comput. complex. **13** (2004), 47 – 68
1016-3328/04/010047–22
DOI 10.1007/s00037-004-0183-5

computational complexity

NON-AUTOMATIZABILITY OF BOUNDED-DEPTH FREGE PROOFS

Maria Luisa Bonet, Carlos Domingo,
Ricard Gavaldà, Alexis Maciel, and Toniann Pitassi

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.

Keywords. Proof complexity, Frege proof systems, interpolation, automatizability of proof systems.
Subject classification. 03F20, 68T15, 68Q17.

1. Introduction

1.1. Feasible interpolation and automatizability. In recent years there has been a lot of interest in studying the complexity of propositional proof systems. The motivation is twofold. First, Cook & Reckhow (1979) 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 $A(\vec{x}, \vec{z}) \to B(\vec{y}, \vec{z})$ 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 F cannot be computed by polynomial-size circuits.

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 $A_0(\vec{x},\vec{z}) \wedge A_1(\vec{y},\vec{z})$ 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 the associated monotone interpolation function must distinguish between graphs containing a k-clique and no other edges, and graphs with $k-1$-cocliques. For appropriately chosen k, this is known to require exponential-size monotone circuits (Alon & Boppana 1987; Andreev 1985; Razborov 1985). Thus, one can obtain superpolynomial lower bounds for any proof system satisfying: (i) the above statement can be expressed by a polynomial-size formula, 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, Cutting Planes and generalizations of Cutting Planes (Bonet *et al.* 1997a; Cook & Haken 1995; Impagliazzo *et al.* 1994; Krajíček 1997, 1998; Pudlák 1997), relativized bounded arithmetic (Razborov 1995), Hilbert's Nullstellensatz (Pudlák & Sgall 1998), the polynomial calculus (Pudlák & Sgall 1998), and the Lovász–Schriver proof system (Pudlák 1999).

On the other hand, in a separate sequence of papers beginning with a key idea due to Krajíček and Pudlák (Krajíček & Pudlák 1995; Bonet *et al.* 1997b), 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 it 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 (Krajíček & Pudlák 1995), Frege and even TC^0-Frege systems (Bonet *et al.* 1997b). These negative results are important not only as a guide for searching for lower bound techniques, but also because they imply that the proof systems in question cannot be automatized. This connection was first made explicit by Bonet *et al.* (1997b) 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 *et al.* (1997b) 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 non-existence of feasible interpolation implies that the proof system cannot be automatized.

1.2. Our results. Our paper nearly completes the picture of which proof systems do and do not have the feasible interpolation property. Resolution is the simplest and most 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 is known to have feasible interpolation (Bonet *et al.* 1997a; Krajíček 1998; Razborov 1995). Bounded-depth Frege systems, a subsystem of Frege systems, is a more powerful proof system than Resolution, where one can manipulate for-

mulas of bounded depth. In a bounded-depth Frege system, one can prove the weak pigeonhole principle in quasipolynomial size (Paris *et al.* 1988), whereas Resolution proofs have exponential size (Buss & Turan 1988). In this paper, we show 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).

Our results use and extend the ideas in Bonet *et al.* (1997b). More precisely, we 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^ϵ} for arbitrarily small $\epsilon > 0$. Note that our assumption is stronger than that of Bonet *et al.* (1997b) 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 (Biham *et al.* 1999). (See also McCurley 1988; Shmuely 1985.)

The basic idea behind the result of Bonet *et al.* (1997b) is as follows. They construct a TC^0-Frege formula 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 TC^0-Frege refutation of DH_n. On the other hand, an interpolation function for DH_n computes one bit of the secret key exchanged by the Diffie–Hellman procedure. Thus, if 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 $\mathrm{polylog}\, n$. Consider DH_m where $m = \mathrm{polylog}\, n$. By directly applying the main theorem of Bonet *et al.* (1997b), DH_m has a TC^0-Frege refutation of size polynomial in m. We will show how to simulate this refutation with an AC^0-Frege refutation of size polynomial in n. More generally, we will show that any TC^0-Frege proof of size polynomial in n in which all the threshold and parity connectives have fan-in $\mathrm{polylog}\, n$ can be simulated by an AC^0-Frege proof of size polynomial in n. Consequently, if AC^0-Frege admits feasible interpolation, then the secret key exchanged by the Diffie–Hellman procedure can be broken using circuits of size 2^{m^ϵ} for every $\epsilon > 0$. This contradicts the widely believed conjecture that factoring Blum integers is exponentially hard, in the sense that it requires circuits of size 2^{m^δ} for some $\delta > 0$.

The idea of scaling the input from n to $\mathrm{polylog}\, n$ in order to be able to represent a cryptosystem in AC^0 instead of TC^0 to obtain certain results is not new. Kharitonov (1993) improved previously known negative results for the

learnability of Boolean circuit classes from TC^0 to AC^0 in a very similar way. However, it is important to note that, although there is a parallel between the negative results for learning and the negative results for feasible interpolation, the proofs in the latter case are more involved.

