Exponential Lower Bounds for AC^0-Frege Imply Superpolynomial Frege Lower Bounds

Yuval Filmus[1,⋆], Toniann Pitassi[1,⋆], and Rahul Santhanam[2]

[1] Univeristy of Toronto
{yuvalf,toni}@cs.toronto.edu
[2] University of Edinburgh
rsanthan@inf.ed.ac.uk

Abstract. We give a general transformation which turns polynomial-size Frege proofs to subexponential-size AC^0-Frege proofs. This indicates that proving exponential lower bounds for AC^0-Frege is hard, since it is a longstanding open problem to prove super-polynomial lower bounds for Frege. Our construction is optimal for tree-like proofs.

As a consequence of our main result, we are able to shed some light on the question of weak automatizability for bounded-depth Frege systems. First, we present a simpler proof of the results of Bonet et al. [5] showing that under cryptographic assumptions, bounded-depth Frege proofs are not weakly automatizable. Secondly, we show that because our proof is more general, under the right cryptographic assumptions, it could resolve the weak automatizability question for lower depth Frege systems.

1 Introduction

The fundamental question in computational complexity is the P vs. NP question. Though we are very far from resolving this question, over the past few decades we have made substantial progress in understanding why certain approaches, for example diagonalization and the use of combinatorial or algebraic techniques to prove circuit lower bounds, are unlikely to work. Various barriers such as the relativization, natural proofs and algebrization barriers have been formulated to capture the limitations of known techniques, and in turn, this meta-level understanding of complexity lower bound problems has led to developments in areas such as low-level complexity and derandomization.

However, there are still approaches whose power is not well understood, such as those used in proof complexity. Proof complexity was introduced by Cook and Reckhow [8] as a framework within which to study the NP vs. coNP problem. Cook and Reckhow defined *propositional proof systems* in a very general way by insisting only that proofs be verifiable in polynomial time, and showed that the existence of a propositional proof system in which all tautologies have polynomial size proofs is equivalent to NP = coNP. They suggested a program to separate NP and coNP (and thereby P and NP) by showing superpolynomial proof size

⋆ Supported by NSERC.

lower bounds for explicit tautologies in progressively stronger proof systems. The hope was that techniques from logic and proof theory could be effective where techniques inspired by recursion theory or combinatorics are not. The fact that the very definition of the P vs. NP question involves the notion of "proof" in a fundamental way makes this hope somewhat plausible.

Indeed, over the past couple of decades, lower bounds have been shown for various natural proof systems [9,3]. However, lower bounds for natural systems such as Frege and Extended Frege still seem out of reach. We seem to have hit a "wall" with proof complexity lower bounds, just as with circuit complexity lower bounds.

To an extent, this reflects the fact that the techniques used for the currently strongest proof complexity lower bounds are adaptations of the techniques used in circuit complexity, and limitations of the circuit complexity techniques carry over to the versions used in proof complexity. There is an informal "mapping" from proof systems to complexity classes, where a proof system Q corresponds to the smallest complexity class C such that the lines of polynomial-sized proofs in Q are functions in C. In this way, Resolution maps to DNFs, Bounded-Depth Frege to non-uniform AC^0, Frege to non-uniform NC^1, and Extended Frege to SIZE(*poly*).

In the circuit complexity world, we have lower bounds for explicit functions against DNFs and non-uniform AC^0, but not against non-uniform NC^1 and SIZE(*poly*); correspondingly, in the proof complexity setting we have strong lower bounds for Resolution and fairly strong lower bounds for bounded-depth Frege, but no non-trivial lower bounds for Frege and Extended Frege.

The question arises whether this connection between circuit complexity and proof complexity is fundamental or not. No formal connection is known either way — we don't have theorems to the effect that circuit complexity lower bounds yield proof complexity lower bounds, nor any implications in the reverse direction. Moreover, barriers such as the natural proofs barrier don't seem to apply to proof complexity. This suggests that perhaps completely different techniques, say from proof theory or finite model theory, might help in showing proof complexity lower bounds. Is there a sense in which there are barriers to making progress in proof complexity? Formulating and understanding such barriers would not only guide us towards the "right" techniques, but might have collateral benefits as well, as in the circuit complexity setting.

In this paper, we shed some light on these questions. We draw a connection between two fundamental lower bound questions in proof complexity. The first question is to prove strong lower bounds for bounded-depth Frege. Superpolynomial lower bounds are known for this proof system, but there aren't any lower bounds known that are purely exponential, i.e., $2^{\Omega(n^c)}$ where the constant c doesn't depend on the depth of lines in the proof (the best known lower bound is $\Omega(2^{n^{5^{-d}}})$ [3]). The second question, which is perhaps the major open question in proof complexity, is to obtain superpolynomial lower bounds for Frege. This question is believed to be very hard — it is non-trivial even to think of plausible candidate tautologies for which superpolynomial lower bounds are believed to hold [4]. We show that progress on the first question would lead to progress on the second, by giving a general simulation of polynomial size Frege proofs by subexponential size bounded-depth Frege proofs. More precisely, we show that even a $2^{n^{\omega(1/d)}}$ proof size lower bound for proving CNF tautologies in depth d Frege would translate to a superpolynomial proof size lower bound for Frege.

