Electronic Colloquium on Computational Complexity, Comment 1 on Report No. 010 (1999)

A Lower Bound for Primality*

Eric Allender†
Dept. of Computer Science
Rutgers University
Piscataway, NJ, USA
allender@cs.rutgers.edu

Michael Saks‡
Mathematics Dept.
Rutgers University
Piscataway, NJ, USA
saks@math.rutgers.edu

Igor Shparlinski§
Dept. of Computing
Macquarie University
NSW 2109, Australia
igor@comp.mq.edu.au

Abstract

Recent work by Bernasconi, Damm and Shparlinski showed that the set of square-free numbers is not in AC^0, and raised as an open question if similar (or stronger) lower bounds could be proved for the set of prime numbers. In this note, we show that the Boolean majority function is AC^0-Turing reducible to the set of prime numbers (represented in binary). From known lower bounds on Maj (due to Razborov and Smolensky) we conclude that primality can not be tested in $\mathrm{AC}^0[p]$ for any prime p. Similar results are obtained for the set of square-free numbers, and for the problem of computing the greatest common divisor of two numbers.

*A preliminary version of this work appeared in Proc. 14th Annual IEEE Conference on Computational Complexity, 1999, pp. 10–14.

†Supported in part by NSF grant CCR-9734918.

‡Supported in part by NSF grant CCR-9700239.

§Supported in part by ARC grant A69700294.

ISSN 1433-8092

1 Introduction

What is the computational complexity of the set of prime numbers? There is a large body of work presenting important upper bounds on the complexity of the set of primes (including [AH87, APR83, Mil76, R80, SS77]), but – as was pointed out recently in [BDS98a, BDS98b, BS99, Shp98], other than the work of [Med91, Man92] almost nothing has been published regarding *lower bounds* on the complexity of this set. In the context of space-bounded computation, it was shown in [HS68] that at least logarithmic space is required, in order to determine if a number is prime. This was improved in [HB76] to show that the same bound holds even if the number is presented in *unary* (and logarithmic space is sufficient in that case). However, these bounds do not address circuit complexity at all; note for instance that the *unary* encoding of prime numbers has trivial circuit complexity. Prior to the current work, it was not known whether primality testing was in AC^0, i.e., could be accomplished by constant-depth, polynomial-size circuits of AND,OR, and NOT gates.[1]

Recall that if F is a family of Boolean functions $\mathrm{AC}^0[F]$ is the class of functions computable by constant-depth polynomial-size circuits using AND,OR, and NOT gates and gates that compute functions from F. Recall that the Boolean majority function MAJ is defined to be 1 if and only if at least half of the input bits are 1 and the circuit class TC^0 is defined to be $\mathrm{AC}^0[\textsc{Maj}]$. Also for a natural number k, the Boolean function $\textsc{Mod}_k$ is defined to be 1 precisely if k divides the sum of the input bits and the circuit class $\mathrm{AC}^0[k]$ is defined to be $\mathrm{AC}^0[\textsc{Mod}_k]$.

Our results concern three number-theoretic functions. In each case, the input is an integer or sequence of integers given in binary.

- PRIMES is the set of prime integers.
- SQUARE-FREE is the set of integers x that are not divisible by any perfect square greater than 1.
- GCD is the set of triples (x, y, i) of integers such that the i^{th} bit of the greatest common divisor of x and y is 1.

In this note, we prove:

[1]Independently, it was shown in [LV99] that primality testing is not in AC^0, under some unproved number-theoretic assumptions.

Theorem 1 *TC^0 is contained in each of the classes AC^0[*PRIMES*], AC^0[*GCD*] and AC^0[*SQUARE-FREE*].*

It is well-known that for any k, $\text{MOD}_k \in \text{TC}^0$ and thus our results imply that MOD_k belongs to each of the classes AC^0[PRIMES], AC^0[GCD] and AC^0[SQUARE-FREE]. A fundamental result of Smolensky [Smo87] (building on earlier work of Razborov [Raz87]) says that if p and q are distinct primes, and $\{C_n : n \in \mathbb{N}\}$ is a sequence of bounded depth circuits using AND, OR, NOT and MOD_p gates such that C_n computes $\text{MOD}_q(x)$ on inputs of size n, then the size of C_n is exponential in n^δ for some constant $\delta > 0$.