Finally, note that we defined feasible interpolation, as usual, in terms of polynomial-size circuits. While our results show that bounded-depth Frege systems do not have feasible interpolation, they leave open the question of a quasipolynomial-size version of this property.

The paper is organized as follows. In Section 2, we provide a more detailed overview of the non-interpolation result for TC^0-Frege (Bonet *et al.* 1997b). In Section 3, we define the AC^0 and TC^0-Frege systems. In Section 4, we define some AC^0 formulas used in the simulation. In Section 5, we prove some preliminary lemmas. In Section 6, we show how to simulate the restricted TC^0-Frege proofs mentioned earlier. In Section 7, we prove our main result.

2. Overview of the TC^0-Frege non-interpolation result

We briefly outline here the proof that TC^0-Frege does not have feasible interpolation, and therefore is not automatizable (Bonet *et al.* 1997b).

The formulas A_0 and A_1 are based on the Diffie–Hellman secret key exchange scheme (Diffie & Hellman 1976). The propositional statement DH is defined as

$$\mathrm{DH}_n = A_0(P, g, X, Y, a, b) \wedge A_1(P, g, X, Y, c, d).$$

The common variables are the integers X, Y, P, and g. P represents a number (not necessarily a prime) of length n, and g an element of the group Z_P^*. The private variables for A_0 are the integers a and b, and the private variables for A_1 are the integers c and 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 $g^c \bmod P = X$, $g^d \bmod P = Y$, and $g^{cd} \bmod P$ is odd. The statement $A_0 \wedge A_1$ is unsatisfiable since (informally) if A_0 and A_1 are both true we have

$$\begin{aligned} g^{ab} \bmod P &= (g^a \bmod P)^b \bmod P = X^b \bmod P \\ &= (g^c \bmod P)^b \bmod P = g^{bc} \bmod P \\ &= (g^b \bmod P)^c \bmod P = Y^c \bmod P \\ &= (g^d \bmod P)^c \bmod P = g^{cd} \bmod P. \end{aligned}$$

An interpolant function for DH_n computes the least significant bit of the secret key exchanged by the Diffie–Hellman procedure. Since the argument also

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 has been proven that for $P = p_1 p_2$, where p_1, p_2 are both primes such that $p_1 \bmod 4 = p_2 \bmod 4 = 3$ (i.e., P is a Blum integer), breaking the Diffie–Hellman cryptographic scheme is harder than factoring P (Biham *et al.* 1999) (see also McCurley 1988; Shmuely 1985). Therefore, if TC^0-Frege admits feasible interpolation, then there exist polynomial-size circuits for factoring Blum integers.

The formal statement of DH_n as a TC^0-Frege formula relies on the fact that the following arithmetic functions can be computed by polynomial-size TC^0 circuits:

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

In fact, the bulk of the work in Bonet *et al.* (1997b) consists in describing TC^0-Frege formulas defining these arithmetic functions, and then showing that many basic properties of these functions have polynomial-size TC^0-Frege proofs.

3. AC^0 and TC^0-Frege systems

We will work with the specific bounded-depth threshold logic system TC^0-Frege defined in Maciel & Pitassi (1998) and also used in Bonet *et al.* (1997b). This system is a sequent-calculus logical system where formulas are built up using the connectives $\vee$, $\wedge$, Th_t, $\neg$, and $\oplus_b$. $\mathrm{Th}_t(x)$ is true if and only if the number of 1's in x is at least t, and $\oplus_b(x)$ is true if and only if the number of 1's in x is equal to $b \bmod 2$.

Definition 3.1. Formulas are built up using the connectives $\wedge$, $\vee$, Th_t, $\oplus_1$, $\oplus_0$, $\neg$. All connectives are assumed to have unbounded fan-in. $\mathrm{Th}_t(A_1, \ldots, A_n)$ is interpreted to be true if and only if the number of true A_i's is at least t; $\oplus_b(A_1, \ldots, A_n)$ 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 Th_t. Thus commutativity of

the connectives is implicit. Our proof system operates on sequents which are sets of formulas of the form $A_1, \ldots, A_p \to B_1, \ldots, B_q$. 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) $A \to A$ where A is any formula; (2) $\to \wedge()$; $\vee() \to$; (3) $\oplus_1() \to$; $\to \oplus_0()$; and (4) $\mathrm{Th}_t() \to$ for $t \geq 1$; $\to \mathrm{Th}_0(A_1, \ldots, A_n)$ for $n \geq 0$. The rules of inference are as follows. Note that the logical rules are defined for $n \geq 1$ and $t \geq 1$. 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 $\Gamma, \Gamma' \to \Delta, \Delta'$ from $\Gamma, A \to \Delta$ and $\Gamma' \to A, \Delta'$.

