# Non-automatizability of bounded-depth Frege proofs

Maria Luisa Bonet* Carlos Domingo† Ricard Gavaldà‡
Alexis Maciel§ Toniann Pitassi¶

August 3, 1999

**Abstract**

In this paper, we show how to extend the argument due to Bonet, Pitassi and Raz to show that bounded-depth Frege proofs do not have feasible interpolation, assuming that factoring of Blum integers or computing the Diffie-Hellman function is sufficiently hard. It follows as a corollary that bounded-depth Frege is not automatizable; in other words, there is no deterministic polynomial-time algorithm that will output a short proof if one exists. A notable feature of our argument is its simplicity.

*Department of Software (LSI), Universitat Politècnica de Catalunya, Barcelona, Spain. bonet@lsi.upc.es. Partially supported by projects ESPRIT 20244 ALCOM-IT, TIC 97-1475-CE and CICYT TIC98-0410-C02-01.

†Department of Mathematical and Computing Science Tokyo Institute of Technology. Ookayama, Meguro-ku, Tokyo, Japan. carlos@is.titech.ac.jp. Partially supported by projects ESPRIT 20244 ALCOM-IT, TIC 97-1475-CE, KOALA DGES PB95-0787 and SGR CIRIT 1997SGR-00366.

‡Department of Software (LSI), Universitat Politècnica de Catalunya, Barcelona, Spain. gavalda@lsi.upc.es. Partially supported by projects ESPRIT 20244 ALCOM-IT, TIC 97-1475-CE, KOALA DGES PB95-0787 and SGR CIRIT 1997SGR-00366.

§Dept. of Math. and Computer Science, Clarkson University, Potsdam, NY 13699-5815, U.S.A. alexis@clarkson.edu.

¶Department of Computer Science, University of Arizona, Tucson, AZ 85721, U.S.A. toni@cs.arizona.edu. Research supported by NSF Grant CCR-9457782, US-Israel BSF Grant 95-00238, and Grant INT-9600919/ME-103 from NSF and MŠMT (Czech Republic).

# 1 Introduction

In the last years there has been a lot of interest in studying the complexity of propositional proof systems. The motivation is twofold. First, Cook and Reckhow [**?**] showed that the existence of a propositional proof system admitting polynomial-size proofs of all tautologies is equivalent to NP = Co-NP. This observation started a program of trying to prove superpolynomial lower bounds for increasingly more powerful proof systems. A second motivation for studying the complexity of proof systems comes from issues related to automated theorem proving. The question is: given a particular propositional proof system, are there efficient algorithms for finding the shortest proofs of a tautology in that system? Our results have to do with both motivations, and in what follows we will explain these relationships in more detail.

Consider first the issue of proving superpolynomial lower bounds for propositional proof systems. The interpolation method has been one of the most used and promising approaches. It is inspired by Craig's interpolation theorem for propositional logic which states that if $A(\vec{x},\vec{z}) \rightarrow B(\vec{y},\vec{z})$ is a tautology where $\vec{z}$ is a vector of shared variables, and $\vec{x}$ and $\vec{y}$ are vectors of separate variables for $A$ and $B$ respectively, then there is a formula $C(\vec{z})$ such that $A(\vec{x},\vec{z}) \rightarrow C(\vec{z})$ and $C(\vec{z}) \rightarrow B(\vec{y},\vec{z})$ are tautologies. There is a different formulation of this theorem in order to use it to prove lower bounds for unsatisfiable formulas as follows. Consider an unsatisfiable formula $A_0(\vec{x},\vec{z}) \wedge A_1(\vec{y},\vec{z})$, where $\vec{z}$ is a vector of shared variables, and $\vec{x}$ and $\vec{y}$ are vectors of separate variables for $A_0$ and $A_1$ respectively. Since the formula is unsatisfiable, it follows that for any truth assignment $\vec{\alpha}$ to $\vec{z}$, either $A_0(\vec{x},\vec{\alpha})$ is unsatisfiable or $A_1(\vec{y},\vec{\alpha})$ is unsatisfiable. An interpolation function associated with the formula is a Boolean function that takes such an assignment $\vec{\alpha}$ as input, and outputs 0 only if $A_0(\vec{x},\vec{\alpha})$ is unsatisfiable, and outputs 1 only if $A_1(\vec{y},\vec{\alpha})$ is unsatisfiable. We say that a proof system $S$ has the *feasible interpolation property* if whenever an unsatisfiable formula of the form $A_0(\vec{x},\vec{z}) \wedge A_1(\vec{y},\vec{z})$ has a polynomial-size refutation in $S$, then that formula has an interpolation function that can be computed by a polynomial-size Boolean circuit. Therefore, in a proof system with the feasible interpolation property, superpolynomial lower bounds on refutation size can be obtained by constructing an unsatisfiable formula of the form $A_0(\vec{x},\vec{z}) \wedge A_1(\vec{y},\vec{z})$ with interpolation function $F$ and proving that: (i) $F$ cannot be computed by polynomial-size circuits, and (ii) the formula has a polynomial-size refutation in $S$.

Unconditional lower bounds for proof systems have been obtained by considering a *monotone* variant of the feasible interpolation property. In the monotone case, one considers statements $A_0(\vec{x},\vec{z}) \wedge A_1(\vec{y},vecz)$ where $\vec{z}$ occurs

only positively in $A_0$. In this case, the interpolation function will be monotone, and we are interested in whether or not it can be computed by monotone polynomial-size circuits. For example, we can define $A_0(x,z)$ to say "the graph $z$ has a clique $x$ of size $k$" and $A_1(y,z)$ to say "the graph $z$ has a $k-1$-coloring $y$". It is easily seen that associated monotone interpolation function must distinguish between graphs containing a $k$-clique and no other edges, and graphs with $k-1$-cocliques, and for appropriately chosen $k$, this is known to require exponential-size monotone circuits [?]. Thus, one can obtain superpolynomial lower bounds for any proof system satisfying: (i) the above statement has an efficient refutation, and (ii) the proof system has the feasible monotone interpolation property.

In the last few years, the interpolation method has been used to prove many lower bounds. In particular, lower bounds have been shown for each of the following systems: Resolution and Cutting Planes [?, ?, ?, ?], generalizations of Cutting Planes [?, ?, ?], relativized bounded arithmetic [?], Hilbert's Nullstellensatz [?], the polynomial calculus [?], and the Lovasz-Schriver proof system [?].

On the other hand, in a separate sequence of papers beginning with a key idea due to Krajíček and Pudlák [?, ?], it has been shown that under sufficiently strong cryptographic assumptions, many stronger proof systems do not have feasible interpolation. The main ideas are as follows. Suppose that $H$ is a permutation that is generally believed to be one-way. Formulate $A_0(\vec{x},\vec{z})$ as saying "$H(x)=z$ and the last bit of $x$ is 0" and $A_1(y,z)$ as saying "$H(y)=z$ and the last bit of $y$ is 1". Since $H$ is injective, $A_0 \wedge A_1$ is a contradiction. If the proof system can have a short refutation of $A_0 \wedge A_1$, then the proof system does not have the feasible interpolation property, unless $H$ is not a one-way permutation. Using a more general reformulation of the ideas just sketched, it has been proved that the following proof systems do not have feasible interpolation, under commonly accepted cryptographic assumptions: Extended Frege, Frege and even $\mathrm{TC}^0$-Frege systems. These negative results are important not only as a guide for searching for lower bound techniques, but also because they imply that the proof system in question cannot be automatized. This connection was first made explicit by [?] and takes us back to the second motivation for studying propositional proof systems.