From this we conclude:

Corollary 2 *For any prime p, any circuit family of bounded depth of AND, OR, NOT and* MOD_p *gates that computes* PRIMES, GCD, *or* SQUARE-FREE *for n bit inputs has size exponential in n^δ for some $\delta > 0$. In particular these three languages are not in $AC^0[p]$.*

For the case of SQUARE-FREE and GCD our results strengthen and simplify those of [BDS98a, BDS98b, BS99] who showed that these two languages are not in AC^0.

2 Preliminaries

2.1 Languages, functions and circuits

For a string $x \in \{0,1\}^*$, $|x|$ denotes the length of x. Since we are dealing with number-theoretic functions and our strings often represent integers, it is convenient to index our strings so that $x = x_{n-1} \dots x_1 x_0$ so that the integer represented by x is $\sum_i x_i 2^i$.

A Boolean function family f is a sequence $\{f_n : n \in \mathbb{N}\}$ of Boolean functions where for some function $l(n) = l_f(n)$ that is bounded by a polynomial in n, f_n maps $\{0,1\}^{l(n)}$ to $\{0,1\}$. (The typical case is $l(n) = n$, but it is often convenient to allow other functions). We follow the common abuse of notation that f can denote both the family of functions or a single function f_n in the class. We make the usual association between languages over $\{0,1\}^*$ and function families.

A *circuit family* C is a set $\{C_n : n \in \mathbb{N}\}$ where each C_n is an acyclic circuit with $l(n) = l_C(n)$ Boolean inputs $x_{l(n)-1}, \dots, x_0$ (where $l(n)$ is bounded by some polynomial function of n), and some number, $r_C(n)$, of outputs. $\{C_n\}$

has *size* $s(n)$ if each circuit C_n has at most $s(n)$ gates; it has *depth* $d(n)$ if the length of the longest path from input to output in C_n is at most $d(n)$.

A function (family) f is said to be in AC^0 if there is a circuit family $\{C_n\}$ of size $n^{O(1)}$ and depth $O(1)$ consisting of unbounded fan-in AND and OR and NOT gates such that for each n, C_n computes f_n on inputs of length $l_f(n)$.

We note the following well-known facts:

Proposition 3 *The following functions can be computed by bounded depth circuits of size polynomial in n:*

1. *Any function with at most* $\log n$ *input bits and poly*(n) *output bits.*

2. *The difference of two integers of at most poly*(n) *bits.*

3. *The product of a* $\log n$ *bit integer and an* n *bit integer.*

2.2 Reducibility

A language A_1 is $\leq_{\mathrm{m}}^{AC^0}$ *reducible* to a language A_2, written $A_1 \leq_{\mathrm{m}}^{\mathrm{AC}^0} A_2$, if there is a function f in AC^0 such that, for all x, $x \in A_1$ if and only if $f(x) \in A_2$.

A_1 is $\leq_{\mathrm{T}}^{AC^0}$ *reducible* to A_2, written $A_1 \leq_{\mathrm{T}}^{\mathrm{AC}^0} A_2$ if A_1 is recognized by a family of circuits of polynomial size and constant depth, consisting of NOT gates, unbounded fan-in AND and OR gates, and oracle gates for A_2. (An oracle gate for A_2 takes m inputs $x_1, \ldots, x_m$ and outputs 1 if $x_1 \ldots x_m$ is in A_2, and outputs 0 otherwise.) We write $\mathrm{AC}^0[L]$ for the class of languages A satisfying $A \leq_{\mathrm{T}}^{\mathrm{AC}^0} L$. It is well-known that $\leq_{\mathrm{T}}^{\mathrm{AC}^0}$ is a transitive relation on languages.

Note that $\leq_{\mathrm{m}}^{\mathrm{AC}^0}$ reducibility is a special case of $\leq_{\mathrm{T}}^{\mathrm{AC}^0}$ reducibility.