The proof of this connection is inspired by a result in circuit complexity, further strengthening the "mapping" between proof complexity and circuit complexity. The circuit complexity result we draw inspiration from is that NC^1 can be simulated by bounded-depth circuits with sub-exponential size [1]. The standard proof of this goes via a divide-and-conquer technique. We use a similar technique in our context, however our task is made harder in a sense by the fact that we need to reason within bounded-depth Frege about equivalence of various alternative representations of a function. The technical heart of our proof involves such reasoning.

Our result is also relevant to algorithmic analysis, which is another major motivation for studying proof complexity. A propositional proof system can be thought of as a non-deterministic algorithm for deciding if a formula is a tautology or not. Proof systems such as bounded-depth Frege and Frege provide particularly simple and natural examples of such algorithms. Indeed, many of the algorithms and heuristics used in practice for solving SAT, such as DPLL and Clause Learning, arise from *determinizing* the non-deterministic algorithm corresponding to some natural proof system. Thus lower bounds for proof systems give us information on the performance of algorithms used in practice.

Algorithmic analysis would appear to be a simpler question than proving complexity lower bounds, since a complexity lower bound is a statement about *any* possible algorithm for a problem, while algorithmic analysis deals with specific algorithms. There are somewhat artificial algorithms such as Levin's optimal algorithm for SAT whose analysis is just as difficult as proving complexity lower bounds. However, one might expect that for more natural algorithms, such as those corresponding to natural propositional proof systems, this is not the case. Our current lack of progress in proving proof complexity lower bounds indicates that there might be barriers even in algorithmic analysis of natural algorithms. Our main result here can be interpreted as saying that the algorithmic analysis question for the algorithm corresponding to bounded-depth Frege is as hard as the question for the algorithm corresponding to Frege (which in some sense is a more sophisticated algorithm). In general, it would be useful to have a theory of algorithmic analysis which gives us information about the relative difficulty of analyzing various natural algorithms. We make a small step in this direction in the setting of non-deterministic algorithms for TAUT.

There are a couple of interesting byproducts of our main result. First, we are able to prove *tight* bounds for proving certain explicit tautologies in treelike bounded-depth Frege. Lower bounds for the tautologies we consider were already shown by Krajíček [10]. We give corresponding upper bounds as a corollary of our simulation of Frege by bounded-depth Frege.

Second, we address the question of *weak automatizability* for bounded-depth Frege systems. A proof system $\mathcal{P}$ is weakly automatizable if there is an algorithm that on input f and a number r in unary, can distinguish the case where f is not a tautology from the case where f has a $\mathcal{P}$-proof of size at most r. Despite considerable effort, the question of whether low depth proof systems are weakly automatizable is unresolved. Bonet, Domingo, Gavaldá, Maciel and Pitassi [5] show that depth k Frege systems are not weakly automatizable under a cryptographic assumption, but their result breaks down for small k (less than 6). We use our main result to re-derive their main theorem. Our proof is cleaner and simpler than theirs, and we show that it could potentially resolve the weak automatizability question for lower depth Frege systems than what is currently known.

1.1 Proof Overview

Suppose that P is a Frege proof of some formula f. We want to simulate P by a subexponential-size depth d Frege proof of f. The high-level idea behind the simulation is to replace every formula in the proof by its equivalent depth d (subexponential-size) *flattened* formula, and then to show that if C was derived by a rule from A and B, then the flattened version of C can be efficiently derived from the flattened versions of A and B.

We can assume without loss of generality that all formulas f in the proof are balanced (Reckhow's theorem). We first review the translation of a balanced formula f to its flattened form. We say that a formula has logical depth at most d if the depth of the binary tree representing the formula is at most d. Suppose that we want to replace f, of size n and logical depth $\log n$, by a depth 3 formula. The idea is to view f as consisting of two layers: the top layer is a formula, f_1, of height $(\log n)/2$, and the bottom layer consists of $2^{(\log n)/2} = \sqrt{n}$ subformulas, $g_1, \ldots, g_{\sqrt{n}}$, each of height $(\log n)/2$. Since f_1 has height $(\log n)/2$, it has at most $\sqrt{n}$ inputs, and thus can be written as either a CNF or a DNF formula (of its inputs) of size $\sqrt{n}2^{\sqrt{n}}$. Similarly, each formula in the bottom layer can be written as either a CNF or a DNF formula of size $\sqrt{n}2^{\sqrt{n}}$. Writing f_1 as a CNF formula, and writing all formulas g_j in the bottom layer as DNF formulas, we obtain a new formula for f of depth 3 and total size $O(n2^{2\sqrt{n}})$. (The depth is 3 because we can merge the middle two AND layers.) In a similar manner, we can replace any formula f, of size n and logical depth $\log n$, by a depth $d+2$ formula: Now we break up f into d equally-spaced layers, each of size $(\log n)/d$. Again, we write the formula at the top layer as a CNF formula, the formulas at the next layer as DNF formulas, and so on. This gives a formula of depth $2(d+1)$ and total size $O(n2^{dn^{1/d}})$, but since we alternated CNF/DNFs, we can collapse every other layer to obtain a new flattened formula of depth $2(d+1) - d = d+2$.