A proof system $S$ is *automatizable* if there exists a deterministic procedure $D$ that takes as input a formula $f$ and returns an $S$-refutation of $f$ (if one exists) in time polynomial in the size of the shortest $S$-refutation of $f$. Automatizability is a crucial concept for automated theorem proving: in proof complexity we are mostly interested in the length of the shortest proof, whereas in theorem proving it is also essential to be able to find the proof. Bonet, Pitassi

and Raz [?] show that if $S$ does not have feasible interpolation, then $S$ is not automatizable. Thus, feasible interpolation is a simple measure that formalizes the complexity/search tradeoff: the existence of feasible interpolation implies superpolynomial lower bounds (sometimes modulo complexity assumptions), whereas the nonexistence of feasible interpolation implies that the proof system cannot be automatized.

In this paper, we use and extend the ideas in [?] to show that bounded-depth Frege systems do not have feasible interpolation unless the Diffie-Hellman function can be computed by circuits of size $2^{n^\epsilon}$ for arbitrarily small $\epsilon > 0$. Note that our assumption is stronger than that of [?] who only needed to assume that the Diffie-Hellman function cannot be computed by polynomial-size circuits. Also note that computing the Diffie-Hellman function is at least as hard as factoring Blum integers [?] . (See also [?, ?].)

Our paper nearly completes the picture of which commonly studied proof system do and do not have the feasible interpolation property. Resolution is the simplest and most well-studied propositional proof system. In a Resolution proof, one begins with an unsatisfiable formula in conjunctive normal form and allowable lines in the refutation are disjunctions of literals. Resolution has been known to have feasible interpolation. Bounded-depth Frege systems, a subsystem of Frege systems, is a more powerful proof system than Resolution, where one can manipulate formulas of bounded-depth. In a bounded-depth Frege system, one can prove the weak pigeonhole principle in quasipolynomial-size, whereas Resolution proofs have exponential size. As a result of this paper, we now know that bounded-depth Frege systems as well as any proof system that can polynomially simulate bounded-depth Frege systems (for example, Frege systems, Extended Frege systems, Substitution Frege), do not have feasible interpolation (under cryptographic assumptions).

The basic idea behind the result of [?] is as follows. They construct a $\mathrm{TC}^0$-Frege formula $\mathrm{DH}_n$ based on the Diffie-Hellman function. The size of the formula is polynomial in $n$, the length of the numbers involved. The bulk of the argument is to show that there exists a polynomial-size $\mathrm{TC}^0$-Frege refutation of $\mathrm{DH}_n$. On the other hand, an interpolation function for $\mathrm{DH}_n$ computes one bit of the secret key exchanged by the Diffie-Hellman procedure. Thus, if $\mathrm{TC}^0$-Frege admits feasible interpolation, then the secret key exchanged by the Diffie-Hellman procedure can be broken using polynomial-size circuits and hence the Diffie-Hellman cryptographic scheme is not secure.

In the present paper, we will scale down the above idea from $n$ to polylog $n$. Consider $\mathrm{DH}_m$ where $m = \text{polylog}\, n$. By directly applying the main theorem of [?], $\mathrm{DH}_m$ has a $\mathrm{TC}^0$-Frege refutation of size polynomial in $m$. We will show how to simulate this refutation with an $\mathrm{AC}^0$-Frege refutation of size polynomial

in $n$. More generally, we will show that any $\mathrm{TC}^0$-Frege proof of size polynomial in $n$ in which all the threshold and parity connectives have fan-in polylog $n$ can be simulated by an $\mathrm{AC}^0$-Frege proof of size polynomial in $n$. Now if $\mathrm{AC}^0$-Frege admits feasible interpolation, then the secret key exchanged by the Diffie-Hellman procedure can be broken using circuits of subexponential size and hence the Diffie-Hellman cryptographic scheme is not secure.

The idea of scaling the input from $n$ to polylogarithmic in $n$ in order to be able to represent a cryptosystem in $ACzero$ instead of $TCzero$ to obtain certain results is not new. Kharitonov in [**?**] improved previously known negative results for the learnability of boolean circuit classes from $\mathrm{TC}^0$ to $\mathrm{AC}^0$ in a very similar way. However, it is important to notice that, although there is a parallelism between the negative results for learning and the negative results for feasible interpolation, the proofs in the latter case are more involved.

The paper is organized as follows. In Section 2, we define the $\mathrm{AC}^0$ and $\mathrm{TC}^0$-Frege systems. In Section 3, we define some $\mathrm{AC}^0$ formulas used in the simulation. In Section 4, we prove some preliminary lemmas. In Section 5, we show how to simulate the restricted $\mathrm{TC}^0$-Frege proofs mentioned in the previous paragraph. In Section 6, we prove our main result.

# 2 Background on the $\mathrm{TC}^0$-Frege non-interpolation result

We will outline here the proof of the result that $\mathrm{TC}^0$-Frege doesn't have feasible interpolation, and therefore is not automatizable, from [**?**].

The formulas $A_0, A_1$ are based on the Diffie-Hellman secret key exchange scheme [**?**]. The propositional statement $DH$ will be

$$DH_n = A_0(P, g, X, Y, a, b) \wedge A_1(P, g, X, Y, c, d).$$

The common variables are the integers $X, Y$, and $P$ and $g$. $P$ represents a number (not necessarily a prime) of length $n$, and $g$ an element of the group $Z_P^*$ (we will see below that actually the common variables are more). The private variables for $A_0$ are integers $a, b$, and the private variables for $A_1$ are integers $c, d$.

Informally, $A_0(P, g, X, Y, a, b)$ says that $g^a \bmod P = X$, $g^b \bmod P = Y$, and that $g^{ab} \bmod P$ is even. Similarly, $A_1(P, g, X, Y, c, d)$ says that $g^c \bmod P = X$, $g^d mod P = Y$, and $g^{cd} \bmod P$ is odd. The formula $DH_n$ is stated for the least significant bit, but the argument works for any bit. The statement

$A_0 \wedge A_1$ is unsatisfiable since (informally) if $A_0, A_1$ are both true we have

$$g^{ab} \bmod P = (g^a \bmod P)^b \bmod P = X^b \bmod P =$$

$$(g^c \bmod P)^b \bmod P = g^{bc} \bmod P = (g^b \bmod P)^c \bmod P =$$

$$Y^c \bmod P = (g^d \bmod P)^c \bmod P = g^{cd} \bmod P.$$

An interpolant function computes the least significant bit of the secret key exchanged by the Diffie-Hellman procedure. Since the argument works for any bit, if $TC^0$-Frege admits feasible interpolation, then all bits of the secret key exchanged by the Diffie-Hellman procedure can be broken using polynomial size circuits, and hence the Diffie-Hellman cryptographic scheme is not secure. It was proven that for $P = p_1 \cdot p_2$, where $p_1, p_2$ are both primes such that $p_1 \bmod 4 = p_2 \bmod 4 = 3$ (i.e., $P$ is Blum integer), breaking the Diffie-Hellman cryptographic scheme is harder than factoring $P$ [**?**] (see also [**?**, **?**]). Therefore if $TC^0$-Frege admits feasible interpolation, this also implies the existence of polynomial size circuits for factoring Blum integers.