3 Proof of the Main theorem

For the proof of our main result, we introduce one more number-theoretic function.

Definition 1 *For natural number* j*, let* p_j *denote the* j^{th} *largest odd prime. The function* MULT *takes two integer arguments* x *and* j *where* j *is between 1 and* $|x|$*, and* $\mathrm{MULT}(x, j) = \mathrm{MOD}_{p_j}(x)$*, i.e., it is 1 if* x *is a multiple of* p_j *and is 0 otherwise.*

Using Chinese remaindering and an observation of Boppana and Lagarias we will show:

Lemma 4 $\textsc{Maj} \leq_{\mathrm{T}}^{AC^0} \textsc{mult}$

Our main lemma is:

Lemma 5

1. $\textsc{mult} \leq_{\mathrm{T}}^{AC^0}$ GCD.
2. $\textsc{mult} \leq_{\mathrm{T}}^{AC^0}$ $\textsc{Primes}$.
3. $\textsc{mult} \leq_{\mathrm{T}}^{AC^0}$ $\textsc{Square-Free}$

Theorem 1 follows immediately by combining the two lemmas and the transitivity of $\leq_{\mathrm{T}}^{\mathrm{AC}^0}$ reducibility.

3.1 Proof of Lemma 4

This lemma is mostly a routine application of Chinese remaindering.

Fix n sufficiently large. We want to build a circuit to compute $\textsc{Maj}(x)$ for n bit strings $x = (x_{n-1}, \ldots, x_0)$ using the $\textsc{mult}$ function. By definition, $\textsc{Maj}(x) = \bigvee_{t=\lceil n/2 \rceil}^{n} \textsc{Sum}_t(x)$, where $\textsc{Sum}_t(x) = 1$ if x has exactly t 1's. Thus it suffices for us to show how to compute $\textsc{Sum}_t(x)$ for fixed $t \leq n$.

For integer s and natural number m, write $s(\bmod\ m)$ for the unique integer r between 0 and $m-1$ such that $s-r$ is divisible by m. For natural number j and integer r satisfying $0 \leq r < p_j$ define $M_{j,r}(x)$ to be 1 if $\sum_i x_i (\bmod\ p_j) = r$.

Let $k = \lceil \log n \rceil$. By the prime number theorem, $p_k \sim \log n \log\log n \leq n$ (for n sufficiently large). For integer t and natural number j, let $t_j = t(\bmod\ p_j)$. The Chinese remainder theorem implies that since $p_1 \cdots p_k \geq n$, $\sum_{i=1}^{n} x_i = t$ if and only if $\sum_{i=1}^{n} x_i (\bmod p_j) = t_j$ for $j \leq k$. That is,

$$\textsc{Sum}_t(x) = \bigwedge_{j=1}^{k} M_{j,t_j}(x).$$

Thus it suffices to show that if j and r are integers satisfying $1 \leq j \leq n$ and $0 \leq r < p_j$ then $M_{j,r}(x)$ can be computed (for n-bit strings x) by a polynomial-size $O(1)$-depth circuit that uses $\textsc{mult}$ gates. This is a slight generalization of an observation of Boppana and Lagarias [BL87]. Since p_j is odd, there is some integer exponent $u_j > 0$ such that $2^{u_j} \equiv 1 \pmod{p_j}$. For $x = x_{n-1} \ldots x_0$, let $f_j(x) = x_n 0^{u_j-1} x_{n-1} 0^{u_j-1} \ldots 0^{u_j-1} x_1 0^{u_j-1} x_0$. The integer represented in binary by $f_j(x)$ is

$$f_j(x) = \sum_i 2^{u_j i} x_i.$$

Since

$$\sum_i x_i \equiv \sum_i x_i 2^{u_j i} \pmod{p_j}$$

we have that $M_{j,r}(x) = \textsc{mult}(f_j(x) - r, j)$. This completes the proof of Lemma 4.

3.2 Proof of Lemma 5

We want to reduce the computation of $\textsc{mult}(x, j)$ (where by definition we need consider only the case where $j \leq |x|$) to each of the three given functions. We first note that by Proposition 3(1), we can build a bounded depth circuit of size $poly(n)$ that on input j outputs p_j.

