A Note on Uniform Circuit Lower Bounds for the Counting Hierarchy (Extended Abstract)[⋆]

Eric Allender[⋆⋆]

Department of Computer Science
Rutgers University
P.O. Box 1179
Piscataway, NJ 08855-1179
`allender@cs.rutgers.edu`.

Abstract. A very recent paper by Caussinus, McKenzie, Thérien, and Vollmer [CMTV95] shows that ACC^0 is properly contained in ModPH, and TC^0 is properly contained in the counting hierarchy. Thus, [CMTV95] shows that there are problems in ModPH that require superpolynomial-size uniform ACC^0 circuits, and problems in the counting hierarchy that require superpolynomial-size uniform TC^0 circuits. The proof in [CMTV95] uses "leaf languages" as a tool in obtaining their separations, and their proof does not immediately yield larger lower bounds for the complexity of these problems. In this paper, we give a simple direct proof of these same separations, and use it to provide "sub-subexponential" size lower bounds on the size of uniform circuits for these problems.

1 Introduction

The central problem in complexity theory is the task of proving lower bounds on the complexity of specific problems. Circuit complexity, in particular the study of constant-depth circuits, is one of the (few) areas where complexity theory has succeeded in actually providing lower bounds, and even in the study of constant-depth circuits one quickly arrives at the limits of current lower-bound technology. It is known that constant-depth circuits of AND, OR, and NOT gates (so-called AC^0 circuits) require exponential size even to compute the parity of n input bits [Hå87, Ya85], and similar lower bounds are known for constant-depth circuits of AND, OR, NOT, and MODp gates where p is prime [Ra87, Sm87]. When MODm gates are allowed for composite m, however, almost nothing is known. It remains an open question if there is any problem in $\mathrm{NTIME}(2^{n^{O(1)}})$ that cannot be done with polynomial size and constant depth with AND and MOD6 gates.

There is considerable reason to be interested in circuits with AND, OR, and MODm gates; circuits of this sort are called ACC^0 circuits (for "Alternating

[⋆] To appear in Proc. 2nd International Computing and Combinatorics Conference (COCOON '96).

[⋆⋆] Supported in part by NSF grant CCR-9509603.

Circuits with Counters"; the superscript 0 refers to the fact that we are considering circuits of depth $O(\log^0 n)$.). The lovely result of [Ba89] characterizing NC^1 (log-depth fan-in two circuits) in terms of constant-width branching programs relies heavily on algebraic techniques, and shows that NC^1 corresponds to computation over *non-solvable* algebras. Barrington also defined the corresponding notion of computation over *solvable* algebras, and it is shown in [BT88] that this notion corresponds exactly to ACC^0 circuits. To restate these two points:

1. The results of [Ba89] establish intimate connections between circuit complexity and algebraic structure.
2. In this algebraic setting, ACC^0 is the most important subclass of NC^1.

Although, as mentioned above, it is unknown if small ACC^0 circuits suffice to compute all problems in NEXPTIME, lower bounds for *uniform* ACC^0 circuits were presented in [AG94]. Since our results, like those of [CMTV95] and [AG94], concern uniform circuits, it is necessary to briefly discuss uniformity.

A circuit family $\{C_n\}$ consists of a circuit for each input length n. If C_n is "sufficiently easy" to construct from n, then the family $\{C_n\}$ is said to be *uniform*. Different notions of "sufficiently easy" give rise to different notions of uniformity, and the question of which notion of uniformity is the "right" one to use when studying classes of circuits is not always clear. For the circuit classes considered here, convincing arguments are presented in [BIS90], arguing that a very restrictive notion of uniformity called *Dlogtime-uniformity* is the correct notion to use. Briefly, a circuit family $\{C_n\}$ is Dlogtime-uniform if, given n, g, h, a deterministic Turing machine can, in time $O(|n, g, h|)$, determine if gate g is connected to gate h in circuit C_n, and determine what sort of gates g and h are. The name "Dlogtime-uniformity" comes from the fact that the length of the input n, g, h is logarithmic in the size of the circuit C_n. Throughout the rest of this paper, all mention of uniform circuits refers to Dlogtime-uniform circuits.

Throughout the rest of this paper, $\mathrm{ACC}^0(S(n))$ will denote the class of languages with *uniform* ACC^0 circuits of size $S(n)$. ACC^0 denotes $\mathrm{ACC}^0(n^{O(1)})$.