We will review briefly how $DH_n$ was formalized in $TC^0$-Frege and how a small size $TC^0$-Frege proof of $DH_n$ was produced. Let us first mention that $TC^0$ circuits can compute in polynomial size the following arithmetical functions:

- addition and subtraction
- iterated addition and product
- modular arithmetic
- iterated producto of $n$ numbers each of length $n$.

The bulk of [**?**] involves describing $TC^0$-Frege formulas defining the previously stated arithmetical functions, and then and proving many basic properties of these functions with small $TC^0$-Frege proofs.

# 3 $\mathrm{AC}^0$ and $\mathrm{TC}^0$-Frege systems

We will work with the specific bounded-depth threshold logic system $\mathrm{TC}^0$-Frege defined in [**?**] and also used in [**?**]. This system is a sequent-calculus logical system where formulas are built up using the connectives $\vee$, $\wedge$, $\mathrm{Th}_k$, $\neg$, and $\oplus_b$. $\mathrm{Th}_k(x)$ is true if and only if the number of 1's in $x$ is at least $k$, and $\oplus_b(x)$ is true if and only if the number of 1's in $x$ is equal to $b \bmod 2$.

**Definition 1** *Formulas are built up using the connectives* $\wedge$, $\vee$, $\mathrm{Th}_k$, $\oplus_1$, $\oplus_0$, $\neg$. *All connectives are assumed to have unbounded fan-in.* $\mathrm{Th}_k(A_1, \ldots, A_n)$ *is interpreted to be true if and only if the number of true* $A_i$*'s is at least* $k$*;* $\oplus_b(A_1, \ldots, A_n)$ *is interpreted to be true if and only if the number of true* $A_i$*'s is equal to* $b \bmod 2$.

The formula $\wedge(A_1, \ldots, A_n)$ denotes the logical AND of the multi-set consisting of $A_1, \ldots A_n$, and similarly for $\vee$, $\oplus_b$ and $\mathrm{Th}_k$. Thus commutativity of the connectives is implicit. Our proof system operates on sequents which are sets of formulas of the form $A_1, \ldots, A_p \rightarrow B_1, \ldots, B_q$. The intended meaning is that the conjunction of the $A_i$'s implies the disjunction of the $B_j$'s. A proof of a sequent $S$ in our logic system is a sequence of sequents, $S_1, \ldots, S_r$, such that each sequent $S_i$ is either an initial sequent, or follows from previous sequents by one of the rules of inference, and the final sequent, $S_r$, is $S$.

The *initial sequents* are of the form: (1) $A \rightarrow A$ where $A$ is any formula; (2) $\rightarrow \wedge()$; $\vee() \rightarrow$; (3) $\oplus_1() \rightarrow$; $\rightarrow \oplus_0()$; and (4) $\mathrm{Th}_k() \rightarrow$ for $k \geq 1$; $\rightarrow \mathrm{Th}_0(A_1, \ldots, A_n)$ for $n \geq 0$. The rules of inference are as follows. Note that the logical rules are defined for $n \geq 1$ and $k \geq 1$. First we have simple structural rules such as weakening (formulas can always be added to the left or to the right), contraction (two copies of the same formula can be replaced by one), and permutation (formulas in a sequent can be reordered). The remaining rules are the cut rule, and logical rules which allow us to introduce each connective on both the left side and the right side. The cut rule allows the derivation of $\Gamma, \Gamma' \rightarrow \Delta, \Delta'$ from $\Gamma, A \rightarrow \Delta$, and $\Gamma' \rightarrow A, \Delta'$.

The logical rules are as follows.

1. (Negation-left) From $\Gamma \rightarrow A, \Delta$, we can derive $\neg A, \Gamma \rightarrow \Delta$.
2. (Negation-right) From $A, \Gamma \rightarrow \Delta$, derive $\Gamma \rightarrow \neg A, \Delta$.
3. (And-left) From $A_1, \wedge(A_2, \ldots, A_n), \Gamma \rightarrow \Delta$ derive $\wedge(A_1, \ldots, A_n), \Gamma \rightarrow \Delta$.
4. (And-right) From $\Gamma \rightarrow A_1, \Delta$ and $\Gamma \rightarrow \wedge(A_2, \ldots, A_n), \Delta$ derive $\Gamma \rightarrow \wedge(A_1, \ldots, A_n), \Delta$
5. (Or-left) From $A_1, \Gamma \rightarrow \Delta$ and $\vee(A_2, \ldots, A_n), \Gamma \rightarrow \Delta$ derive $\vee(A_1, \ldots, A_n), \Gamma \rightarrow \Delta$
6. (Or-right) From $\Gamma \rightarrow A_1, \vee(A_2, \ldots, A_n), \Delta$ derive $\Gamma \rightarrow \vee(A_1, \ldots, A_n), \Delta$.
7. (Mod-left) From $A_1, \oplus_{b-1}(A_2, \ldots, A_n), \Gamma \rightarrow \Delta$ and $\oplus_b(A_2, \ldots, A_n), \Gamma \rightarrow A_1, \Delta$ derive $\oplus_b(A_1, \ldots, A_n),\ \Gamma \rightarrow \Delta$.

8. (Mod-right) From $A_1, \Gamma \to \oplus_{b-1}(A_2, \ldots, A_n), \Delta$ and $\Gamma \to A_1, \oplus_b(A_2, \ldots, A_n), \Delta$ derive $\Gamma \to \oplus_b(A_1, \ldots, A_n), \Delta$.

9. (Threshold-left) From $\mathrm{Th}_k(A_2, \ldots, A_n), \Gamma \to \Delta$ and $A_1, \mathrm{Th}_{k-1}(A_2, \ldots, A_n), \Gamma \to \Delta$ derive $\mathrm{Th}_k(A_1, \ldots, A_n), \Gamma \to \Delta$.

10. (Threshold-right) From $\Gamma \to A_1, \mathrm{Th}_k(A_2, \ldots, A_n), \Delta$ and $\Gamma \to \mathrm{Th}_{k-1}(A_2, \ldots, A_n), \Delta$ derive $\Gamma \to \mathrm{Th}_k(A_1, \ldots, A_n), \Delta$.

The *size* of a proof is the total size of all the formulas that occur in the proof. The *depth* of a proof is the maximum depth of all the formulas that occur in the proof.