The reduction of MULT to GCD is trivial, since it suffices to compute $\mathrm{GCD}(x, p_j, i)$ for all integers $i \leq n$ to determine whether x is a multiple of p_j.

To reduce MULT to PRIMES we need a fact regarding the distribution of primes. For natural numbers m and l, let $\pi(2^n, m, l)$ denote the number of primes $q \leq 2^n$ such that $q \equiv l \pmod{m}$. It is known that for n sufficiently large, if p is prime and $1 \leq l < p \leq n$, then:

$$\pi(2^n, p, l) \sim \frac{2^n}{n(p-1)\ln 2}. \tag{1}$$

This is easily deduced from Theorems 1 and 2 of [P35] (see also Theorem 7.4 and comments at the beginning of Section 8 of Chapter 4 of [P57].) Thus there is a constant c such that for all large n, for any $1 \leq l \leq p-1$, at least $2^n/cn$ of the numbers having at most n bits that appear in the sequence $l, l+p, l+2p, \ldots$ are prime.

This gives rise to the following probabilistic test to see if an n-bit number x is a multiple of prime $p \leq n$. Given x, choose a random n-bit integer y and a random sign $\varepsilon \in \{-1, 1\}$, and compute $x + \varepsilon py$. (Note that, with probability at least $O(n^{-1})$, $x + \varepsilon py$ is a positive n-bit number.) The test accepts if and only if $x + \varepsilon py$ is prime (which we test using a PRIMES gate) and is not equal to p or $-p$. If x is a multiple of p the test will always reject. If x is not a multiple of p, then equation (1) implies that the test will accept with probability at least $O(n^{-2})$.

Now consider a circuit that performs n^4 independent trials of this test in parallel and takes the OR of the trials. If x is a multiple of p_j, all of the tests reject, whereas if x is not a multiple of p_j, then with probability at least $1 - 1/2^{\omega(n)}$, at least one of the tests will accept. Now, as in the standard argument of [Adl78], there must be at least one sequence of probabilistic inputs for the circuit having the property that, for all n-bit inputs x, the OR of the n^4 tests is equal to $\neg\text{MULT}(x, j)$.

This can be seen to be an $\leq_{\mathrm{T}}^{\mathrm{AC}^0}$ reduction from MULT to PRIMES, since by Proposition 3 (parts 2 and 3), $x + \varepsilon py$ can be computed by bounded depth circuits of size $poly(n)$.

To reduce MULT to SQUARE-FREE, we need an analogue of (1) for square-free numbers. Accordingly for natural number n, prime p and integer l satisfying $1 \leq l \leq p-1$, let $S(n,p,l)$ denote the number of square-free numbers s satisfying $0 \leq s \leq 2^{n-1}$, such that $s \equiv l \pmod{p^2}$. To estimate $S(n,p,l)$, for natural number d, let $T_d(n,p,l)$ be the number of integers s, $0 \leq s \leq 2^n - 1$ such that

$$s \equiv l \pmod{p^2} \qquad \text{and} \qquad s \equiv 0 \pmod{d^2}.$$

By applying the inclusion-exclusion principle we derive that

$$S(n,p,l) = \sum_{1 \leq d \leq 2^{n/2}} \mu(d) T_d(n,p,l),$$

where $\mu(d)$ is the Möbius function, which is defined to be 0 if d is not square free, and otherwise is $(-1)^{\nu(d)}$, where $\nu(d)$ is the number of prime divisors of d. Obviously, $T_d(n,p,l) = 0$ if p divides d and

$$\left| T_d(n,p,l) - \frac{2^n}{p^2 d^2} \right| \leq 1$$

otherwise (because if $\gcd(d,p) = 1$ the above system of congruences defines each such $s \pmod{p^2 d^2}$ uniquely). Therefore

$$S(n,p,l) \;=\; \sum_{\substack{1 \leq d \leq 2^{n/2} \\ \gcd(d,p)=1}} \mu(d) \left(\frac{2^n}{p^2 d^2} + O(1) \right)$$