Now that we have flattened translations of each formula in P, it remains to fill in the proof, to show that the flattened versions can be derived from one another. In order to carry this out, we define a more general procedure for flattening a formula as follows. Let $\boldsymbol{d}$ be any *depth vector* – i.e., it is a sequence of increasing numbers, where each number in the sequence is between 1 and $\log n$. Then from a balanced formula f of size n and logical depth $\log n$, $\boldsymbol{d}$ defines a new flattened formula of depth $|\boldsymbol{d}|+2$: we break f up into $|\boldsymbol{d}|$ many levels, where now instead of the levels being equally spaced, the breakpoints are specified by $\boldsymbol{d}$. For example, if $\boldsymbol{d} = (4, 12)$ and f has depth 20, then the $\boldsymbol{d}$-flattened version of f will have 3 levels, the top level containing levels 1 through 3, the second level 4 through 11, and the third level 12 through 20. Our main lemma shows that for any balanced formula f and any two depth vectors $\boldsymbol{d}_1$, $\boldsymbol{d}_2$, there are efficient low-depth Frege proofs showing that the $\boldsymbol{d}_1$-flattened version of f is equivalent to the $\boldsymbol{d}_2$-flattened version of f. This main lemma will then allow us to prove that for any rule of our proof system, the flattened versions of the antecedent formulas derive the flattened version of the consequent formula.

2 Proof Systems

We will work with the propositional sequent calculus, PK. In the fundamental work of Cook and Reckhow [8], many reasonable formulations of Frege systems (including *all* PK-like systems) were studied and shown to be polynomially equivalent; we work with PK for convenience, but any other Frege system will do.

Each line in a PK proof is a *sequent* of the form $A_1, \ldots, A_k \longrightarrow B_1, \ldots, B_m$ where $\longrightarrow$ is a new symbol, and A_i, B_j are formulas. The intended meaning is that the conjunction of the A_i's implies the disjunction of the B_j's.

A PK *proof* of $\longrightarrow f$ is a sequence of sequents, such that each sequent is either an instance of the axiom $A \longrightarrow A$, or follows from previous sequents from one of the inference rules, and such that the final sequent is $\longrightarrow f$.

The rules of PK are of three types: (i) the structural rules, (ii) the logical rules, and (iii) the cut rule.

The structural rules are weakening, contraction and permutation.

The logical rules allow us to introduce each connective on both the left side and the right side.

The final rule is the cut rule, which allows us to derive $\Gamma \longrightarrow \Delta$ from $A, \Gamma \longrightarrow \Delta$ and $\Gamma \longrightarrow A, \Delta$. We call formula A the *cut formula*.

A full description of PK is found in the appendix.

The *size* of a PK proof is the sum of the sizes of all formulas occurring in the proof.

The *logical depth* of a formula φ, denoted by $\mathrm{ldp}(\varphi)$, is the depth of the formula when considered as a binary tree. For example, $(A \wedge B) \wedge C$ has logical depth 2. A formula whose logical depth is D has size at most $2^{D+1} - 1$, and can depend on at most 2^D variables.

The *depth* of a formula φ, denoted by $\mathrm{dp}(\varphi)$, is the maximum number of alternations between AND and OR connectives from root to leaf, not counting negations, plus one. For example, $(A \wedge B) \wedge C$ has depth 1, and the depth of a CNF or DNF formula is two.

We have given definitions of two different notions of depth. We will use logical depth to reason about formulas in Frege proofs, and depth to reason about formulas in bounded depth proofs.

A *cut-depth* k proof, also called an AC^0_k*-Frege* proof, is a PK proof where every *cut formula* in the proof has depth at most k (other formulas are allowed to have arbitrary depth). Note that in the literature, an AC^0_k-Frege proof is often defined to be a PK proof where *all* formulas have depth at most k. This definition is equivalent to ours if the proven formula has depth at most k.

A PK proof is *tree-like* if the underlying dag structure of the proof forms a tree, i.e. each sequent is used only once.

For technical reasons, we will need all the formulas in our proofs to be balanced. By the following result of Reckhow, this can be assumed without loss of generality.