A family of sequents $(\Gamma_1 \to \Delta_1), (\Gamma_2 \to \Delta_2), (\Gamma_3 \to \Delta_3), \ldots$ has $\mathrm{TC}^0$-Frege proofs if each sequent has a bounded-depth proof of size polynomial in the size of the sequent. More precisely,

**Definition 2** *Let $F = \{(\Gamma_n \to \Delta_n) : n \in N\}$ be a family of sequents. Then $\{R_n : n \in \mathbf{N}\}$ is a family of $\mathrm{TC}^0$-Frege proofs for $F$ if there exist constants $c$ and $d$ such that the following conditions hold: (1) Each $R_n$ is a valid proof of $(\Gamma_n \to \Delta_n)$ in our system; (2) For all $i$, the depth of $R_n$ is at most $d$; and (3) For all $n$, the size of $R_n$ is at most $(\mathrm{size}(\Gamma_n \to \Delta_n))^c$.*

We say that a formula $f$ can be *arranged into $d$ levels* if the connectives of $f$ can be arranged into $d$ groups $L_1, \ldots, L_d$ called levels such that all the inputs of every connective at some level are either propositional variables or connectives from the previous levels. Note that $f$ can be arranged into $d$ levels if and only if $f$ has depth at most $d$. Moreover, if $f$ has depth less than $d$, then some of the levels may be empty.

**Definition 3** *The $\mathrm{AC}^0$-Frege system is a restriction of the $\mathrm{TC}^0$-Frege system, where we omit the parity and threshold connectives and the associated rules.*

In the following sections, we will use the symbols 0 and 1 in our formulas. These will simply stand for the formulas $x \wedge \neg x$ and $x \vee \neg x$, respectively. Thus the sequents $0 \to$ and $\to 1$ have constant-size $\mathrm{AC}^0$-Frege proofs.

# 4 Notation and $\mathrm{AC}^0$ counting formulas

In this section we will describe some of the $\mathrm{AC}^0$ formulas that we will be using. Recall that our goal is to show that $\mathrm{TC}^0$-Frege proofs of size polynomial in $n$ in which all the threshold and parity connectives have fan-in polylog $n$ can be

simulated by $AC^0$-Frege proofs of size polynomial in $n$. To this end, we will define $AC^0$ circuits of size polynomial in $n$ that can simulate threshold and parity gates of fan-in polylog $n$.

We will first show how to add polylog $n$ many bits using $AC^0$ circuits of size polynomial in $n$. The general idea is as follows. Suppose that the original input bits are $x_1, \ldots, x_m$, where $m = (\log n)^k$ for some $k$. We will sum these numbers in a divide and conquer fashion, by dividing these inputs into $(\log n)^{1/2}$ consecutive groups, where each group will have size $(\log n)^{k-1/2}$. After adding the numbers in each group (recursively), we will have $(\log n)^{1/2}$ numbers, each of length $(k-1/2)\log\log n$. For the final step, we notice that the total number of bits is less than $\log n$, and thus these $(\log n)^{1/2}$ numbers can be added using a DNF formula of size at most $n$. To summarize, the $AC^0$ circuit to add $(\log n)^k$ 1-bit numbers will be composed of $2k$ levels. The input level (level $2k$) will consist of $(\log n)^{k-1/2}$ "truth table" subcircuits, $TT^1$, where each truth-table subcircuit will take $(\log n)^{1/2}$ numbers, each of length 1, and output their sum. Finally the output level (level one), will consist of a single truth-table subcircuit, $TT^{(k-1/2)\log\log n}$, which will again take $(\log n)^{1/2}$ numbers, each of length $(k-1/2)\log\log n$ and output their sum.

We proceed more carefully below. We define five types of $AC^0$ circuits as follows.

1. $TT^j$: this will be a depth 2 circuit that takes as input $(\log n)^{1/2}$ numbers, each of length $j$ and outputs their sum. We will only use $TT^j$ for $j = O(\log\log n)$, thus these circuits take less than $\log n$ inputs, and can therefore be defined by the obvious DNF formulas. (TT thus stands for truth-table definition.) Note that if $j = k\log\log n$, then the number of output bits of $TT^j$ will be $(k+1/2)\log\log n$. The formula $TT^j_l$ represents the $l^{th}$ output bit of $TT^j$.

2. $+^j$: This circuit takes two numbers, each $j$-bits long, and outputs their sum. Since we will use this circuit only for $j = O(\log\log n)$, again the total number of bits is much less than $\log n$, so we will use the obvious depth-2 truth-table circuit. Note that the number of output bits of $+^j$ will be $j+1$.

3. $GE^j$: This is a depth-2 formula that takes two $j$-bit numbers $x$ and $y$ as input and outputs 1 if and only if $x$ is greater than or equal to $y$. We will be using $GE^j$ only for $j = O(\log\log n)$, so again this circuit will be the obvious depth-2 truth-table formula.

4. $EQUIV^j$: This is a depth-2 formula that takes two $j$-bit numbers $x$ and $y$ as input and outputs 1 if and only if $x$ is congruent to $y$ modulo 2. We

will be using $\mathrm{EQUIV}^j$ only for $j = O(\log\log n)$, so again this circuit will be the obvious depth-2 truth-table formula.

5. $\mathrm{SUM}^{j,i}$: This circuit takes as input $i$ numbers, each $j$ bits long and outputs their sum. The circuit will be defined inductively using the TT subcircuits repeatedly. First, $\mathrm{SUM}^{j,0}() = 0$ and $\mathrm{SUM}^{j,1}(x_1) = x_1$. Next, consider $\mathrm{SUM}^{j,i}(x_1, \ldots, x_i)$ for $i > 1$. There are two cases, depending on whether or not $i$ is a power of $(\log n)^{1/2}$. First, if $i$ is not a power of $(\log n)^{1/2}$, then $\mathrm{SUM}^{j,i}(x_1, \ldots, x_i)$ is equal to $\mathrm{SUM}^{j,i}(x_1, \ldots, x_i, 0, \ldots, 0)$, where we pad with the minimum number of zeroes such that the total number of inputs is a power of $(\log n)^{1/2}$. In the second case, assume that $i$ is a power of $(\log n)^{1/2}$, and specifically let $i = (\log n)^k$. The idea is that $\mathrm{SUM}^{j,i}(x_1, \ldots, x_i)$ will be a full tree consisting of $2k$ levels of TT's. We define $\mathrm{SUM}^{j,i}$ as follows:

$$\mathrm{SUM}^{j,(\log n)^k}(x_1, \ldots, x_{(\log n)^k}) = \mathrm{TT}^{j+(k-1/2)\log\log n}(A_1, \ldots, A_{(\log n)^{1/2}})$$

where $A_r = \mathrm{SUM}^{j,(\log n)^{k-1/2}}(x_{m_{r-1}+1}, \ldots, x_{m_r})$ and $m_t = t(\log n)^{k-1/2}$.

6. $\mathrm{TH}^i_k(x_1, \ldots, x_i)$: This is a constant-depth formula that takes $i$ one-bit inputs, and outputs 1 if and only if the number of 1's is $k$ or greater. It is defined to be equal to $\mathrm{GE}^{\log i}(\mathrm{SUM}^{1,i}(x_1, \ldots, x_i), k)$. It is important to note that in simulating the original threshold gate, $\mathrm{Th}_k$, we are going from an unordered list of the variables to an ordered list of the variables. That is, in our formula for $\mathrm{TH}^i_k$, the order of the variables matters. Even though commutativity of the underlying variables was implicit in $\mathrm{Th}_k$, we will need to show that permutation of $\mathrm{TH}_k$ can be simulated by our formulas.