The logical rules are as follows.

1. (Negation-left) From $\Gamma \to A, \Delta$, derive $\neg A, \Gamma \to \Delta$.
2. (Negation-right) From $A, \Gamma \to \Delta$, derive $\Gamma \to \neg A, \Delta$.
3. (And-left) From $A_1, \wedge(A_2, \ldots, A_n), \Gamma \to \Delta$ derive $\wedge(A_1, \ldots, A_n), \Gamma \to \Delta$.
4. (And-right) From $\Gamma \to A_1, \Delta$ and $\Gamma \to \wedge(A_2, \ldots, A_n), \Delta$ derive $\Gamma \to \wedge(A_1, \ldots, A_n), \Delta$.
5. (Or-left) From $A_1, \Gamma \to \Delta$ and $\vee(A_2, \ldots, A_n), \Gamma \to \Delta$ derive $\vee(A_1, \ldots, A_n), \Gamma \to \Delta$.
6. (Or-right) From $\Gamma \to A_1, \vee(A_2, \ldots, A_n), \Delta$ derive $\Gamma \to \vee(A_1, \ldots, A_n), \Delta$.
7. (Mod-left) From $A_1, \oplus_{b-1}(A_2, \ldots, A_n), \Gamma \to \Delta$ and $\oplus_b(A_2, \ldots, A_n), \Gamma \to A_1, \Delta$ derive $\oplus_b(A_1, \ldots, A_n), \Gamma \to \Delta$.
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 $\mathrm{Th}_t(A_2, \ldots, A_n), \Gamma \to \Delta$ and $A_1, \mathrm{Th}_{t-1}(A_2, \ldots, A_n), \Gamma \to \Delta$ derive $\mathrm{Th}_t(A_1, \ldots, A_n), \Gamma \to \Delta$.
10. (Threshold-right) From $\Gamma \to A_1, \mathrm{Th}_t(A_2, \ldots, A_n), \Delta$ and $\Gamma \to \mathrm{Th}_{t-1}(A_2, \ldots, A_n), \Delta$ derive $\Gamma \to \mathrm{Th}_t(A_1, \ldots, A_n), \Delta$.

Note that in the mod rules, the subscript of the $\oplus$ connective should be taken modulo 2. As written, these rules can also be used to introduce mod connectives for other moduli.

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 TC^0-Frege proofs if each sequent has a bounded-depth proof of size polynomial in the size of the sequent. More precisely,

DEFINITION 3.2. Let $F = \{(\Gamma_n \to \Delta_n) : n \in \mathbb{N}\}$ be a family of sequents. Then $\{R_n : n \in \mathbb{N}\}$ is a *family of* 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 $(\mathrm{size}(\Gamma_n \to \Delta_n))^c$.

DEFINITION 3.3. The AC^0-Frege system is a restriction of the TC^0-*Frege system*, where we omit the parity and threshold connectives and the associated rules.

Note that even though the proof systems AC^0-Frege and TC^0-Frege are defined in terms of families of sequents, we will sometimes talk of AC^0-Frege or TC^0-Frege in the context of a single sequent or formula. In that case, the bounds on depth and size are not meaningful and the terminology is used simply to restrict the type of connectives that can appear in the formulas or in the proofs.

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 easy AC^0-Frege proofs.

4. AC^0 counting formulas

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

We will first show how to add $\mathrm{polylog}\, n$ 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, $\mathrm{TT}^{(k-1/2)\log\log n}$, 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. A parity test of the low order bits of x and y gives a constant-size, depth-2 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 $i = (\log n)^{k/2}$. The idea is that $\mathrm{SUM}^{j,i}(x_1, \ldots, x_i)$ will be a full tree consisting of k levels of TT's. We define $\mathrm{SUM}^{j,i}$ as follows:

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

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

6. $\mathrm{TH}^i_t(x_1, \ldots, x_i)$: 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 t or greater. It is defined to be equal to $\mathrm{GE}^{\log i}(\mathrm{SUM}^{1,i}(x_1, \ldots, x_i), t)$. It is important to note that in simulating the original threshold gate, Th_t, we are going from an unordered list of the variables to an ordered list of the variables. That is, in our formula for TH^i_t, the order of the variables matters. Even though commutativity of the underlying variables was implicit in Th_t, we will need to show that permutation of TH_t 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 PARITY_b can be simulated by our formulas.

To simplify notation, we will usually omit the superscripts on the above 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 write $f = g$, 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. The formula 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 5.1 can be proved by induction on the structure of the formula F. Lemma 5.2 then follows from Lemma 5.1, and Lemma 5.3 from Lemma 5.2.