Theorem 1 (Reckhow, [11, Lemma 4.4.14]). *If a formula of logical depth D has a PK proof of size s, then it has a PK proof of size $s^{O(1)}$ in which all formulas have logical depth $D + O(\log s)$. If the original proof is tree-like, then the new balanced proof is also tree-like.*

Definition 1. *A proof system S is automatizable if there exists an algorithm A such that for all unsatisfiable formulas f, $A(f)$ returns an S-proof of f, and the runtime of A on f is polynomial in the size of the smallest S-proof of f. S is weakly automatizable if there exists a proof system that polynomially simulates S and that is automatizable.*

3 Reducing Formula Depth

We reduce the depth of a formula using a divide-and-conquer technique. The idea is to decompose the formula into relatively small sub-trees, and replace each sub-tree by a CNF or DNF which is equivalent to the formula computed by the sub-tree.

Definition 2. *Let φ be an arbitrary formula depending on n variables. Denote by* $\mathrm{CNF}(\varphi)$ *(*$\mathrm{DNF}(\varphi)$*) some canonically chosen CNF (DNF) representing φ of size $O(n2^n)$. We require that* $\mathrm{CNF}(p \wedge q) = \mathrm{DNF}(p \wedge q) = p \wedge q$*, and similarly for $p \vee q$ and $\neg p$, when p and q are variables.*

We think of formulas as trees in which internal nodes are either binary (if the corresponding connective is $\wedge$ or $\vee$) or unary (when the connective is $\neg$), and leaves are labelled by variables. Each formula has an equivalent formula of the same size where negations only appear immediately above leaves, just by applying De Morgan's laws repeatedly to "move" negations down. We will call such formulas *quasi-monotone*, and will work with them throughout our simulation.

Definition 3. *A* quasi-monotone *formula is one in which negations only appear next to variables, and there are no double negations. Let φ be a quasi-monotone formula. Its* dual *form* $\mathrm{M}(\varphi)$ *is obtained from φ by switching $\wedge$ and $\vee$ and negating all literals, that is for each variable x switching x and $\neg x$; $M(\varphi)$ is logically equivalent to $\neg\varphi$.*

We define two *canonical flattened forms* in parallel.

Definition 4. *Let $\boldsymbol{d} = d_1, \ldots, d_k$ be a vector of increasing positive integers. The* conjunctive flattened form $\mathrm{C}(\varphi; \boldsymbol{d})$ *and* disjunctive flattened form $\mathrm{D}(\varphi; \boldsymbol{d})$ *of a formula φ are defined recursively as follows. If $k = 0$ (i.e., $\boldsymbol{d}$ is the empty vector) or $d_1 \geq \mathrm{ldp}(\varphi)$ then $\mathrm{C}(\varphi; \boldsymbol{d}) = \mathrm{CNF}(\varphi)$ and $\mathrm{D}(\varphi; \boldsymbol{d}) = \mathrm{DNF}(\varphi)$. Otherwise, let ψ be the formula obtained from φ by trimming the tree at depth d_1. The formula ψ depends on the variables of φ as well as on variables corresponding to subformulas of φ at depth d_1; we call these* true variables *and* subformula variables, *respectively. Let v_χ denote the subformula variable corresponding to the subformula χ.*

We explain how to calculate the conjunctive flattened form; the disjunctive flattened form is analogous. Start with $\mathrm{CNF}(\psi)$. Let $\boldsymbol{e} = d_2 - d_1, \ldots, d_k - d_1$. Replace each positive occurrence of a subformula variable v_χ in $\mathrm{CNF}(\psi)$ with $\mathrm{D}(\chi; \boldsymbol{e})$, and each negative occurrence with $\mathrm{M}(\mathrm{C}(\chi; \boldsymbol{e}))$. The result is $\mathrm{C}(\varphi)$.

The flattened forms are both shallow and not too large.

Definition 5. *Let φ be a formula and $\boldsymbol{d} = d_1, \ldots, d_k$ be a vector of increasing positive integers, such that $d_1 \leq \mathrm{ldp}(\varphi)$. Let $d_0 = 0$ and $d_{k+1} = \mathrm{ldp}(\varphi)$. The* extent *of φ with respect to $\boldsymbol{d}$ is*

$$\mathrm{ex}(\varphi; \boldsymbol{d}) = \max\{d_{i+1} - d_i : 0 \leq i \leq k\}.$$

Lemma 1. *Let φ be a formula and $\boldsymbol{d}$ a vector of length k and extent $x = \mathrm{ex}(\varphi; \boldsymbol{d})$. Then $\mathrm{C}(\varphi; \boldsymbol{d})$ and $\mathrm{D}(\varphi; \boldsymbol{d})$ are formulas of depth at most $k + 2$ and size $2^{O(k2^x)}$ equivalent to φ.*

4 Proof of Main Theorem

In this section, we will prove the following theorem.