7. $\mathrm{PARITY}^i_b(x_1, \ldots, x_i)$: This is a constant-depth formula that takes $i$ one-bit inputs, and outputs 1 if and only if the number of 1's is congruent to $b$ modulo 2. It is defined to be equal to $\mathrm{EQUIV}^{\log i}(\mathrm{SUM}^{1,i}(x_1, \ldots, x_i), b)$. Again, we will need to show that permutation of $\mathrm{PARITY}_b$ can be simulated by our formulas.

To simplify notation, we will usually omit the superscripts on the above $\mathrm{AC}^0$ formulas. (They can be figured out from context.) It will be helpful to keep in mind that the length of all intermediate numbers will be at most $O(\log\log n)$ (i.e., $j = O(\log\log n)$.)

Also, sometimes we will use the notation $f = g$, where $f$ and $g$ are circuits, each with $j$ outputs. For example, $\mathrm{SUM}(A_1, A_2, \ldots, A_m) =$

$\mathrm{SUM}(A_2, A_1, \ldots, A_m)$. This notation is shorthand for the sequent $\rightarrow \bigwedge_{i=1}^{j}((\neg f_i \vee g_i) \wedge (\neg g_i \vee f_i))$. However, when $f = g$ occurs in a sequent, then it represents the *formula* $\bigwedge_{i=1}^{j}((\neg f_i \vee g_i) \wedge (\neg g_i \vee f_i))$. Lastly, in general, we will write the above formulas in prefix notation (i.e., $\mathrm{GE}(x, y)$), but for the $+$ formulas we will usually use infix notation (i.e., $x + y$).

# 5 Preliminaries

The lemmas of this section will greatly simplify the arguments in the rest of the article. Let $F(x)$ be a formula depending on propositional variable $x$. $F$ may also depend on other variables; the notation $F(x)$ means that only $x$ is relevant in the context. Given another formula $A$, $F(A)$ will denote the formula obtained by replacing every occurrence of $x$ by $A$. A derivation of a sequent $S$ from $S_1, \ldots, S_p$ is a proof of $S$ that uses the sequents $S_1, \ldots, S_p$ as additional initial sequents.

Lemma 4 can be proved by induction on the structure of the formula $F$. Lemma 5 then follows from Lemma 4, and Lemma 6, from Lemma 5.

**Lemma 4** *In* $\mathrm{AC}^0$*-Frege, for every formula* $A$*,* $B$ *and* $F(x)$*, and for every sequence of formulas* $\Gamma$ *and* $\Delta$*, the following sequents can be derived in size polynomial in the size of* $A$*,* $B$*,* $F(x)$*,* $\Gamma$ *and* $\Delta$*:*

1. $(\Gamma \rightarrow F(A), \Delta)$ *from* $(\Gamma \rightarrow F(B), \Delta)$, $(\Gamma, B \rightarrow A, \Delta)$ *and* $(\Gamma, A \rightarrow B, \Delta)$.
2. $(\Gamma, F(A) \rightarrow \Delta)$ *from* $(\Gamma, F(B) \rightarrow \Delta)$, $(\Gamma, B \rightarrow A, \Delta)$ *and* $(\Gamma, A \rightarrow B, \Delta)$.

**Lemma 5** *In* $\mathrm{AC}^0$*-Frege, for every formula* $A$ *and* $F(x)$*, the following sequents can be proved in size polynomial in the size of* $A$ *and* $F(x)$*:*

1. $F(0) \rightarrow A, F(A)$
2. $F(1), A \rightarrow F(A)$
3. $F(A), A \rightarrow F(1)$
4. $F(A) \rightarrow A, F(0)$

**Lemma 6** *In* $\mathrm{AC}^0$*-Frege, for every formula* $A$ *and* $F(x)$*, the following sequents can be derived in size polynomial in the size of* $A$ *and* $F(x)$*:*

1. $\rightarrow F(A)$ *from* $\rightarrow F(0)$ *and* $\rightarrow F(1)$.

2. $F(A) \rightarrow$ *from* $F(0) \rightarrow$ *and* $F(1) \rightarrow$.

**Lemma 7** *In* $\mathrm{AC}^0$*-Frege, for every formula* $F(x_1, \ldots, x_n)$ *and for every sequence of formulas* $A_1, \ldots, A_n$*, if* $\rightarrow F(A_1, \ldots, A_n)$ *is a tautology, then* $\rightarrow F(A_1, \ldots, A_n)$ *can be derived from sequents of the form* $F(B_1, \ldots, B_n) \rightarrow F(B_{\pi(1)}, \ldots, B_{\pi(n)})$ *where* $\pi$ *is a permutation. The size of the derivation is polynomial in the size of* $F(x_1, \ldots, x_n)$ *and of the* $A_i$*'s.*

**Proof** By induction on $m$, we show how to derive the sequents $\rightarrow F(A_1, \ldots, A_m, 0^i, 1^{n-m-i})$, $0 \leq i \leq n-m$. The base case, $m = 0$, is easy since the sequents $\rightarrow F(0^i, 1^{n-i})$, $0 \leq i \leq n$, contain no variables.

Suppose that the case $m$ holds. Let $i$ be arbitrary. We want to derive the sequent $\rightarrow F(A_1, \ldots, A_{m+1}, 0^i, 1^{n-(m+1)-i})$. By Lemma 6, it is sufficient to derive $\rightarrow F(A_1, \ldots, A_m, 0, 0^i, 1^{n-(m+1)-i})$ and $\rightarrow F(A_1, \ldots, A_m, 1, 0^i, 1^{n-(m+1)-i})$. These two sequents follow from the inductive hypothesis by permuting the arguments of $F$.

The bound on the size of the derivation is easy to verify. In particular, the total number of permutation sequents used is bounded by $n^2$. □

**Lemma 8** *In* $\mathrm{AC}^0$*-Frege, if* $\Gamma \rightarrow \Delta$ *is a tautology with at most* $O(\log n)$ *variables, then* $\Gamma \rightarrow \Delta$ *can be proved in size polynomial in* $n$ *and in the size of* $\Gamma \rightarrow \Delta$.

**Proof** Since the total number of variables is only $O(\log n)$, the total number of truth assignments to the variables is $n^{O(1)}$. The proof proceeds by giving linear size proofs (in the size of the sequent) of $\tau, \Gamma \rightarrow \Delta$, where $\tau$ is a set of literals, corresponding to a particular truth assignment to all $O(\log n)$ variables. Then these proofs are combined using repeated applications of the cut rule to remove the literals in $\tau$, one-by-one. □

**Lemma 9** *Let* $\Gamma \rightarrow \Delta$ *be an* $\mathrm{AC}^0$*-Frege tautology with underlying variables* $x_1, \ldots, x_m$*. Let* $f_1, \ldots, f_q$ *be disjoint subformulas occurring in* $\Gamma \rightarrow \Delta$*. Let* $\Gamma' \rightarrow \Delta'$ *be the result of replacing every occurrence of each subformula* $f_i$ *by the variable* $A_i$*. Suppose that the* $A_i$*'s are now the only variables in* $\Gamma' \rightarrow \Delta'$*. If* $\Gamma' \rightarrow \Delta'$ *is also a tautology and* $q = O(\log n)$*, then* $\Gamma \rightarrow \Delta$ *has an* $\mathrm{AC}^0$*-Frege proof of size polynomial in* $n$ *and in the size of* $\Gamma \rightarrow \Delta$.