$$
\begin{aligned}
&= \left(\frac{2^n}{p^2} \sum_{\substack{1 \le d \le 2^{n/2} \\ \gcd(d,p)=1}} \frac{\mu(d)}{d^2} \right) + O(2^{n/2}) \\
&= \frac{2^n}{p^2} \sum_{\substack{d=1 \\ \gcd(d,p)=1}}^{\infty} \frac{\mu(d)}{d^2} + O(2^{n/2}),
\end{aligned}
$$

since we can upper bound the difference of the two sums in the last two expressions by $\sum_{d>2^{n/2}} \frac{1}{d^2} = O(2^{n/2})$. Now, we have:

$$
\begin{aligned}
\sum_{\substack{d=1 \\ \gcd(d,p)=1}}^{\infty} \frac{\mu(d)}{d^2} &= \sum_{d=1}^{\infty} \frac{\mu(d)}{d^2} - \sum_{c=1}^{\infty} \frac{\mu(cp)}{(pc)^2} \\
&= \sum_{d=1}^{\infty} \frac{\mu(d)}{d^2} - \frac{1}{p^2} \sum_{\substack{c=1 \\ \gcd(c,p)=1}}^{\infty} \frac{-\mu(c)}{c^2},
\end{aligned}
$$

which implies:

$$
\sum_{\substack{d=1 \\ \gcd(d,p)=1}}^{\infty} \frac{\mu(d)}{d^2} = \frac{p^2}{p^2-1} \sum_{d=1}^{\infty} \frac{\mu(d)}{d^2}.
$$

Now, it is known (see Theorem 4.4 of [P57]) that for $s > 1$,

$$
\sum_{d=1}^{\infty} \frac{\mu(d)}{d^s} = \frac{1}{\zeta(s)},
$$

where $\zeta(s) = \sum_{n=1}^{\infty} n^{-s}$ is the Riemann ζ-function. Since $\zeta(2) = \pi^2/6$, we conclude:

$$
S(n,p,l) = \gamma(p) 2^n + O(2^{n/2}), \tag{2}
$$

where

$$
\gamma(p) = \frac{6}{\pi^2(p^2-1)}.
$$

Thus (2) provides the desired analogue of (1).

Now, to determine $\text{MULT}(x,j)$ it is enough to check that xp_j is not divisible by p_j^2. Pick a random n-bit number y and $\varepsilon \in \{-1,1\}$ and (using an oracle gate for SQUARE-FREE) check if $p_j x + \varepsilon p_j^2 y$ is square-free. If x is a multiple of p_j, the oracle gate always rejects. If x is not a multiple of p_j, the oracle gate accepts with probability $\gamma(p) + o(1)$. The rest of the argument is analogous to the case of PRIMES.

4 Conclusions and Open Problems

The reduction of $\textsc{mult}(x, j)$ to $\textsc{Primes}$ of the last section is *nonuniform* even for fixed j. That is, we can provide no efficient procedure to *build* the AC^0 circuits that perform the reduction; we can show only that they exist, via a probabilistic argument. Surely it is obvious that telling if a number if a multiple of 3 is no harder than telling if a number is composite! Is there a direct, *uniform* reduction that captures this intuition?

We have seen that, for a fixed odd prime p, the $\textsc{Mod}_p$ problem is reducible to the set of primes, written in base two. A similar argument shows that, for any distinct primes p and q, the $\textsc{Mod}_p$ problem is reducible to the set of primes, written in base q. However, the following question requires a different proof strategy: Is $\textsc{Mod}_2 \leq_{\mathrm{T}}^{\mathrm{AC}^0} \textsc{Primes}$?

