $x' < x$, and the single axiom $(f(0)=x), G_{i,x}^y \rightarrow G_{i-1,0}$ (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, as there are many subtrees $A_{i,x}$, many subtrees $B_{i,x}^y$ and many structures $T_i \setminus T_{i+1}$, and each of these structures has polynomially many clauses. $\square$

The proof method of Theorem 13 fails if it is applied to $\Phi_{b,c,d}$ instead of $\Gamma_{a,b,c}$. The subtrees $A_{i,x}$ resolve on the variables $G_{i+1,x}^y$ and the subtree T_{i+1} resolves on the variables $G_{i+1,0}^y$. The subtree T_i resolves on the variables $G_{i,x}^y$ below $A_{i,x}$ and on the variables $G_{i,x}^y$ below $T_{i+1,x}$. In $\Phi_{b,c,d}$, some of the variables $G_{i,x}^u$ and $G_{i+1,x}^y$ become identified by the substitution of Definition 3 and this destroys the regularity property needed for RegRTL or pool refutations.

CRediT authorship contribution statement

Sam Buss: Writing – review & editing, Writing – original draft. **Neil Thapen:** Writing – review & editing, Writing – original draft.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgement

We thank the two anonymous referees for helpful comments and corrections which improved the paper.

Data availability

No data was used for the research described in the article.

References

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

- [2] Albert Atserias, Moritz Müller, Automating resolution is NP-hard, *J. Assoc. Comput. Mach.* 67 (5) (2020) 31:1–31:17.
- [3] Paul Beame, Chris Beck, Russell Impagliazzo, Time-space trade-offs in resolution: superpolynomial lower bounds for superlinear space, *SIAM J. Comput.* 43 (4) (2016) 1612–1645.
- [4] Paul Beame, Henry A. Kautz, Ashish Sabharwal, Towards understanding and harnessing the potential of clause learning, *J. Artif. Intell. Res.* 22 (2004) 319–351.
- [5] Paul Beame, Toniann Pitassi, Propositional proof complexity: past, present and future, in: G. Paun, G. Rozenberg, A. Salomaa (Eds.), *Current Trends in Theoretical Computer Science Entering the 21st Century*, World Scientific, 2001, pp. 42–70, Earlier version appeared in *Computational Complexity Column*, Bulletin of the EATCS, 2000.
- [6] Maria Luisa Bonet, 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*, in: *Lecture Notes in Computer Science*, vol. 7317, 2012, pp. 45–57.
- [7] Maria Luisa Bonet, Samuel R. Buss, Jan Johannsen, Improved separations of regular resolution from clause learning proof systems, *J. Artif. Intell. Res.* 49 (2014) 669–703.
- [8] Sam Buss, Leszek Kołodziejczyk, Small stone in pool, *Log. Methods Comput. Sci.* 10 (2) (2014), Paper 2.
- [9] Sam Buss, Jakob Nordström, Proof complexity and SAT solving, in: *Handbook of Satisfiability*, second edition, in: *Frontiers in Artificial Intelligence and Applications*, vol. 336, IOS Press, 2021, pp. 233–350.
- [10] Samuel R. Buss, Lower bounds on Nullstellensatz proofs via designs, in: P. Beame, S. Buss (Eds.), *Proof Complexity and Feasible Arithmetics*, American Mathematical Society, 1998, pp. 59–71.
- [11] Samuel R. Buss, Jan Hoffmann, Jan Johannsen, Resolution trees with lemmas: resolution refinements that characterize DLL-algorithms with clause learning, *Log. Methods Comput. Sci.* 4 (13) (2008) 1–18.
- [12] Samuel R. Buss, Jan Krajíček, An application of Boolean complexity to separation problems in bounded arithmetic, *Proc. Lond. Math. Soc.* 69 (1994) 1–21.
- [13] Matt Clegg, Jeff Edmonds, 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*, 1996, pp. 174–183.
- [14] Susanna F. de Rezende, Noah Fleming, Duri A. Janett, Jakob Nordström, Shuo Pang, Truly supercritical trade-offs for resolution, cutting planes, monotone circuits, and Weisfeiler-Leman, *arXiv:2411.14267*, in: *STOC'25*, 2024, in press.
- [15] Noah Fleming, Toniann Pitassi, Robert Robere, Extremely deep proofs, in: *Proc. 13th Innovations in Theoretical Computer Science Conference, ITCS*, in: *LIPIcs*, vol. 215, 2022, pp. 70:1–70:23.
- [16] Jan Krajíček, Alan Skelley, Neil Thapen, NP search problems in low fragments of bounded arithmetic, *J. Symb. Log.* 72 (2) (2007) 649–672.
- [17] João P. Marques-Silva, Karem A. Sakallah, GRASP — a new search algorithm for satisfiability, *IEEE Trans. Comput.* 48 (5) (1999) 506–521.
- [18] Christos H. Papadimitriou, On the complexity of the parity argument and other inefficient proofs of existence, *J. Comput. Syst. Sci.* 48 (3) (1994) 498–532.
- [19] Knot Pipatsrisawat, Adnan Darwiche, On the power of clause-learning SAT solvers as resolution engines, *Artif. Intell.* 172 (2) (2011) 512–525.
- [20] Pavel Pudlák, Russell Impagliazzo, A lower bound for DLL algorithms for k-SAT (preliminary version), in: *Proc. 11th AM-SIAM Symposium on Discrete Algorithms, SODA*, 2000, pp. 128–136.
- [21] Ran Raz, Resolution lower bounds for the weak pigeonhole principle, in: *Proc. 34th ACM Symp. on Theory of Computing, STOC*, 2002, pp. 553–562.
- [22] Alexander A. Razborov, Resolution lower bounds for the weak pigeonhole principle, *Theor. Comput. Sci.* 303 (1) (2003) 233–243.
- [23] Alexander A. Razborov, A new kind of tradeoffs in propositional proof complexity, *J. ACM* 62 (3) (2016) 16:1–16:14.
- [24] Neil Thapen, A trade-off between length and width in resolution, *Theory Comput.* 12 (5) (2016) 1–14.
- [25] Alasdair Urquhart, A near-optimal separation of regular and general resolution, *SIAM J. Comput.* 40 (1) (2011) 107–121.
- [26] 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*, in: *Lecture Notes in Computer Science*, vol. 3835, Springer-Verlag, 2005, pp. 580–594.

Sponsor names

Do not correct this page. Please mark corrections to sponsor names and grant numbers in the main text.

Simons Foundation, *country* = United States, *grants* = 578919

Czech Academy of Sciences, *country* = Czechia, *grants* = RVO 67985840

GAČR, *country* = Czechia, *grants* = 23-04825S

Highlights

- We describe simple CNFs for which any small resolution refutation must have very large height.
- Small resolution refutations of these must thus be highly irregular.
- This makes them candidates to separate CDCL with restarts from CDCL without.
- We show that for a simple form of the formulas, there is no separation. It is open for the general formulas.