LEMMA 5.1. *Let A, B and $F(x)$ be AC^0-Frege formulas and let Γ and Δ be sequences of AC^0-Frege formulas. Let n be the total size of all these formulas and let d be their maximum depth. Then there is an absolute constant c such that the following sequents have AC^0-Frege proofs of size n^c and depth d:*

(i) $(\Gamma \to F(A), \Delta)$ *from* $(\Gamma \to F(B), \Delta)$, $(\Gamma, B \to A, \Delta)$ *and* $(\Gamma, A \to B, \Delta)$.

(ii) $(\Gamma, F(A) \to \Delta)$ *from* $(\Gamma, F(B) \to \Delta)$, $(\Gamma, B \to A, \Delta)$ *and* $(\Gamma, A \to B, \Delta)$.

LEMMA 5.2. *Let A and $F(x)$ be AC^0-Frege formulas of total size n and maximum depth d. Then there is an absolute constant c such that the following sequents have AC^0-Frege proofs of size n^c and depth d:*

(i) $F(0) \to A, F(A)$.

(ii) $F(1), A \to F(A)$.

(iii) $F(A), A \to F(1)$.

(iv) $F(A) \to A, F(0)$.

PROOF. The first sequent is easily obtained from Lemma 5.1(i) by substituting $F(0)$ for Γ, A for Δ and 0 for B. The other sequents are proved in a similar way. □

LEMMA 5.3. *Let A and $F(x)$ be AC^0-Frege formulas of total size n and maximum depth d. Then there is an absolute constant c such that the following sequents have AC^0-Frege proofs of size n^c and depth d:*

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

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

LEMMA 5.4. *Let $F(x_1, \ldots, x_n)$ be a tautological AC^0-Frege formula. Then, for every sequence of formulas $A_1, \ldots, A_n$, the sequent $\rightarrow F(A_1, \ldots, A_n)$ can be derived in AC^0-Frege from at most n^2 sequents of the form $F(B_1, \ldots, B_n) \rightarrow F(B_{\pi(1)}, \ldots, B_{\pi(n)})$ where π is a permutation. The size of the derivation is a fixed polynomial in the size of $F(A_1, \ldots, A_n)$, and its depth is at most that of $F(A_1, \ldots, A_n)$.*

PROOF. By applying 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 5.3, 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 5.5. *In AC^0-Frege, if $\Gamma \rightarrow \Delta$ is a tautology of size n and depth d with at most m variables, then it has a proof of size $cn^2 2^m$ and depth d, for an absolute constant c. In particular, if $m \leq b \log n$, then the size of the proof is at most cn^{2+b}.*

PROOF. Since the number of variables is m, the number of truth assignments to the variables is 2^m. The proof proceeds by giving proofs of $\tau, \Gamma \rightarrow \Delta$, where τ is a set of literals corresponding to a particular truth assignment to all q variables. These proofs each have size n^2. They are then combined using $2^m - 1$ repeated applications of the cut rule to remove the literals in the τ's, in a tree-like way. □

LEMMA 5.6. *Let $\Gamma \rightarrow \Delta$ be an AC^0-Frege tautology of size n and depth d with underlying variables $x_1, \ldots, x_m$. Let $\Gamma' \rightarrow \Delta'$ be the result of replacing every*

occurrence of each x_i by some formula f_i. Then $\Gamma' \to \Delta'$ has an AC^0*-Frege proof of size $cn^2 2^m$ and depth d, for an absolute constant c. In particular, if $m \leq b \log n$, then the size of the proof is at most cn^{2+b}.*

PROOF. The proof is very similar to the one above, except that we obtain proofs of $\tau, \Gamma \to \Delta$ where now τ corresponds to a particular sequence of truth values for the formulas $f_1, \ldots, f_m$. Since $\Gamma' \to \Delta'$ is a tautology, each of these 2^m sequents is true and has a simple proof of size n^2. Now again, we use $2^m - 1$ applications of the cut rule (now applied to formulas instead of literals) to remove all of the formulas in the τ's. □

6. Simulating the restricted TC^0-Frege proofs

Let P denote a 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 $\operatorname{polylog} n$. Our goal in this section is to show that P can be simulated by an 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 AC^0-Frege sequents that will constitute the skeleton of an 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 AC^0-Frege proof of $L'_{|P|}$.

An AC^0 formula A' is an AC^0 *translation* of a 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 $\operatorname{polylog} n$, 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 AC^0 translation extends in the obvious way to sequents.

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

THEOREM 6.1. *Suppose that the family of sequents $\Gamma \to \Delta$ has a* TC^0*-Frege proof of size polynomial in n in which all the threshold and parity connectives have fan-in* $\operatorname{polylog} n$*. Then any* AC^0 *translation of $\Gamma \to \Delta$ has an* 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 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 AC^0-Frege proofs of the initial sequents of the 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—AC^0 translations of A. Our first task is therefore to give a polynomial-size AC^0-Frege proof of $A' \to A''$. We start with the following lemma.