**Proof** The proof is very similar to the one above, except that now we obtain linear size proofs (in the size of the sequent) of $\tau, \Gamma \rightarrow \Delta$, but where now $\tau$

corresponds to a particular truth assignment to all of the $O(\log n)$ formulas $A_1, \ldots, A_q$. Since $\Gamma' \to \Delta'$ is a tautology, each of these $n^{O(1)}$ sequents is true and has a simple linear sized proof. Now again, we use repeated applications of the cut rule (now applied to constant-depth formulas) to remove all of the formulas in $\tau$, one-by-one. □

# 6 Simulating the restricted $\mathrm{TC}^0$-Frege proofs

Let $P$ denote a $\mathrm{TC}^0$-Frege proof of a sequent $\Gamma \to \Delta$. Suppose that $P$ has size polynomial in $n$ and that all the threshold and parity connectives in $P$ have fan-in polylog $n$. Our goal in this section is to show that $P$ can be simulated by an $\mathrm{AC}^0$-Frege proof of size polynomial in $n$. This will be done by *translating* the lines $L_1, \ldots, L_{|P|}$ of $P$ into equivalent $\mathrm{AC}^0$-Frege sequents that will constitute the skeleton of an $\mathrm{AC}^0$-Frege proof. More precisely, each line $L_i$ will be translated into $L'_i$ and $L'_1, \ldots, L'_{|P|-1}$ will become intermediate lines in an $\mathrm{AC}^0$-Frege proof of $L'_{|P|}$.

An $\mathrm{AC}^0$ formula $A'$ is an $\mathrm{AC}^0$ *translation* of a $\mathrm{TC}^0$ formula $A$ if $A'$ can be obtained by replacing every threshold and parity connective in $A$ by the TH and PARITY formulas defined in Section 4. Note that if $A$ has size polynomial in $n$ and if the threshold and parity connectives in $A$ all have fan-in polylog $n$, then $A'$ has size polynomial in $n$. Also note that $A'$ is not unique since the arguments of the connectives are multi-sets while the inputs to the TH and PARITY formulas are ordered. The notion of an $\mathrm{AC}^0$ translation extends in the obvious way to sequents.

The main result of this section can now be stated precisely.

**Theorem 10** *If $\Gamma \to \Delta$ has a $\mathrm{TC}^0$-Frege proof of size polynomial in $n$ in which all the threshold and parity connectives have fan-in* polylog $n$*, then the $\mathrm{AC}^0$ translation of $\Gamma \to \Delta$ has an $\mathrm{AC}^0$-Frege proof of size polynomial in $n$.*

The proof will be by induction on the number of steps in $P$. For $i = 1, \ldots, |P|$, we will show that there is an $\mathrm{AC}^0$-Frege proof of $L'_i$, of size polynomial in $n$, with intermediate lines $L'_1, \ldots, L'_{i-1}$.

For the inductive basis, we need to give polynomial-size $\mathrm{AC}^0$-Frege proofs of the initial sequents of the $\mathrm{TC}^0$-Frege system. The first of these sequents is $A \to A$ which translates to $A' \to A''$ where $A'$ and $A''$ are two—possibly different—$\mathrm{AC}^0$ translations of $A$. Our first task is therefore to give a polynomial-size $\mathrm{AC}^0$-Frege proof of $A' \to A''$. We start with the following lemma.

**Lemma 11** *Let $m = \text{polylog}\, n$. The sequents*

$$\text{TH}_k(A_1, \ldots, A_m) \rightarrow \text{TH}_k(A_{\pi(1)}, \ldots, A_{\pi(m)})$$

*and*

$$\text{PARITY}_b(A_1, \ldots, A_m) \rightarrow \text{PARITY}_b(A_{\pi(1)}, \ldots, A_{\pi(m)}),$$

*where $\pi$ is any permutation, have $\text{AC}^0$-Frege proofs of size polynomial in $n$.*

**Proof** The formula $\text{TH}_k(A_1, \ldots, A_m)$ is defined as $\text{GE}(\text{SUM}(A_1, \ldots, A_m), k)$ and the circuit $\text{SUM}(A_1, \ldots, A_m)$ has only $O(\log\log n)$ outputs. Therefore, by Lemma 9, the sequent

$$\begin{array}{l} \text{TH}_k(A_1, \ldots, A_m), \\ (\text{SUM}(A_1, \ldots, A_m) = \text{SUM}(A_{\pi(1)}, \ldots, A_{\pi(m)})) \\ \quad \rightarrow \quad \text{TH}_k(A_{\pi(1)}, \ldots, A_{\pi(m)}) \end{array}$$

has an $\text{AC}^0$-Frege proof. This implies that to prove $\text{TH}_k(A_1, \ldots, A_m) \rightarrow \text{TH}_k(A_{\pi(1)}, \ldots, A_{\pi(m)})$, it is sufficient to prove that $\text{SUM}(A_1, \ldots, A_m) = \text{SUM}(A_{\pi(1)}, \ldots, A_{\pi(m)})$. The same is true for the $\text{PARITY}_b$ sequent.

In order to show that $\text{SUM}(A_1, \ldots, A_m) = \text{SUM}(A_{\pi(1)}, \ldots, A_{\pi(m)})$, it suffices to show that

$$\begin{array}{l} \text{SUM}(A_1, \ldots, A_r, \ldots, A_s, \ldots, A_m) \\ \quad = \quad \text{SUM}(A_1, \ldots, A_s, \ldots, A_r, ..A_m). \end{array}$$

In other words, it suffices to show that the result holds when we transpose two elements, $A_r$ and $A_s$. The idea will be to rewrite $\text{SUM}(A_1, \ldots, A_r, \ldots, A_s, \ldots, A_m)$ in terms of the variables $A_r$ and $A_s$, and $O(\log n)$ new (meta)variables. This will be done by replacing most of the subformulas of the original formula by these new variables. The formula $\text{SUM}(A_1, \ldots, A_s, \ldots, A_r, \ldots, A_m)$ will be rewritten in a similar way. The resulting two formulas will be truth-functionally equivalent, and since they will involve only $O(\log n)$ variables, we will be able to apply Lemma 9 to complete the proof. In order to see how to do this, we will need some notation.

Recall that the SUM circuit on $m = (\log n)^k$ 1-bit inputs is divided into $2k$ levels, where each level consists of depth-2 TT circuits. Let $j = (\log n)^{1/2}$. Then the SUM circuit on $A_1, \ldots, A_m$ can be viewed as a tree with $2k$ levels. Let $\rho$ denote a particular path in this tree. (So the nodes in the tree at level 1 have path names $1, \ldots, j$; the nodes in the tree at level 2 have path names $11, 12, \ldots, 1j, 21, 22, \ldots, 2j, \ldots, j1, \ldots, jj$ and so on.) Then $X_i^\rho$ will denote

the subcircuit at level $i$ in the tree obtained by following the path $\rho$. In this notation, we have $\mathrm{SUM}(A_1, \ldots, A_m) = \mathrm{TT}(X_1^1, X_1^2, \ldots, X_1^j)$ and in general $X_i^\rho = \mathrm{TT}(X_{i+1}^{\rho,1}, X_{i+1}^{\rho,2}, \ldots, X_{i+1}^{\rho,j})$. Also, notice that $X_{2k}^\rho$ are vectors of $j$ input variables.