Theorem 2. *Let φ be a formula provable in Frege in size s, satisfying $\mathrm{ldp}(\varphi) \leq C \log s$. For every $k \geq 1$ there is an AC^0_{k+2}-Frege proof of φ of size $2^{O(ks^{O(C/k)})}$. Furthermore, if the original proof is tree-like, so is the new one.*

Corollary 1. *Let φ be a formula of size s and logical depth at most $C \log s$. If φ has a Frege proof of size $O(s^c)$ then for every $k \geq 1$ there is an AC^0_{k+2}-Frege proof of φ of size $2^{O(cks^{O(C/k)})}$.*

We will first state some simple lemmas which will enable us to reason about flattened forms. The proofs of these lemmas appear in the Appendix.

Lemma 2. *Let $\Gamma \longrightarrow \Delta$ be a valid sequent of size m, in which n variables appear. The sequent is provable using a tree-like proof of size $O(m^2 n 2^n)$ which cuts only on variables.*

Our next lemma states that we can substitute formulas for variables to get a valid proof.

Lemma 3. *Let π be a proof of $\Gamma \longrightarrow \Delta$ of size s, and let x be a variable appearing in $\Gamma \longrightarrow \Delta$. If we substitute everywhere a formula φ of size m for x then we get a valid proof of size at most sm.*

The preceding lemma shows that we can lift a proof of a sequent by attaching stuff 'below'. The next lemma shows that we can also lift a proof by attaching stuff 'above'; this corresponds to deep inference.

Definition 6. *The double sequent $P \longleftrightarrow Q$ is the pair of sequents $P \longrightarrow Q$ and $Q \longrightarrow P$.*

Lemma 4. *Let $P \longrightarrow Q$ be a sequent of size m, and $\varphi(x)$ be a formula of size n in which the variable x appears only once (other variables may also appear). The double sequent $\varphi(x|P) \longleftrightarrow \varphi(x|Q)$ has a cut-free, tree-like proof from the double sequent $P \longleftrightarrow Q$ of size $O(n(m+n))$ (this means that each of $P \longrightarrow Q$ and $Q \longrightarrow P$ is used only once in the joint proof).*

We next state two easy lemmas on dualization.

Lemma 5. *Let φ be a quasi-monotone formula of size n. The double sequent $\mathrm{M}(\varphi) \longleftrightarrow \neg\varphi$ has a cut-free, tree-like proof of size $O(n^2)$.*

The second lemma allows us to lift an equivalence to its dualized version.

Lemma 6. *Let φ, ψ be quasi-monotone formulas. Suppose that the double sequent $\varphi \longleftrightarrow \psi$ has a proof of size s cutting on formulas of depth at most D. Then the double sequent $\mathrm{M}(\varphi) \longleftrightarrow \mathrm{M}(\psi)$ has a proof of size $O(s)$ cutting on formulas of depth at most D. Furthermore, if the original proof is tree-like then so is the new proof.*

We comment that the preceding lemma can be strengthened to produce cut-free proofs.

4.1 Moving Down the Depth Vector

In this section we show how to prove the equivalence of two flattened forms of the same formula which correspond to two different depth vectors.

Lemma 7. *Let φ be a formula of logical depth D, and δ a positive integer. Consider $\mathrm{CNF}(\varphi)$ and $\mathrm{C}(\varphi;\delta)$ as monotone formulas depending on literals $x, \bar{x}$; in other words, for each variable x, we replace $\neg x$ by $\bar{x}$. The double sequent $\mathrm{CNF}(\varphi) \longleftrightarrow \mathrm{C}(\varphi;\delta)$ has a tree-like proof of size $2^{O(2^D)}$ cutting only on literals.*

Lemma 8. *Let φ be a formula, $\boldsymbol{d} = d_1, \ldots, d_k$ a vector of increasing positive integers, and $\delta < d_1$ be a positive integer. The double sequent $\mathrm{C}(\varphi;\boldsymbol{d}) \longleftrightarrow \mathrm{C}(\varphi;\delta,\boldsymbol{d})$ has a tree-like proof of size $2^{O(k2^x)}$ cutting only on formulas of depth at most $k+1$, where $x = \mathrm{ex}(\varphi;\boldsymbol{d})$.*

Lemma 9. *Let φ be a formula, $\boldsymbol{d} = d_1, \dots, d_k$ a vector of increasing positive integers, and $d_i < \delta < d_{i+1}$, where $1 \le i \le k$. Define $\boldsymbol{e} = d_1, \dots, d_i, \delta, d_{i+1}, \dots, d_k$. The double sequent has a tree-like proof of size $2^{O(k2^x)}$ cutting only on formulas of depth at most $k+1$, where $x = \mathrm{ex}(\varphi; \boldsymbol{d})$.*

Lemma 10. *Let φ be a formula, and $\boldsymbol{d} = d_1, \dots, d_k$ be a vector of increasing positive integers. Define $\boldsymbol{e} = 1, d_1 + 1, \dots, d_k + 1$. The double sequent $\mathrm{C}(\varphi; \boldsymbol{d}) \longleftrightarrow \mathrm{C}(\varphi; \boldsymbol{e})$ has a tree-like proof of size $2^{O(k2^x)}$ cutting only on formulas of depth at most $k+3$, where $x = \max(\mathrm{ex}(\varphi; \boldsymbol{d}), \mathrm{ex}(\varphi; \boldsymbol{e}))$.*

The same methods used to prove Lemma 9 enable us to prove the following lemma.