LEMMA 6.2. *Consider the following sequents:*

(i) $\mathrm{TH}_t(A_1, \ldots, A_m) \to \mathrm{TH}_t(A_{\pi(1)}, \ldots, A_{\pi(m)})$,

(ii) $\mathrm{PARITY}_b(A_1, \ldots, A_m) \to \mathrm{PARITY}_b(A_{\pi(1)}, \ldots, A_{\pi(m)})$,

where $A_1, \ldots, A_m$ are AC^0-Frege formulas and π is any permutation. Assume that these sequents have size n, maximum depth d and that $m = (\log n)^k$. Then these sequents have AC^0-Frege proofs of size n^c and depth d, where c depends only on k.

PROOF. The formula $\mathrm{TH}_t(A_1, \ldots, A_m)$ is defined as $\mathrm{GE}(\mathrm{SUM}(A_1, \ldots, A_m), t)$ and the circuit $\mathrm{SUM}(A_1, \ldots, A_m)$ has only $\log(\log n)^k$ outputs. Therefore, by Lemma 5.6, the sequent

$$\mathrm{TH}_t(A_1, \ldots, A_m), (\mathrm{SUM}(A_1, \ldots, A_m) = \mathrm{SUM}(A_{\pi(1)}, \ldots, A_{\pi(m)}) \to \mathrm{TH}_t(A_{\pi(1)}, \ldots, A_{\pi(m)}),$$

which is of size at most $2n$, has an AC^0-Frege proof of size $c_1(2n)^2(\log 2n)^k$, for some absolute constant c_1. This implies that to prove $\mathrm{TH}_t(A_1, \ldots, A_m) \to \mathrm{TH}_t(A_{\pi(1)}, \ldots, A_{\pi(m)})$, it is sufficient to prove that $\mathrm{SUM}(A_1, \ldots, A_m) = \mathrm{SUM}(A_{\pi(1)}, \ldots, A_{\pi(m)})$. The same is true for the PARITY_b sequent.

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

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

In other words, it suffices to show that the result holds when we transpose two elements, A_r and A_s. The idea is to rewrite $\mathrm{SUM}(A_1, \ldots, A_r, \ldots, A_s, \ldots, A_m)$ 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 $\mathrm{SUM}(A_1, \ldots, A_s, \ldots, A_r, \ldots, A_m)$ 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 5.6 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 ρ 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^ρ will denote the subcircuit at level i in the tree obtained by following the path ρ. 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}^ρ are vectors of j input variables.

Assume for notational simplicity that $A_r \in X_{2k}^{11\ldots1}$ and $A_s \in X_{2k}^{jj\ldots j}$. 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:

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

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, at each of the $2k$ levels, we are adding $2j$ new variables, each of length $\log(\log^k n)$. Therefore, the number of metavariables that are represented in total is $4kj \log(\log^k n) = 4k(\log n)^{1/2}(k \log\log n) \leq 4k^2 \log n$.

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 at most $4k^2 \log n$ metavariables, and these metavariables will be identical to the metavariables involved in $\mathrm{SUM}(A_1, \ldots, A_m)$. Furthermore, these two formulas (on $4k^2 \log n$ metavariables) are equivalent. Thus, by Lemma 5.6, the sequent

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

has a proof of size $c_1 n^{2+4k^2}$ and depth d. This implies that $\mathrm{SUM}(A_1, \ldots, A_m) = \mathrm{SUM}(A_{\pi(1)}, \ldots, A_{\pi(m)})$ has a proof of size $m^2 c_1 n^{2+4k^2}$ and depth d. Therefore, $\mathrm{TH}_t(A_1, \ldots, A_m) \to \mathrm{TH}_t(A_{\pi(1)}, \ldots, A_{\pi(m)})$ has a proof of size $c_1(2n)^2(\log 2n)^k + m^2 c_1 n^{2+4k^2}$, which (when $n \geq 2$) is at most n^c for some number c that depends only on k. □

LEMMA 6.3. *Let A' and A'' be AC^0 translations of the same TC^0-Frege formula A. Suppose that A has size n, depth d and that all the threshold and parity connectives in A have fan-in $m = (\log n)^k$. Then the sequent $A' \to A''$ has an AC^0-Frege proof of size n^c and depth d, where c depends only on k.*

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 $\mathrm{Th}_t(A_1, \ldots, A_m)$. Then $A' = \mathrm{TH}_t(A'_{\pi(1)}, \ldots, A'_{\pi(m)})$ and $A'' = \mathrm{TH}_t(A''_{\sigma(1)}, \ldots, A''_{\sigma(m)})$, where π and σ are permutations and the primes and double primes indicate different AC^0 translations of the same formula. We want to derive $A' \to A''$. By Lemma 6.2, it is sufficient to derive $\mathrm{TH}_t(A'_1, \ldots, A'_m) \to \mathrm{TH}_t(A''_1, \ldots, A''_m)$.