Assume for notational simplicity that $A_r \in X_{2k}^{11\cdots1}$ and $A_s \in X_{2k}^{jj\cdots j}$. That is, $A_r$ is the very first variable and $A_s$ is the very last variable. Then we will write $\mathrm{SUM}(A_1, \ldots, A_m)$ as follows:

$$
\begin{aligned}
&\mathrm{SUM}(A_1, \ldots, A_m) \\
&\quad = \mathrm{TT}(X_1^1, X_1^2, \ldots, X_1^j) \\
&\quad = \mathrm{TT}( \\
&\qquad \mathrm{TT}(X_2^{11}, X_2^{12}, \ldots, X_2^{1j}), \\
&\qquad X_1^2, \ldots, X_1^{j-1}, \\
&\qquad \mathrm{TT}(X_2^{j1}, \ldots, X_2^{jj}) ) \\
&\quad = \mathrm{TT}( \\
&\qquad \mathrm{TT}( \\
&\qquad\quad \mathrm{TT}(X_3^{111}, \ldots, X_3^{11j}), \\
&\qquad\quad X_2^{12}, \ldots, X_2^{1j} ), \\
&\qquad X_1^2, \ldots, X_1^{j-1}, \\
&\qquad \mathrm{TT}( \\
&\qquad\quad X_2^{j1}, \ldots, X_2^{j(j-1)}, \\
&\qquad\quad \mathrm{TT}(X_3^{jj1}, \ldots, X_3^{jjj}) ) ).
\end{aligned}
$$

The idea of the above representation is that we are representing most of the SUM circuit by large subformulas that are never looked at; only the part of the circuit that must be opened up in order to look at $A_r$ and $A_s$ will be represented. Thus, in this representation, the number of metavariables that are represented in total is $4kj(\mathrm{polylog}\, n) = O(\log n)$. This is because at each level, we are adding $2j$ new variables, each of length $\mathrm{polylog}\, n$ and the number of levels is $2k$.

In the same manner, we break up the formula

$$\mathrm{SUM}(A_1, \ldots, A_s, \ldots, A_r, \ldots, A_m)$$

with $A_r$ and $A_s$ transposed. Again, this formula will involve $O(\log n)$ metavariables, and these metavariables will be identical to the metavariables involved in $\mathrm{SUM}(A_1, \ldots, A_m)$. Furthermore, these two formulas (on $O(\log n)$ metavariables) are equivalent. Thus we can apply Lemma 9 to complete the proof. □

**Lemma 12** *Let $A'$ and $A''$ be* $\mathrm{AC}^0$ *translations of the same* $\mathrm{TC}^0$*-Frege formula $A$. Suppose that $A$ has size polynomial in $n$ and that all the threshold and parity connectives in $A$ have fan-in* $\mathrm{polylog}\, n$*. Then the sequent $A' \to A''$ has an* $\mathrm{AC}^0$*-Frege proof of size polynomial in $n$.*

**Proof** The proof is by induction on the structure of $A$. The inductive basis is trivial. For the inductive step, several cases need to be considered depending on the top connective of $A$. Suppose, for example, that $A$ is a formula of the form $\mathrm{Th}_k(A_1, \ldots, A_m)$. Then $A' = \mathrm{TH}_k(A'_{\pi(1)}, \ldots, A'_{\pi(m)})$ and $A'' = \mathrm{TH}_k(A''_{\sigma(1)}, \ldots, A''_{\sigma(m)})$, where $\pi$ and $\sigma$ are permutations and the primes and double primes indicate different $\mathrm{AC}^0$ translations of the same formula. We want to derive $A' \to A''$. By Lemma 11, it is sufficient to derive $\mathrm{TH}_k(A'_1, \ldots, A'_m) \to \mathrm{TH}_k(A''_1, \ldots, A''_m)$.

Let $F(x) = \mathrm{TH}_k(A'_1, \ldots, A'_{m-1}, x)$. By Lemma 4, and by the inductive hypothesis applied to $A_m$, we can derive $F(A'_m) \to F(A''_m)$, that is,

$$\mathrm{TH}_k(A'_1, \ldots, A'_{m-1}, A'_m) \to \mathrm{TH}_k(A'_1, \ldots, A'_{m-1}, A''_m).$$

Repeat this, with a different formula $F(x)$, to get

$$\mathrm{TH}_k(A'_1, \ldots, A'_{m-2}, A'_{m-1}, A''_m) \to \mathrm{TH}_k(A'_1, \ldots, A'_{m-2}, A''_{m-1}, A''_m).$$

Continue repeating until we get

$$\mathrm{TH}_k(A'_1, A''_2, \ldots, A''_m) \to \mathrm{TH}_k(A''_1, A''_2, \ldots, A''_m).$$

A series of cuts will now produce the desired sequent.

The other cases are similar and the bound on the size of the proof is easy to verify. □

Note that the proof of Lemma 11 is the only place in the proof of Theorem 10 where we mention the particular definitions we are using for the TH and PARITY formulas. Therefore, our proof of Theorem 10 works with any kind of $\mathrm{AC}^0$ translation that is obtained by replacing every threshold and parity connective by $\mathrm{AC}^0$ formulas that satisfy the property stated in Lemma 11.

Let us now return to the inductive basis of the proof of Theorem 10. The initial sequent $A \to A$ is taken care of by Lemma 12. The sequents $\to \wedge()$ and $\vee() \to$ remain unchanged under $\mathrm{AC}^0$ translation and are therefore handled by the identical $\mathrm{AC}^0$-Frege initial sequents. Next, the sequents $\oplus_1() \to$, $\to \oplus_0()$ and $\mathrm{Th}_k() \to$, for $k \geq 1$, become $\mathrm{PARITY}_1() \to$, $\to \mathrm{PARITY}_0()$ and $\mathrm{TH}_k() \to$, respectively. These are all tautologies with no variables that can

therefore be easily proven. Finally, the sequent $\rightarrow \mathrm{Th}_0(A_1, \ldots, A_m)$ becomes $\rightarrow \mathrm{TH}_0(A_1, \ldots, A_m)$, a tautology that can be proven using Lemmas 7 and 11.

We now move to the inductive step. Suppose that we have an $\mathrm{AC}^0$-Frege proof of $L'_i$, of size polynomial in $n$, with intermediate lines $L'_1, \ldots, L'_{i-1}$. We want to get an $\mathrm{AC}^0$-Frege proof of $L'_{i+1}$, of size polynomial in $n$, with intermediate lines $L'_1, \ldots, L'_i$. In the original $\mathrm{TC}^0$-Frege proof $P$, $L_{i+1}$ is either an initial sequent or obtained from previous sequents by one of the $\mathrm{TC}^0$-Frege inference rules. If $L_{i+1}$ is an initial sequent, then we are done by the argument used in the inductive basis. So suppose that $L_{i+1}$ was obtained from previous sequents by one of the $\mathrm{TC}^0$-Frege inference rules. We will show how to simulate these rules using $\mathrm{AC}^0$-Frege proofs of size polynomial in $n$.