In contrast to our lack of lower bounds for nonuniform ACC^0 circuits for sets in $\mathrm{NTIME}(2^{n^{O(1)}})$, it was shown in [AG94] that exponential size (i.e., size at least 2^{n^ϵ}) is required to compute the permanent (and other problems complete for #P) on *uniform* ACC^0 circuits. Thus there are sets in $\mathrm{P}^{\#\mathrm{P}}$ that require exponential-sized uniform ACC^0 circuits.

The complexity class PP is closely related to #P (for instance, $\mathrm{P}^{\#\mathrm{P}} = \mathrm{P}^{\mathrm{PP}}$), and one might expect that similar exponential lower bounds would hold there, but [AG94] was able only to show that sets complete for these classes require more than "sub-subexponential" size ACC^0 circuits, where a function t is said to be sub-subexponential if $t(t(n)) = 2^{n^{o(1)}}$. (Note that for all "natural" and interesting size bounds t, t is subexponential according to this definition if and only if for each k, $t(t(n)^k) = 2^{n^{o(1)}}$. For the rest of this paper, this will be the definition of "sub-subexponential". Observe that size bounds such as $2^{\log^k n}$ and $2^{(\log n)^{k \log\log n}}$ are sub-subexponential.)

Another class of constant-depth circuits that has attracted interest uses threshold (or MAJORITY) gates instead of counters. Let $\mathrm{TC}^0(S(n))$ denote the class of sets accepted by uniform constant-depth threshold circuits of size $S(n)$; TC^0 will denote $\mathrm{TC}^0(n^{O(1)})$. It is easy to observe that $\mathrm{ACC}^0 \subseteq \mathrm{TC}^0$, and thus we have even fewer lower bounds for the threshold circuit model than for ACC^0 circuits. It is an easy consequence of the space hierarchy theorem that PSPACE-complete sets require exponential size uniform TC^0 circuits, but there is still no smaller complexity class in PSPACE that is known to require exponential-size TC^0 circuits.

There are well-studied subclasses of PSPACE that correspond in a natural way to the complexity classes AC^0, ACC^0, and TC^0. The relationship between the polynomial hierarchy and AC^0 is well-known and was established by [FSS84]. One way to present this correspondence is to observe that, when one considers alternating Turing machines that make only $O(1)$ alternations, a polynomial running time yields the polynomial hierarchy, while a logarithmic running time yields uniform AC^0. The analogous subclasses of PSPACE corresponding to ACC^0 and TC^0 are ModPH, and the counting hierarchy, respectively.

ModPH is in some sense a generalization of the polynomial hierarchy and of $\oplus\mathrm{P}$ (formal definitions appear in the next section). The counting hierarchy (defined in [Wa86] and studied by several authors) consists of the union of the complexity classes PP, $\mathrm{PP}^{\mathrm{PP}}$, $\mathrm{PP}^{\mathrm{PP}^{\mathrm{PP}}}$, ... In the next section, we present models of computation (similar to alternating Turing machines) such that polynomial time on this model characterizes ModPH (or the counting hierarchy) while logarithmic time characterizes ACC^0 (or TC^0, respectively).

A very recent paper by Caussinus, McKenzie, Thérien, and Vollmer [CMTV95] shows that ACC^0 is properly contained in ModPH, and TC^0 is properly contained in the counting hierarchy. The proof given by [CMTV95] uses "leaf languages" as a tool, and does not explicitly present a lower bound for any language in ModPH or in the counting hierarchy. The present work began as an attempt to discover if these techniques could be used to present an explicit lower bound. Unfortunately, this attempt did not succeed. For each given language A in ModPH (or in the counting hierarchy) it is still an open question if A has polynomial size uniform ACC^0 circuits (threshold circuits, respectively).

On the other hand, this paper does give a very simple direct proof of the separations presented in [CMTV95], and shows that for every sub-subexponential function t *there exist* sets A in ModPH (or in the counting hierarchy) requiring size greater than $t(n)$ to compute on uniform ACC^0 circuits (threshold circuits, respectively).

2 Machine Models

We assume the reader is familiar with nondeterministic oracle Turing machines. Given natural number m and oracle A, $\mathrm{Mod}_m\mathrm{P}^A$ is the class of languages B such that, for some nondeterministic polynomial-time Turing machine M, x is in B if and only if the number of accepting computations of M^A on input x is a multiple of m. Then the class ModPH is defined to be the smallest class of languages containing P and with the property that if A is in ModPH, then so are NP^A and Mod_m^A for every natural m. ModPH has been studied by several authors.