Let $F(x) = \mathrm{TH}_t(A'_1, \ldots, A'_{m-1}, x)$. By Lemma 5.1, and by the inductive hypothesis applied to A_m, we can derive $F(A'_m) \to F(A''_m)$, that is,

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

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

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

Continue repeating until we get

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

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 6.2 is the only place in the proof of Theorem 6.1 where we mention the particular definitions we are using for the TH and PARITY formulas. Therefore, our proof of Theorem 6.1 works with any kind of AC^0 translation that is obtained by replacing every threshold and parity connective by AC^0 formulas that satisfy the property stated in Lemma 6.2.

Let us now return to the inductive basis of the proof of Theorem 6.1. The initial sequent $A \to A$ is taken care of by Lemma 6.3. The sequents $\to \wedge()$ and $\vee() \to$ remain unchanged under AC^0 translation and are therefore handled by the identical AC^0-Frege initial sequents. Next, the sequents $\oplus_1() \to$, $\to \oplus_0()$ and $\mathrm{Th}_t() \to$, for $t \geq 1$, become $\mathrm{PARITY}_1() \to$, $\to \mathrm{PARITY}_0()$ and $\mathrm{TH}_t() \to$, respectively. These are all tautologies with no variables that can therefore be easily proven. Finally, the sequent $\to \mathrm{Th}_0(A_1, \ldots, A_m)$ becomes $\to \mathrm{TH}_0(A_1, \ldots, A_m)$, a tautology that can be proven using Lemma 5.4 and Lemma 6.2.

We now move to the inductive step. Suppose that we have an 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 AC^0-Frege proof of L'_{i+1}, of size polynomial in n, with intermediate lines $L'_1, \ldots, L'_i$. In the original TC^0-Frege proof P, L_{i+1} is either an initial sequent or obtained from previous sequents by one of the 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 TC^0-Frege inference rules. We will show how to simulate these rules using 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 6.3 and the corresponding 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 $\Gamma \to \mathrm{Th}_t(A_1, \ldots, A_n), \Delta$ and that L_{i+1} was derived from $\Gamma \to A_1, \mathrm{Th}_t(A_2, \ldots, A_n), \Delta$ and $\Gamma \to \mathrm{Th}_{t-1}(A_2, \ldots, A_n), \Delta$. We need to show that the sequent $\Gamma' \to \mathrm{TH}_t(A'_{\pi(1)}, \ldots, A'_{\pi(n)}), \Delta'$ can be derived from $\Gamma'' \to A''_1, \mathrm{TH}_t(A''_{\sigma(2)}, \ldots, A''_{\sigma(n)}), \Delta$ and $\Gamma''' \to \mathrm{TH}_{t-1}(A'''_{\tau(2)}, \ldots, A'''_{\tau(n)}), \Delta'''$, where π, σ and τ are permutations and the primes, double primes and triple primes indicate different AC^0 translations of the same formula or sequent. By Lemmas 6.2 and 6.3, it is sufficient to show that $\Gamma' \to \mathrm{TH}_t(A'_1, \ldots, A'_n), \Delta'$ can be derived from $\Gamma' \to A'_1, \mathrm{TH}_t(A'_2, \ldots, A'_n), \Delta$ and $\Gamma' \to \mathrm{TH}_{t-1}(A'_2, \ldots, A'_n), \Delta'$. We will use the following lemma:

LEMMA 6.4. *Consider the following sequents:*

(i) $\mathrm{TH}_t(A_2, \ldots, A_m) \to \mathrm{TH}_t(A_1, \ldots, A_m)$,

(ii) $A_1, \mathrm{TH}_{t-1}(A_2, \ldots, A_m) \to \mathrm{TH}_t(A_1, \ldots, A_m)$,

(iii) $\mathrm{TH}_t(A_1, \ldots, A_m) \to \mathrm{TH}_{t-1}(A_2, \ldots, A_m)$,

(iv) $\mathrm{TH}_t(A_1,\ldots,A_m) \to A_1, \mathrm{TH}_t(A_2,\ldots,A_m)$,

where $A_1,\ldots,A_m$ are AC^0-Frege formulas. Suppose that these sequents have size n, maximum depth d and that $m = (\log n)^k$. Then these sequents have AC^0-Frege proofs of size n^c and depth $d+2$, where c depends only on k.