All of the structural rules as well as the cut, $\neg$-left, $\neg$-right, $\wedge$-left, $\wedge$-right, $\vee$-left and $\vee$-right rules can be easily simulated by using Lemma 12 and the corresponding $\mathrm{AC}^0$-Frege rules. We are left with the $\oplus$-left, $\oplus$-right, Th-left and Th-right rules.

Consider the Th-right rule. Suppose that $L_{i+1}$ is a sequent of the form $\Gamma \rightarrow \mathrm{Th}_k(A_1, \ldots, A_n), \Delta$ and that $L_{i+1}$ was derived from $\Gamma \rightarrow A_1, \mathrm{Th}_k(A_2, \ldots, A_n), \Delta$ and $\Gamma \rightarrow \mathrm{Th}_{k-1}(A_2, \ldots, A_n), \Delta$. We need to show that $\Gamma' \rightarrow \mathrm{TH}_k(A'_{\pi(1)}, \ldots, A'_{\pi(n)}), \Delta'$ can be derived from $\Gamma'' \rightarrow A''_1, \mathrm{TH}_k(A''_{\sigma(2)}, \ldots, A''_{\sigma(n)}), \Delta$ and $\Gamma''' \rightarrow \mathrm{TH}_{k-1}(A'''_{\tau(2)}, \ldots, A'''_{\tau(n)}), \Delta'''$, where $\pi$, $\sigma$ and $\tau$ are permutations and the primes, double primes and triple primes indicate different $\mathrm{AC}^0$ translations of the same formula or sequent. By Lemmas 11 and 12, it is sufficient to show that $\Gamma' \rightarrow \mathrm{TH}_k(A'_1, \ldots, A'_n), \Delta'$ can be derived from $\Gamma' \rightarrow A'_1, \mathrm{TH}_k(A'_2, \ldots, A'_n), \Delta$ and $\Gamma' \rightarrow \mathrm{TH}_{k-1}(A'_2, \ldots, A'_n), \Delta'$. We will use the following lemma:

**Lemma 13** *Let $m = \mathrm{polylog}\, n$. The following sequents have $\mathrm{AC}^0$-Frege proofs of size polynomial in $n$.*

1. $\mathrm{TH}_k(A_2, \ldots, A_m) \rightarrow \mathrm{TH}_k(A_1, \ldots, A_m)$
2. $A_1, \mathrm{TH}_{k-1}(A_2, \ldots, A_m) \rightarrow \mathrm{TH}_k(A_1, \ldots, A_m)$
3. $\mathrm{TH}_k(A_1, \ldots, A_m) \rightarrow \mathrm{TH}_{k-1}(A_2, \ldots, A_m)$
4. $\mathrm{TH}_k(A_1, \ldots, A_m) \rightarrow A_1, \mathrm{TH}_k(A_2, \ldots, A_m)$

**Proof** Consider the first sequent. Let

$$G_0(A_2, \ldots, A_m) = \mathrm{TH}_k(0, A_2, \ldots, A_m) \vee \neg \mathrm{TH}_k(A_2, \ldots, A_m)$$

and

$$G_1(A_2, \ldots, A_m) = \mathrm{TH}_k(1, A_2, \ldots, A_m) \vee \neg \mathrm{TH}_k(A_2, \ldots, A_m).$$

Using Lemma 11, we can easily prove that the arguments of $G_0$ and $G_1$ can be permuted. Therefore, by Lemma 7, we get $\rightarrow G_0(A_2, \ldots, A_m)$ and $\rightarrow G_1(A_2, \ldots, A_m)$. Now by Lemma 6, we get

$$\rightarrow \mathrm{TH}_k(A_1, A_2, \ldots, A_m) \vee \neg \mathrm{TH}_k(A_2, \ldots, A_m).$$

The first sequent can be easily derived from this. The proof of the other sequents is similar. □

Continuing with the simulation of the Th-right rule, let $A' = A'_1$, $B' = \mathrm{TH}_k(A'_2, \ldots, A'_m)$, $C' = \mathrm{TH}_{k-1}(A'_2, \ldots, A'_m)$ and $D' = \mathrm{TH}_k(A'_1, \ldots, A'_m)$. We want to derive $\Gamma' \rightarrow D', \Delta'$ from $\Gamma' \rightarrow A', B', \Delta'$ and $\Gamma' \rightarrow C', \Delta'$. From the second sequent in Lemma 13, we have $A', C' \rightarrow D'$. Using this together with $\Gamma' \rightarrow C', \Delta'$ we can apply weakening and cut to derive $\Gamma', A' \rightarrow D', \Delta'$. Now applying cut to this formula together with $\Gamma' \rightarrow A', B', \Delta'$ yields the formula $\Gamma' \rightarrow D', B', \Delta'$. Finally, applying weakening and cut to this formula together with $B' \rightarrow D'$, the first sequent in Lemma 13, we derive $\Gamma' \rightarrow D', \Delta'$ as desired.

The simulation of the threshold-left rule is similar. The simulation of the $\oplus$ rules is also similar except that it uses the following lemma instead of Lemma 13:

**Lemma 14** *Let $m = \mathrm{polylog}\, n$. The following sequents have $\mathrm{AC}^0$-Frege proofs of size polynomial in $n$.*

1. $A_1, \mathrm{PARITY}_b(A_1, \ldots, A_m) \rightarrow \mathrm{PARITY}_{b-1}(A_2, \ldots, A_m)$
2. $\mathrm{PARITY}_b(A_1, \ldots, A_m) \rightarrow A_1, \mathrm{PARITY}_b(A_2, \ldots, A_m)$
3. $A_1, \mathrm{PARITY}_{b-1}(A_2, \ldots, A_m) \rightarrow \mathrm{PARITY}_b(A_1, \ldots, A_m)$
4. $\mathrm{PARITY}_b(A_2, \ldots, A_m) \rightarrow A_1, \mathrm{PARITY}_b(A_1, \ldots, A_m)$

The proof this lemma is similar to that of Lemma 13.

# 7 Our main result

We are now ready to prove our main theorem.

**Theorem 15** *Assuming that the Diffie-Hellman function cannot be computed with circuits of size $2^{n^\epsilon}$ for any $\epsilon > 0$, $\mathrm{AC}^0$-Frege does not have feasible interpolation.*

**Proof** $\mathrm{DH}_m$, as defined by [**?**], is a $\mathrm{TC}^0$-Frege formula with $m$ variables and of size polynomial in $m$. By the main theorem of [**?**], $\mathrm{DH}_m$ has a $\mathrm{TC}^0$-Frege refutation of size polynomial in $m$. Setting $m = \mathrm{polylog}\, n$, by Theorem 10, it follows that the $\mathrm{AC}^0$ translation of $\mathrm{DH}_m$ has an $\mathrm{AC}^0$-Frege refutation of size polynomial in $n$. Note that any $\mathrm{AC}^0$ translation of $\mathrm{DH}_m$ has the same interpolation function as $\mathrm{DH}_m$ itself. Thus, if $\mathrm{AC}^0$-Frege has feasible interpolation, then for every $k$, the Diffie-Hellman function on $(\log n)^k$ many bits has circuits of size polynomial in $n$. The result follows. □

# References