The theory of many-one reducibility has been extremely useful in characterizing the complexity of many problems, although it has not turned out to be very useful for studying number-theoretic problems. For example, although we know that $\textsc{Mod}_3$ is AC^0-Turing reducible to $\textsc{Primes}$, we do not know if it is many-one reducible to $\textsc{Primes}$. Might it be possible to *prove* that there is no many-one reduction from $\textsc{Mod}_3$ (or PARITY) to $\textsc{Primes}$? This would show that $\textsc{Primes}$ is not NP-complete (under $\leq_{\mathrm{m}}^{\mathrm{AC}^0}$ reductions), and in fact would show that it is not complete for any familiar complexity class. Although in general it is difficult to show that there *is* no $\leq_{\mathrm{m}}^{\mathrm{AC}^0}$ reduction from one problem to another (since, for example, the NP $\neq$ NC^1 question can be phrased this way), it is worth noting that a set A in NP is presented in [AAIPR97] such that there is no $\leq_{\mathrm{m}}^{\mathrm{AC}^0}$ reduction from PARITY to A.

If $\textsc{Primes}$ were complete for NP (or for any other reasonable complexity class) under $\leq_{\mathrm{m}}^{\mathrm{AC}^0}$ reductions, the isomorphism theorems of [AAR98, AAIPR97] show that $\textsc{Primes}$ would be isomorphic to all of the other complete sets for that class, under isomorphisms computable and invertible by P-uniform depth-three AC^0 circuits. In particular, there would be an isomorphism of this sort between $\textsc{Primes}$ and $\textsc{Primes} \times \{0, 1\}^*$. Among other things, this would yield a fairly "dense" set of primes in P, by looking at the isomorphic image of $\{2\} \times \{0, 1\}^*$. (Observe that it was shown only fairly recently that there is an infinite set of primes in P [PPS89].) Perhaps the existence of such an isomorphism would bestow $\textsc{Primes}$ with some properties that it provably does not have. Perhaps such an isomorphism must involve multiplication (which cannot be computed by AC^0 circuits). That is, perhaps it is possible to prove that PRIMES is not complete for any familiar complexity class. Of course, in the foregoing discussion we are considering only *unconditional* proofs. It is well known, thanks to [Mil76], that PRIMES is in P under the Extended Riemann Hypothesis.

Additional observations and speculations of this sort pertaining to the factoring problem can be found in [All98].

We remark that MULT can be AC^0-reduced to other natural number-theoretic problems and thus these problems are also hard for TC^0. For example, consider the problem of computing the parity of $\omega(x)$, which is the number of distinct prime divisors of $x \in \mathbb{N}$. For any prime p:

$$\textsc{Mod}_p(x) = 0 \quad \Longleftrightarrow \quad \omega(x) + 1 \equiv \omega(px) \pmod 2.$$

Thus MULT (and MAJ) is $\le_{\mathrm{T}}^{\mathrm{AC}^0}$ reducible to the parity of ω.

It would be very interesting to obtain similar results for other number-theoretic problems. For example, it is shown [Shp99] that deciding quadratic residuosity modulo a large prime q is not in AC^0. Note that this question is equivalent to computing the rightmost bit of the discrete logarithm modulo q. It would be very desirable to extend this lower bound to the classes $\mathrm{AC}^0[p]$ and/or TC^0.

Acknowledgment. This paper was essentially written during a visit by the third author to Rutgers University, whose hospitality is gratefully acknowledged.

References

[Adl78] L. Adleman, *Two theorems on random polynomial time*, in "Proc. 19th IEEE Symposium on Foundations of Computer Science", 1978, 75–83.

[AH87] L. Adleman and M.-D. Huang, *Recognizing primes in random polynomial time*, in "Proc. 19th ACM Symposium on Theory of Computing", 1987, 462–469.

[APR83] L. Adleman, C. Pomerance and R. S. Rumely, *On distinguishing prime numbers from composite numbers*, Annals Math. **117** (1987), 173–206.

[Aj83] M. Ajtai, Σ_1^1 *formulae on finite structures*, Annals of Pure and Applied Logic **24** (1983), 1–48.

[AAIPR97] M. Agrawal, E. Allender, R. Impagliazzo, T. Pitassi and S. Rudich, *Reducing the complexity of reductions*, in "Proc. 29th ACM Symposium on Theory of Computing", 1997, 730–738.