Proof. Consider the first sequent. Let

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

and

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

Using Lemma 6.2, we can easily prove that the arguments of G_0 and G_1 can be permuted. Therefore, by Lemma 5.4, we get the sequents $\to G_0(A_2,\ldots,A_m)$ and $\to G_1(A_2,\ldots,A_m)$. Now by Lemma 5.3, we get

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

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 $A' = A'_1$, $B' = \mathrm{TH}_t(A'_2,\ldots,A'_m)$, $C' = \mathrm{TH}_{t-1}(A'_2,\ldots,A'_m)$ and $D' = \mathrm{TH}_t(A'_1,\ldots,A'_m)$. We want to derive $\Gamma' \to D', \Delta'$ from $\Gamma' \to A', B', \Delta'$ and $\Gamma' \to C', \Delta'$. From the second sequent in Lemma 6.4, we have $A', C' \to D'$. Using this together with $\Gamma' \to C', \Delta'$ we can apply weakening and cut to derive $\Gamma', A' \to D', \Delta'$. Now applying cut to this formula together with $\Gamma' \to A', B', \Delta'$ yields the formula $\Gamma' \to D', B', \Delta'$. Finally, applying weakening and cut to this formula together with $B' \to D'$, the first sequent in Lemma 6.4, we derive $\Gamma' \to D', \Delta'$ 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 6.4:

Lemma 6.5. *Consider the following sequents:*

(i) $A_1, \mathrm{PARITY}_b(A_1,\ldots,A_m) \to \mathrm{PARITY}_{b-1}(A_2,\ldots,A_m)$,

(ii) $\mathrm{PARITY}_b(A_1,\ldots,A_m) \to A_1, \mathrm{PARITY}_b(A_2,\ldots,A_m)$,

(iii) $A_1, \mathrm{PARITY}_{b-1}(A_2,\ldots,A_m) \to \mathrm{PARITY}_b(A_1,\ldots,A_m)$,

(iv) $\mathrm{PARITY}_b(A_2,\ldots,A_m) \to A_1, \mathrm{PARITY}_b(A_1,\ldots,A_m)$,

where $A_1, \ldots, A_m$ are AC^0*-Frege formulas. Suppose that these sequents have size n, maximum depth d and that $m = (\log n)^k$. Then these sequents have* AC^0*-Frege proofs of size n^c and depth $d + 2$, where c depends only on k.*

The proof this lemma is similar to that of Lemma 6.4.

7. Our main result

We are now ready to prove our main theorem.

THEOREM 7.1. *If there is a constant $\delta > 0$ such that the Diffie–Hellman function cannot be computed with circuits of size less than 2^{n^δ}, then* AC^0*-Frege does not have feasible interpolation.*

PROOF. DH_m, as defined by Bonet *et al.* (1997b), is a TC^0-Frege formula with m variables and of size polynomial in m. By the main theorem of Bonet *et al.* (1997b), DH_m has a TC^0-Frege refutation of size polynomial in m. Let $m = (\log n)^k$, where $k > 1/\delta$. By Theorem 6.1, it follows that the AC^0 translation of DH_m has an AC^0-Frege refutation of size polynomial in n. Note that any AC^0 translation of DH_m has the same interpolation function as DH_m itself. Thus, if AC^0-Frege has feasible interpolation, then the Diffie–Hellman function on $(\log n)^k$ many bits has circuits of size polynomial in n. The result follows. □

Acknowledgements

We thank the anonymous referees for several useful comments. Bonet was supported in part by projects PB98-0937-C04-03, HA2000,41 and TIC2001-1577-C03-02. Gavaldà was supported in part by the IST Programme of the EU under contract number IST-1999-14186 (ALCOM-FT), by CIRIT 1997SGR-00366, TIC2000-1970-CE, and by project FRESCO (PB98-0937-C04-04). Maciel was supported in part by NSF grant CCR-9877150. Pitassi was supported in part by an NSERC Research grant, a Premiere's Research Excellence Award (Ontario), and NSF grant CCR9820831. Most of this work was done while Domingo was at the Department of Software (LSI), Universitat Politècnica de Catalunya, Barcelona, Spain, and Pitassi was at the Department of Computer Science, University of Arizona, Tucson, AZ, U.S.A.

References

N. ALON & R. B. BOPPANA (1987). The monotone circuit complexity of Boolean functions. *Combinatorica* **7**, 1–22.

A. E. Andreev (1985). On a method for obtaining lower bounds for the complexity of individual monotone functions. *Dokl. Akad. Nauk SSSR* **282**, 1033–1037 (in Russian). English transl.: *Soviet Math. Dokl.* **31**, 530–534.

E. Biham, D. Boneh & O. Reingold (1999). Breaking generalized Diffie–Hellman modulo a composite is no easier than factoring. *Inform. Process. Lett.* **70**, 83–87.

M. Bonet, T. Pitassi & R. Raz (1997a). Lower bounds for Cutting Planes proofs with small coefficients. *J. Symbolic Logic* **62**, 708–728.