Lemma 11. *Let φ be a formula, and $\boldsymbol{d} = d_1, \dots, d_k$ be a vector of increasing positive integers. The double sequent $\mathrm{C}(\varphi; \boldsymbol{d}) \longleftrightarrow \mathrm{D}(\varphi; \boldsymbol{d})$ has a tree-like proof of size $2^{O(k2^x)}$ cutting only on formulas of depth at most $k+2$, where $x = \mathrm{ex}(\varphi; \boldsymbol{d})$.*

4.2 Putting It Together

In this section we show how to transform a Frege proof to an AC^0-Frege proof. We begin by proving intensional comprehension.

Lemma 12. *Let φ, ψ be formulas, and $\boldsymbol{d}$ be a vector of increasing positive integers of length k. The double sequents*

$$\mathrm{C}(\varphi \land \psi; \boldsymbol{d}) \longleftrightarrow \mathrm{C}(\varphi; \boldsymbol{d}) \land \mathrm{C}(\psi; \boldsymbol{d}), \quad \mathrm{C}(\varphi \lor \psi; \boldsymbol{d}) \longleftrightarrow \mathrm{C}(\varphi; \boldsymbol{d}) \lor \mathrm{C}(\psi; \boldsymbol{d})$$

have tree-like proofs of size $2^{O(k2^x)}$ with cuts on formulas of depth at most $k+3$, where $x = \mathrm{ex}(\varphi \land \psi; \boldsymbol{d})$.

Lemma 13. *Let φ be a formula, and $\boldsymbol{d}$ be a vector of increasing positive integers of length k. The double sequent $\mathrm{C}(\neg\varphi; \boldsymbol{d}) \longleftrightarrow \neg\,\mathrm{C}(\varphi; \boldsymbol{d})$ has a tree-like proof of size $2^{O(k2^x)}$ with cuts on formulas of depth at most $k+3$, where $x = \mathrm{ex}(\neg\varphi; \boldsymbol{d})$.*

The preceding lemmas allow us to unroll flattened forms.

Lemma 14. *Let φ be a formula, and $\boldsymbol{d}$ be a vector of increasing positive integers of length k. The double sequent $\varphi \longleftrightarrow \mathrm{C}(\varphi; \boldsymbol{d})$ has a tree-like proof of size $2^{O(k2^x)}$ with cuts on formulas of depth at most $k+3$, where $x = \mathrm{ex}(\varphi; \boldsymbol{d})$.*

Proof. The proof is by structural induction. If φ is a literal then there is nothing to prove. If $\varphi = \neg\psi$, then use Lemma 13 to prove $\mathrm{C}(\varphi; \boldsymbol{d}) \longleftrightarrow \neg\,\mathrm{C}(\psi; \boldsymbol{d})$. The induction hypothesis gives us a proof of $\psi \longleftrightarrow \mathrm{C}(\psi; \boldsymbol{d})$; move both ψ and its flattened form to the other side using four $\neg$ introduction rules, and apply cut twice to prove the required double sequent.

If $\varphi = \psi \land \chi$ then start with proofs of the following sequents, obtained by Lemma 12 and the induction hypothesis:

$$\mathrm{C}(\varphi; \boldsymbol{d}) \longleftrightarrow \mathrm{C}(\psi; \boldsymbol{d}) \land \mathrm{C}(\chi; \boldsymbol{d}), \qquad \psi \longleftrightarrow \mathrm{C}(\psi; \boldsymbol{d}), \qquad \chi \longleftrightarrow \mathrm{C}(\chi; \boldsymbol{d}).$$

Now prove $\mathrm{C}(\psi;\boldsymbol{d}) \wedge \mathrm{C}(\chi;\boldsymbol{d}) \longleftrightarrow \psi \wedge \chi$ as follows:

$$\dfrac{\dfrac{\mathrm{C}(\psi;\boldsymbol{d}) \longrightarrow \psi}{\mathrm{C}(\psi;\boldsymbol{d}) \wedge \mathrm{C}(\chi;\boldsymbol{d}) \longrightarrow \psi}\,\wedge\mathsf{L} \qquad \dfrac{\mathrm{C}(\chi;\boldsymbol{d}) \longrightarrow \chi}{\mathrm{C}(\psi;\boldsymbol{d}) \wedge \mathrm{C}(\chi;\boldsymbol{d}) \longrightarrow \chi}\,\wedge\mathsf{L}}{\mathrm{C}(\psi;\boldsymbol{d}) \wedge \mathrm{C}(\chi;\boldsymbol{d}) \longrightarrow \psi \wedge \chi}\,\wedge\mathsf{R}$$

The other sequent is proved similarly. Complete the proof using the cut rule. The case $\varphi = \psi \vee \chi$ is similar.