[AAR98] M. Agrawal, E. Allender and S. Rudich, *Reductions in circuit complexity: An isomorphism theorem and a gap theorem*, J. Comp. Sys. Sci. **57** (1998), 127–143.

[All98] E. Allender, *News from the isomorphism front*, Computational Complexity Column, Bulletin of the EATCS **66** (1998), 73–82.

[BDS98a] A. Bernasconi, C. Damm and I. E. Shparlinski, *Circuit and decision tree complexity of some number theoretic problems*, Tech. Report 98-21, Dept. of Math. and Comp. Sci., Univ. of Trier, 1998, 1–17.

[BDS98b] A. Bernasconi, C. Damm and I. E. Shparlinski, *On the average sensitivity of testing square-free numbers*, in "Proc. 5th Intern. Computing and Combin. Conf.", Lect. Notes in Comp. Sci., Springer-Verlag, Berlin, (to appear).

[BS99] A. Bernasconi and I. E. Shparlinski, *Circuit complexity of testing square-free numbers*, in "Proc. 16th Intern. Symp. on Theor. Aspects in Comp. Sci.", Lect. Notes in Comp. Sci., Springer-Verlag, Berlin, **1563** (1999), 47–56.

[BL87] R. Boppana and J. Lagarias, *One-way functions and circuit complexity*, Information and Computation **74** (1987), 226–240.

[FSS84] M. Furst, J. Saxe and M. Sipser, *Parity, circuits, and the polynomial-time hierarchy*, Math. Systems Theory **17** (1984), 13–27.

[HB76] J. Hartmanis and L. Berman, *On tape bounds for single letter alphabet language processing*, Theoretical Computer Science **3** (1976), 213–224.

[HS68] J. Hartmanis and H. Shank, *On the recognition of primes by automata*, J ACM **15** (1968), 382–389.

[LV99] R. Lipton and A. Viglas, *On the Circuit Complexity of Primality*, manuscript.

[Man92] S.-G. Mantzivis, *Circuits in bounded arithmetic, 1*, Ann. Math. Artificial Intelligence **6** (1992), 127–156.

[Med91] J. Meidânis, *Lower bounds for arithmetic problems*, Inform. Proc. Letters **38** (1991), 83–87.

[Mil76] G. Miller, *Riemann's hypothesis and tests for primality*, J. Comput. System Sci. **13** (1976), 300–317.

[P35] A. Page, *On the number of primes in an arithmetic progression*, Proc. Lond. Math. Soc. **39** (1935), 116–141.

[PPS89] J. Pintz, W. Steiger and E. Szemerédi, *Two infinite sets of primes with fast primality tests*, Math. Comp. **53** (1989), 399–406.

[P57] K. Prachar, Primzahlverteilung, Springer-Verlag, 1957.

[R80] M. Rabin. *Probabilistic algorithm for primality testing*, Journal of Number Theory **12** (1980), 128–138.

[Raz87] A. A. Razborov, *Lower bounds on the size of bounded depth networks over a complete basis with logical addition*, Mathematicheskie Zametki, **41** (1987), 598–607, English translation in Mathematical Notes of the Academy of Sciences of the USSR **41** (1987), 333–338.

[Shp98] I. E. Shparlinski, *On polynomial representations of Boolean functions related to some number theoretic problems*, Electronic Colloq. on Comp. Compl., TR98-054, 1998, 1–13.

[Shp99] I. E. Shparlinski, *Number theoretic methods in cryptography: Complexity lower bounds*, Birkhäuser, 1999.

[Smo87] R. Smolensky, *Algebraic methods in the theory of lower bounds for Boolean circuit complexity*, in "Proc. 19th ACM Symposium on Theory of Computing", 1987, 77–82.

[SS77] R. Solovay and V. Strassen, *A fast Monte Carlo test for primality*, SIAM J. Comput. **6** (1977), 84–85, erratum **7** (1978), 118.

ECCC ISSN 1433-8092
http://www.eccc.uni-trier.de/eccc
ftp://ftp.eccc.uni-trier.de/pub/eccc
ftpmail@ftp.eccc.uni-trier.de, subject 'help eccc'