M. Bonet, T. Pitassi & R. Raz (1997b). No feasible interpolation for TC^0-Frege proofs. In *Proc. 38th IEEE Symposium on Foundations of Computer Science*, 254–263. Final version in *SIAM J. Computing* **29** (2000), 1939–1967, with the title: On interpolation and automatization for Frege proof systems.

S. R. Buss & G. Turan (1988). Resolution proofs of generalized pigeonhole principles. *Theoret. Comput. Sci.* **62**, 311–317.

S. Cook & A. Haken (1995). An exponential lower bound for the size of monotone real circuits. In *Proc. 36th IEEE Symposium on Foundations of Computer Science*. Final version in *J. Comput. System Sci.* **58** (1999), 326–335.

S. Cook & R. Reckhow (1979). The relative efficiency of propositional proof systems. *J. Symbolic Logic* **44**, 36–50.

W. Diffie & M. Hellman (1976). New directions in cryptography. *IEEE Trans. Inform. Theory* **22**, 423–439.

R. Impagliazzo, T. Pitassi & A. Urquhart (1994). Upper and lower bounds for tree-like Cutting Planes proofs. In *Proc. IEEE Symposium on Logic in Computer Science*.

M. Kharitonov (1993). Cryptographic hardness of distribution-specific learning. In *Proc. 25th ACM Symposium on Theory of Computing*, 372–381.

J. Krajíček (1997). Interpolation theorems, lower bounds for proof systems, and independence results for bounded arithmetic. *J. Symbolic Logic* **62**, 457–486.

J. Krajíček (1998). Discretely ordered modules as a first-order extension of the cutting planes proof system. *J. Symbolic Logic* **63**, 1582–486.

J. Krajíček & P. Pudlák (1995). Some consequences of cryptographical conjectures for S^1_2 and EF. In *Logic and Computational Complexity*, D. Leivant (ed.), Lecture Notes in Comput. Sci. 960, Springer, 210–220.

A. Maciel & T. Pitassi (1998). Towards lower bounds for bounded-depth Frege proofs with modular connectives. In *Proof Complexity and Feasible Arithmetics*, P. Beame and S. Buss (eds.), DIMACS Ser. Discrete Math. Theoret. Comput. Sci. 39, Amer. Math. Soc., 195–227.

K. McCurley (1988). A key distribution system equivalent to factoring. *J. Cryptology* **1**, 95–105.

J. B. Paris, A. J. Wilkie & A. R. Woods (1988). Provability of the pigeonhole principle and the existence of infinitely many primes. *J. Symbolic Logic* **53**, 1235–1244.

P. Pudlák (1997). Lower bounds for resolution and cutting planes proofs and monotone computations. *J. Symbolic Logic* **62**, 981–998.

P. Pudlák (1999). On the complexity of propositional calculus. In *Sets and Proofs, Logic Colloquium 97*, B. Cooper and J. Truss (eds.), London Math. Soc. Lecture Note Ser. 258, Cambridge Univ. Press, 197–218.

P. Pudlák & J. Sgall (1998). Algebraic models of computation and interpolation for algebraic proof systems. In *Proof Complexity and Feasible Arithmetics*, P. Beame and S. Buss (eds.), DIMACS Ser. Discrete Math. Theoret. Comput. Sci. 39, Amer. Math. Soc., 179–295.

A. A. Razborov (1985). Lower bounds on the monotone complexity of some Boolean functions. *Dokl. Akad. Nauk SSSR* **281**, 798–801 (in Russian). English transl.: *Soviet Math. Dokl.* **31**, 354–357.

A. A. Razborov (1995). Unprovability of lower bounds on the circuit size in certain fragments of bounded arithmetic. *Izv. Ross. Akad. Nauk Ser. Mat.* **59**, 201–224 (in Russian). English transl.: *Izv. Math.* **59**, 205–227.

Z. Shmuely (1985). Composite Diffie-Hellman public-key generating systems are hard to break. Technical Report 356, Computer Sci. Dept., Technion, Haifa.

Manuscript received 2 August 2000

Maria Luisa Bonet
Department of Software (LSI)
Universitat Politècnica de Catalunya
Barcelona, Spain
bonet@lsi.upc.es

Carlos Domingo
Department of Mathematical and Computing Science
Tokyo Institute of Technology
Ookayama, Meguro-ku, Tokyo, Japan

Ricard Gavaldà
Department of Software (LSI)
Universitat Politècnica de Catalunya
Barcelona, Spain
gavalda@lsi.upc.es

Alexis Maciel
Department of Mathematics and
Computer Science
Clarkson University
Potsdam, NY 13699-5815, U.S.A.
alexis@clarkson.edu

Toniann Pitassi
Department of Computer Science
University of Toronto
Toronto, Ontario, Canada, M5S 3G4
toni@cs.toronto.edu