The proof of the main theorem is now simple.

Lemma 15. *Let φ be a formula provable in Frege in size s using a proof with maximum logical depth D. For every k there is an AC^0_{k+2}-Frege proof of φ of size $s2^{O(k2^{D/k})}$. Furthermore, if the original proof is tree-like, so is the new one.*

Proof. Let $\boldsymbol{d} = \lceil D/k \rceil, 2\lceil D/k \rceil, \ldots, (k-1)\lceil D/k \rceil$. Note that the extent of each formula with respect to $\boldsymbol{d}$ is at most $x = \lceil D/k \rceil$. Take the original proof and replace each formula ψ by $\mathrm{C}(\psi;\boldsymbol{d})$. Each application of a rule is still valid, but the proof as a whole isn't valid since not all formulas are in flattened form. We address this issue by tampering with the introduction rules, as in the following example, corresponding to the right $\wedge$ introduction rule:

$$\dfrac{\dfrac{\varGamma \longrightarrow \varDelta, \mathrm{C}(\psi;\boldsymbol{d}) \qquad \varGamma \longrightarrow \varDelta, \mathrm{C}(\psi;\boldsymbol{d})}{\varGamma \longrightarrow \varDelta, \mathrm{C}(\psi;\boldsymbol{d}) \wedge \mathrm{C}(\chi;\boldsymbol{d})}\,\wedge\mathsf{R} \qquad \dfrac{}{\mathrm{C}(\psi;\boldsymbol{d}) \wedge \mathrm{C}(\chi;\boldsymbol{d}) \longrightarrow \mathrm{C}(\psi \wedge \chi;\boldsymbol{d})}\,\mathsf{Lem.}\ 12}{\varGamma \longrightarrow \varDelta, \mathrm{C}(\psi \wedge \chi;\boldsymbol{d})}\,\mathsf{Cut}$$

Applying the same transformation for all introduction rules, we are left with a valid proof of $\longrightarrow \mathrm{C}(\varphi;\boldsymbol{d})$, where each sequent is now replaced by sequents of total size $2^{O(k2^x)}$; the total size so far is $s2^{O(k2^x)}$. Lemma 14 proves $\mathrm{C}(\varphi;\boldsymbol{d}) \longrightarrow \varphi$, and the proof is complete by cutting on $\mathrm{C}(\varphi;\boldsymbol{d})$.

The lemmas we used employ cuts of depth at most $k+2$. All cuts in the original proof now cut flattened formulas, which are of depth at most $k+1$.

Proof (of Theorem 2). Reckhow's Theorem (Theorem 1) supplies us with an $\mathrm{AC}^0_{O(\log s)}$ proof of φ of size $s^{O(1)}$. The theorem now follows by substituting $D = C\log(s)$ in Lemma 15.

5 Applications and Consequences

5.1 Tightness of Our Simulation

We first address the tightness of our simulation. The analogous result for circuit complexity shows that any function computable by a polynomial-size formula can be computed by depth d circuits of size $\exp(n^{O(1/d)})$. This result is tight, since Håstad's theorem proves that the parity function on n Boolean variables requires AC^0_d circuits of size $\exp(n^{1/d})$.

Similarly we can show that our result is also tight. The following theorem states that there are formulas that have polynomial-size Frege proofs, but that require AC^0_d proofs of size exponential in $n^{1/d}$.

Theorem 3. *For every d there is a sequence of balanced formulas φ_n of depth $d+2$ provable in Frege by a tree-like proof of size s_n such that every tree-like AC^0_d proof of φ_n requires size $2^{s_n^{\Omega(1/d)}}$.*

Proof. The formula φ_n is PHP_n, the pigeonhole principle with $n+1$ pigeons and n holes, with each variable replaced by a Sipser function of depth d. Buss [7] has shown how to prove PHP_n using a Frege proof of size $n^{O(1)}$, which can be made tree-like by squaring its size. Substituting the Sipser functions, we obtain a Frege proof of size $n^{d+O(1)}$.

Conversely, Krajíček [10] gives a lower bound of $2^{n^{\Omega(1)}}$ for proving φ_n in tree-like AC^0_d.

Since the formulas φ_n are balanced, Theorem 2 applies, and with $k=d-2$, gives proofs essentially matching the lower bound.