It is useful to have a model of computation characterizing ACC^0 and ModPH, in the same way that alternating Turing machines characterize both AC^0 and the polynomial hierarchy. The appropriate model of computation was defined in [AG94] as a variant of alternating Turing machines. For the purposes of this extended abstract, we will not present the detailed definitions, but the reader can probably guess what is meant by augmenting the usual existential and universal states of an alternating Turing machine with Mod_m states. Details can be found in [AG94].

Let a *signature* σ be a finite string from $\{\forall, \exists, \mathrm{Mod}_2, \mathrm{Mod}_3, \mathrm{Mod}_4, \ldots\}^*$. For any alternating Turing machine making $O(1)$ alternations, each path in the alternating tree of the machine on any input x has a signature given by the sequence of types of states the machine enters. If M is an alternating machine such that on all inputs x, all paths have the same signature σ, then M is said to be a σ machine. For instance, the signature of a Σ_2 machine is $\forall\exists$, and the signature of a machine accepting a language in $\mathrm{NP}^{\oplus\mathrm{P}^{\mathrm{Mod}_7\mathrm{P}}}$ is $\exists\mathrm{Mod}_2\mathrm{Mod}_7$. Let $\sigma\mathrm{time}(t(n))$ denote the class of languages accepted by σ machines running in time $t(n)$. The technical lemmas in [AG94] essentially prove the following proposition.

Definition 1. Let us call a function f *constructible* if $f(n) = 2^{g(n)}$, where $g(n)$ can be computed from n (in binary) in time $O(g(n))$.

Proposition 2. *Let $t(n)$ be a constructible function, $t(n) = \Omega(\log n)$. Then Uniform $\mathrm{ACC}^0(2^{O(t(n))}) = \bigcup_\sigma \sigma time(O(t(n)))$.*

It will turn out to be useful to us to note that a "tape reduction theorem" holds for σ machines; if a set is accepted in time $t(n)$ by a σ machine with k worktapes, then it is also accepted in time $O(t(n))$ be a σ machine with two worktapes. (*Proof sketch:* Given a k-tape σ machine, follow the construction in [AG94] and build an ACC circuit, such that σ is the sequence of types of gates encountered in a root-to-leaf path. In the construction given in [AG94], the deterministic linear-time machine that checks the uniformity condition needs k tapes. However, by changing the naming convention for the gates in the circuit in a way that makes use of the ideas in the original tape-reduction proof of [BG70] for nondeterministic machines, we can make do with a two-tape deterministic machine checking the uniformity condition. Now given a uniform σ-circuit family where the uniformity condition is checked by a 2-tape machine, the construction in [AG94] yields a two-tape σ-machine accepting the original language.)

Similarly, we will find it very convenient to have a single model of computation that is sufficient for describing both TC^0 and the counting hierarchy. Fortunately, such a model was described in [PS88]. In their model, which they call a "threshold Turing machine", TC^0 corresponds to $O(\log n)$ time and $O(1)$ uses of the "threshold" operation, and the counting hierarchy corresponds to polynomial time and $O(1)$ uses of the threshold operation. The characterization of the counting hierarchy in terms of threshold Turing machines is given in [PS88], but the corresponding characterization of TC^0 is *not* presented there (since [PS88] predates the uniformity considerations of [BIS90]), and it also does not seem to have been published anywhere else. Although [BIS90] *does* give many equivalent characterizations of TC^0, the threshold Turing machine model is not mentioned in [BIS90]. Nonetheless, the proof of the following proposition is quite standard and follows along the lines of related results in [PS88, BIS90]:

Proposition 3. *Let $t(n)$ be a constructible function, $t(n) = \Omega(\log n)$. Then Uniform threshold circuit depth($O(1)$), size($2^{O(t(n))}$) = Threshold Turing machine time($O(t(n))$), thresholds($O(1)$).*

As is the case with the σ machines considered above, the Threshold Turing machines also enjoy a tape-reduction property, proved in essentially the same way. If a set is accepted in time $t(n)$ by a k-tape Threshold Turing machine, then it is accepted in time $O(t(n))$ by a Threshold Turing machine with two tapes.

The tape-reduction properties are useful in diagonalization arguments.

3 Diagonalization

It is important to note that the techniques used to prove the nondeterministic time hierarchy (originally proved in [SFM78], although we will use the very simple and general version proved by Žák [Z83]) can be used to prove analogous hierarchies for other computational models defined in terms of nondeterministic Turing machines (with a fixed bound on the number of worktapes). In particular, an essentially word-for-word translation of the proof in [Z83] shows the following.