The above result proves tightness for formulas of high depth. We conjecture that our simulation is also tight with respect to CNF formulas and general, dag-like proofs. The obvious formula for witnessing the lower bound is the pigeonhole principle itself. However, as an artifact of the switching lemma technique used to obtain depth d Frege lower bounds for the pigeonhole principle, the current best lower bound is exponential in $n^{1/2^d}$. It is a well-known open problem to improve the lower bound to $\exp(n^{1/d})$ for the pigeonhole principle, or for any other CNF formula. Such a result would show that our simulation is tight even for CNF formulas and arbitrary dag-like proofs.

5.2 Weak Automatizability

Using our theorem, we are able to show that bounded-depth Frege is not weakly automatizable, under an assumption about the hardness of factoring. While this result has already been known [5], we first show how to prove it as a simple corollary of our main theorem.

Theorem 4 ([6]). *Frege systems do not have feasible interpolation and are not weakly automatizable unless the Diffie Hellman problem is computable by polynomial size circuits.*

The Diffie Hellman problem is based on a prime number p, $|p|=n$. The input to the problem is a number g less than p, and numbers $g^a \pmod{p}$, $g^b \pmod{p}$, for some numbers $a,b \leq p$. The output should be $g^{ab} \pmod{p}$. The main lemma from [6] shows that a particular tautology, DH_p, stating that the Diffie Hellman function is well defined, has Frege proofs of size $O(|p|^c)$, where $c \leq 4$. Take DH_p where $|p| = (\log n)^q$ for some constant q. By our normal form theorem, this implies that DH_p has AC^0_k-Frege proofs of size $2^{O(k(\log n)^{cq/k})}$. Thus for $k > cq$, this is polynomial in n. Hence it follows that if AC^0_k-Frege is weakly automatizable (or has feasible interpolation), then the Diffie Hellman problem for $|p| = n' = (\log n)^{k/c}$ can be solved in time $n^{O(k)} = 2^{O(k \log n)} = \exp O(k(n')^{c/k})$.

Unfortunately, the quality of this negative result degrades for small k. Indeed despite considerable effort, it is unknown whether or not very low depth Frege systems (when k is less than 5) are weakly automatizable (the recent paper [2] reveals a connection between automatizability of AC^0_2-Frege with bottom fan-in 2 and feasibility of mean-payoff games). The main reason for this is that the Diffie Hellman function is not hard enough! Algorithms exist for computing discrete log over all finite fields, and hence for Diffie Hellman, that run in time exponential in $\sqrt{n}$. Moreover, the number field sieve is conjectured to solve discrete log (and thus Diffie Hellman) in time exponential in $\sqrt[3]{n}$. On the other hand, it seems entirely possible to come up with a different interpolant statement for another function that is much harder – truly exponential in n, and that still has efficient Frege proofs. Using our main theorem (which scales down *any* Frege proof), this would imply new negative results for weak automatizability and feasible interpolation for lower depth Frege systems than what is currently known.

References

1. Allender, E., Hellerstein, L., McCabe, P., Pitassi, T., Saks, M.: Minimizing disjunctive normal form formulas and AC^0 circuits given a truth table. SIAM Journal on Computing 38(1), 63–84 (2008)
2. Atserias, A., Maneva, E.: Mean-payoff games and propositional proofs. Inf. and Comp. 209(4), 664–691 (2011)
3. Beame, P.W., Impagliazzo, R., Krajíček, J., Pitassi, T., Pudlák, P., Woods, A.: Exponential lower bounds for the pigeonhole principle. In: Proceedings of the Twenty-Fourth Annual ACM Symposium on Computing, Victoria, B.C., Canada, pp. 200–220 (May 1992)
4. Bonet, M.L., Buss, S.R., Pitassi, T.: Are there hard examples for Frege systems? In: Feasible Mathematics II, pp. 30–56. Birkhäuser, Basel (1995)
5. Bonet, M.L., Domingo, C., Gavaldà, R., Maciel, A., Pitassi, T.: Non-automatizability of bounded-depth Frege proofs. Computational Complexity 13(1-2), 47–68 (2004)
6. Bonet, M.L., Pitassi, T., Raz, R.: On interpolation and automatization for Frege systems. SIAM Journal on Computing 29(6), 1939–1967 (2000)
7. Buss, S.R.: Polynomial size proofs of the pigeonhole principle. Journal of Symbolic Logic 57, 916–927 (1987)
8. Cook, S.A., Reckhow, R.A.: The relative efficiency of propositional proof systems. Journal of Symbolic Logic 44(1), 36–50 (1979)
9. Haken, A.: The intractability of resolution. Theoretical Computer Science 39, 297–305 (1985)
10. Krajíček, J.: Lower bounds to the size of constant-depth propositional proofs. Journal of Symbolic Logic 59(1), 73–86 (1994)
11. Krajíček, J.: Bounded arithmetic, propositional logic, and complexity theory. Cambridge University Press, New York (1995)