Theorem 4. *Let 2^T be constructible. Then there is a set B in $\sigma time(T(n))$ such that, for all t with $t(n+1) = o(T(n))$, B is not in $\sigma time(t(n))$. Also, there is a set in D in Threshold Turing machine time($O(T(n))$),thresholds(k) such that, for all t with $t(n+1) = o(T(n))$, B is not in Threshold Turing machine time($O(t(n))$),thresholds(k).*

Proof: For completeness, we present the main outline of the proof. Let $M_1, M_2, \ldots$ be an enumeration of 2-tape σ-machines (threshold machines, respectively). Let f be a rapidly-growing function such time $T(f(i,n,s))$ is enough time for a *deterministic* machine to compute the function

$$(i,n,s) \mapsto \begin{cases} 1 \text{ if } M_i \text{ accepts } 1^n \text{ in } \le s \text{ steps} \\ 0 \text{ otherwise} \end{cases}$$

(Letting $f(i,n,s)$ be greater than $T^{-1}(2^{2^{i+n+s}})$ is sufficient; note that it is important in our setting to handle *sublinear* functions T.)

Now divide Σ^* into regions, so that in region $j = (i,y)$, we diagonalize against machine M_i, thus ensuring that each machine is considered infinitely often. The regions are defined by functions $\mathit{start}(j)$ and $\mathit{end}(j)$, defined as follows: $\mathit{start}(1) = 1$, $\mathit{start}(j+1) = \mathit{end}(j)+1$, where $\mathit{end}(j) = f(i, \mathit{start}(j), T(\mathit{start}(j)-1))$ (where $j = (i, y)$). The important point is that, on input $1^{\mathit{end}(j)}$, a deterministic machine can, in time T, determine whether M_i accepts $1^{\mathit{start}(j)}$ in $\leq T(\mathit{start}(j)-1)$ steps.

By picking f appropriately easy to invert, we can guarantee that, on input 1^n, we can in time $T(n)$ determine which region j contains n.

Now it is easy to verify that the following routine can be computed in time $T(n)$ by a σ-machine (or a threshold machine, respectively). (In the pseudo-code below, U is a "universal" σ-machine (or threshold machine) with 4 tapes which is therefore able to simulate one step of machine M_i in about i^3 steps.)

1. On input 1^n, determine which region j contains n. Let $j = (i, y)$.
2. If $n = \mathit{end}(j)$, then accept iff M_i does *not* accept $1^{\mathit{start}(j)}$ in $\leq T(\mathit{start}(j)-1)$ steps.
3. Otherwise, accept iff U accepts $(i, 1^{n+1})$ in $\leq T(n)$ steps. (Here, it is important that we are talking about $T(n)$ steps of U, which may be only about $T(n)/i^3$ steps of M_i.)

Let us call the set defined by the preceding pseudo-code A. Clearly, A is in σtime$(T(n))$. We now claim that is is not in σtime$(t(n))$.

Assume otherwise, and let M_i be the σ machine accepting A in time $t(n)$. Let c be a a constant such that $i^3 t(n+1) < T(n)$ for all $n \geq c$. Let y be a string of length $\geq c$, and consider stage $j = (i, y)$. Then for all n such that $\mathit{start}(j) \leq n < \mathit{end}(j)$, we have $1^n \in A$ iff $1^{n+1} \in A$. However this contradicts the fact that $1^{\mathit{start}(j)} \in A$ iff $1^{\mathit{end}(j)} \notin A$. □

4 Main Result

Once the definitions are in hand, the proof is now quite straightforward.

Theorem 5. *Let t be a sub-subexponential constructible function. Then there exist sets A in* ModPH *requiring size greater than $t(n)$ to compute on uniform* ACC^0 *circuits.*

Proof: Let t be given. Let C be a set complete for P under Dlogtime-uniform projections. (For instance, the standard complete set $\{(i, x, 0^j) : M_i$ accepts x in time $j\}$ is a good choice for C.)

The proof consists of two cases.

Case 1: [C requires size greater than $t(n)$ to compute on uniform ACC^0 circuits.] In this case, of course there is nothing to prove.

Case 2: [C can be computed by uniform ACC^0 circuits of size $t(n)$.]

Since t is constructible, let g be the function such that $t(n) = 2^{g(n)}$.

In this case, it must happen that there is some σ such that ACC^0 is in σtime$(g(n^{O(1)}))$, because uniform circuits for any set reducible to C can easily be constructed from the circuits for C.

Now standard translational techniques can be used to show that for any signature τ, τtime$(g(n))$ is contained in σtime$(g(t(n)^{O(1)}))$. To see this, consider any language A in τtime$(g(n))$. Let $A' = \{x10^j : j + |x| + 1 = t(|x|) \text{ and } x \in A\}$. Our constructibility assumptions on t assure that A is in ACC^0, and hence is in σtime$(g(n^l))$ for some l. Let M be this $g(n^l)$-time-bounded σ machine accepting A'. The σ machine M' that, on input x, simulates M on input $x10^{t(|x|)-|x|-1}$ runs in time $g(t(n)^l)$.

Since t is sub-subexponential, $2^{n^\epsilon} > t(t(n)^l) = 2^{g(t(n)^l)}$ and thus $g(t(n)^l) = o(n)$. Thus it follows from Theorem 4 that there is a set B in σtime(n) (and hence in ModPH) such that, for all l, B is not in σtime$(g(t(n^l)))$, and thus B is not in τtime$(g(n))$ and thus B does not have uniform ACC^0 circuits of size $t(n)$. $\square$

It is important to note that, because of the nonconstructive nature of the proof of this theorem, the proof offers no clue as to *what* set in ModPH has large ACC^0 circuits.

An essentially identical proof yields the following theorem.

Theorem 6. *Let t be a sub-subexponential constructible function. Then there exist sets A in the counting hierarchy requiring size greater than $t(n)$ to compute on uniform threshold circuits.*

5 More Separations

Rather than asking for the *largest* lower bounds that one can prove for sets in ModPH or in the counting hierarchy, one can ask the dual question: what is the *smallest* class that one can separate from ACC^0 (or TC^0)? That is, [CMTV95] shows that ModPH is not equal to ACC^0; is there a *smaller* class than ModPH that we can show is not equal to ACC^0? This section gives an affirmative answer.

First we make a simple observation:

Proposition 7. *For all $\epsilon > 0$,* ACC^0 *is properly contained in*

$$(\mathit{DTIME}(n^\epsilon) \cup \bigcup_\sigma \sigma\mathit{time}(\log n \log^* n)).$$

Proof: By standard padding methods, it is easy to construct a set A that is complete for P in DTIME(n^ϵ) under projections. This set A is thus also hard for ACC^0 under projections. If A is not in ACC^0 then this yields the desired conclusion.

Otherwise A is in ACC^0 and is therefore in σtime$(O(\log n))$ for some σ. Since σtime$(O(\log n))$ is closed under projections, it follows that ACC^0 is equal to σtime$(O(\log n))$. By diagonalization, we obtain that ACC^0 is properly contained in σtime$(O(\log n \log^* n))$. $\square$

An identical proof yields

Proposition 8. *For all $\epsilon > 0$,* TC^0 *is properly contained in*

$$(\mathit{DTIME}(n^\epsilon) \cup \mathrm{TC}^0(n^{O(\log^* n)})).$$

(Note that one can replace DTIME(n^ϵ) with the potentially smaller class ATIME($\epsilon \log n$), where ATIME denotes alternating Turing machine time; there is a complete problem for NC^1 in ATIME($\epsilon \log n$), for each $\epsilon > 0$.)

We immediately get the following corollaries, which seem only marginally better than the results of [CMTV95] showing proper inclustion in ModPH and the counting hierarchy:

Corollary 9. *Let ϵ be greater than 0. Then:*

$$\mathrm{ACC}^0 \textit{ is properly contained in } \mathrm{ACC}^0(2^{n^\epsilon}).$$
$$\mathrm{TC}^0 \textit{ is properly contained in } \mathrm{TC}^0(2^{n^\epsilon}).$$

But now we will use the technique of [ABHH] to get a better separation.

Lemma 10. *Let S be a constructible function, $S(n) \geq n$.*
If $\mathrm{ACC}^0 = \mathrm{ACC}^0(S(n))$, *then* $\mathrm{ACC}^0 = \mathrm{ACC}^0(S(S(n)))$.

Proof: Let A be any set in σtime($O(\log S(S(n)))$). Since a constructible function $S(n)$ is of the form $2^{g(n)}$, this means that A is in σtime($O(g(S(n)))$). Let A' be the padded version $\{x10^{S(|x|)-|x|-1} : x \in A\}$. Our assumption implies that A' is in ACC^0, and thus is in σ'time($O(\log n)$) for some σ'. This in turn implies that A is in σ'time($O(\log(S(n)))$), and thus by assumption A is in ACC^0. □

Corollary 11. *Let T be a constructible function such that, for some k, $T^{(k)}(n) > 2^n$, where $T^{(k)}$ is T composed with itself k times. Then*

$$\mathrm{ACC}^0 \textit{ is properly contained in } \mathrm{ACC}^0(T(n)).$$

Corollary 12. *Let T be a constructible function such that, for some k, $T^{(k)}(n) > 2^n$. Then*

$$\mathrm{TC}^0 \textit{ is properly contained in } \mathrm{TC}^0(T(n)).$$

6 Conclusions and Open Problems

It is often harder to ask the right question than to answer that question. In [AG94] we presented lower bounds on the uniform circuit complexity of certain problems in PSPACE, and did not see any way to prove lower bounds on the ACC^0 circuit complexity of any given problem in ModPH. Given the inspiration of [CMTV95], it is easy to give a direct proof showing that there *exist* sets in ModPH having large ACC^0 circuit complexity, without giving lower bounds on any specific set in ModPH.

An obvious question is whether the sub-subexponential lower bounds given here and in [AG94] can be improved to exponential lower bounds. Of course, an even more desirable step would be to prove directly that MAJORITY requires exponential size for ACC^0 circuits. (The "natural proofs" framework of [RR94] indicates that many lower bound proofs may be quite difficult to obtain. However, since ACC^0 is a very limited class in many respects (and in particular it is not clear that one should expect pseudorandom generators to be computable in ACC^0), it it not clear that lower bounds for ACC^0 should be hard to obtain.)

Acknowledgments

I thank the authors of [CMTV95] for making their manuscript available to me.

References

[ABHH] E. Allender, Richard Beigel, Ulrich Hertrampf, and Steven Homer, *Almost-everywhere complexity hierarchies for nondeterministic time*, Theoretical Computer Science 115 (1993) 225–242.

[AG94] E. Allender and Vivek Gore, *A uniform circuit lower bound for the permanent*, SIAM Journal on Computing 23 (1994) 1026–1049.

[Ba89] D. A. Barrington, *Bounded-width polynomial-size branching programs recognize exactly those languages in NC^1*, J. Comput. and System Sci. 38, 150–164.

[BIS90] D. A. Mix Barrington, N. Immerman, and H. Straubing, *On uniformity within* NC^1, Journal of Computer and System Sciences 41, 274–306.

[BT88] D. Barrington and D. Thérien, *Finite monoids and the fine structure of NC^1*, J. Assoc. Comput. Mach., *35* (1988), pp. 941–952.

[BG70] R. Book and S. Greibach, *Quasi-realtime languages*, Mathematical Systems Theory 4 (1970) 97–111.

[CMTV95] H. Caussinus, P. McKenzie, D. Thérien, H. Vollmer, *Nondeterministic NC^1 computation*, to appear in Proc. 11th Annual IEEE Conference on Computational Complexity, 1996.

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

[Hå87] J. Håstad, Computational Limitations for Small Depth Circuits, MIT Press, Cambridge, MA, 1987.

[PS88] I. Parberry and G. Schnitger, *Parallel computation with threshold functions*, J. Computer and System Science **36**, 278–302.

[Ra87] A. A. Razborov, *Lower bounds on the size of bounded depth networks over a complete basis with logical addition*, Mathematicheskie Zametki **41(4)**, 598–607. English translation in Mathematical Notes of the Academy of Sciences of the USSR **41:4**, 333-338.

[RR94] A. Razborov and S. Rudich, *Natural proofs*, in Proc. 26th Annual Symposium on Theory of Computing, 1994, pp. 204–213.

[SFM78] J. Seiferas, M. Fischer, and A. Meyer, Separating nondeterministic time complexity classes, *J. ACM* **25** (1978) 146–167.

[Sm87] R. Smolensky, *Algebraic methods in the theory of lower bounds for Boolean circuit complexity*, Proc. 19th ACM Symposium on Theory of Computing, pp. 77–82.

[Wa86] K. W. Wagner, *The complexity of combinatorial problems with succinct input representation*, Acta Informatica **23**, 325–356.

[Ya85] A. C. Yao, *Separating the polynomial-time hierarchy by oracles*, Proc. 26th IEEE Symposium on Foundations of Computer Science, pp. 1–10.

[Z83] S. Žák, A Turing machine hierarchy, *Theoretical Computer Science* **26** (1983) 327–333.

This article was processed using the LaTeX macro package with LLNCS style