Time-Space Tradeoffs in the Counting Hierarchy

ERIC ALLENDER [*]
Dept. of Computer Science
Rutgers University
allender@cs.rutgers.edu

MICHAL KOUCKÝ [†]
Dept. of Computer Science
Rutgers University
mkoucky@paul.rutgers.edu

DETLEF RONNEBURGER [‡]
Dept. of Computer Science
Rutgers University
detlef@paul.rutgers.edu

SAMBUDDHA ROY
Dept. of Computer Science
Rutgers University
samroy@paul.rutgers.edu

V VINAY
Dept. of Computer Science and Automation
Indian Institute of Science
vinay@csa.iisc.ernet.in

Abstract

We extend the lower bound techniques of [14], to the unbounded-error probabilistic model. A key step in the argument is a generalization of Nepomnjaščiĭ's theorem from the Boolean setting to the arithmetic setting. This generalization is made possible, due to the recent discovery of logspace-uniform TC^0 *circuits for iterated multiplication [9].*

Here is an example of the sort of lower bounds that we obtain: we show that MAJ·MAJSAT is not contained in $\mathrm{PrTiSp}(n^{1+o(1)}, n^{\epsilon})$ *for any* $\epsilon < 1$. *We also extend a lower bound of [14], from showing that* $\overline{\mathrm{SAT}}$ *does not have uniform NC*^{1} *circuits of size* $n^{1+o(1)}$, *to a similar result for SAC*^{1} *circuits.*

1. Introduction

This work takes as its starting point the lower bounds presented by Fortnow in [14], in which it was shown that SAT $\notin \mathrm{DTiSp}(n^{1+o(1)}, n^{1-\epsilon})$, and $\overline{\mathrm{SAT}} \notin \mathrm{NTiSp}(n^{1+o(1)}, n^{1-\epsilon})$.

The time-space tradeoffs of [14] have been extended in several ways. Lipton and Viglas [19] and Fortnow and van Melkebeek [15] showed that better lower bounds on time could be obtained, if smaller space bounds were considered. For instance, it is shown in [15] that the set $\overline{\mathrm{SAT}} \notin \mathrm{NTiSp}(n^{1.4}, n^{o(1)})$. Tourlakis [26] extended results of [14] and [19] further to a nonuniform setting where machines can take $n^{o(1)}$ advice. A survey of these developments can be found in [20].

More dramatically, results in the branching program model proved by Ajtai [1] and extended by Beame *et al.* [7] show that deterministic time-space tradeoffs that are nearly as strong can be obtained even for problems of much lower complexity than SAT (i.e., problems in P), and even in the fully nonuniform setting. For instance, specific instances of combinatorial problems in P are presented, that require time $\Omega(n\sqrt{\log n/\log\log n})$ on machines using space $O(n^{1-\epsilon})$. (Van Melkebeek points out in [20] that the results of [7] seem to imply nothing about the complexity of SAT, since the problems in P considered in that work are not known to be reducible to SAT in time $n \log^{O(1)} n$.)

We extend the time-space tradeoffs in yet another direction. Instead of deterministic or nondeterministic computation, we consider the seemingly much more powerful mode of unbounded-error probabilistic computation. Note that the class $\mathrm{PrTiSp}(n \log n, n^{\epsilon})$ is very powerful. It contains problems complete for PP under $\leq_m^p$ reducibility, and hence, by [24], it contains problems hard for the polynomial hierarchy under $\leq_T^p$ reductions.

Just as the techniques of [14] do not yield lower bounds for SAT in the nondeterministic time- and space-bounded model (instead yielding results only for $\overline{\mathrm{SAT}}$), our extensions do not yield lower bounds for SAT or MAJSAT, but only for problems complete for the second level of the

[*] Supported in part by NSF grant CCR-9734918.
[†] Supported in part by NSF grant CCR-9734918.
[‡] Supported in part by NSF grant CCR-9734918.

counting hierarchy.

We had hoped to provide lower bounds on the size and depth of uniform arithmetic circuits computing functions in the counting hierarchy, but we do not see how do do this. However, we are able to extend the circuit lower bounds of [14] to a larger class of circuits. Fortnow shows in [14] that SAT (and hence $\overline{\text{SAT}}$) cannot be solved by logspace-uniform NC^1 circuits of size $n^{1+o(1)}$. A slight modification of that proof shows that $\overline{\text{SAT}}$ cannot be solved by uniform SAC^1 circuits of size $n^{1+o(1)}$. (We sketch a proof in Section 4.)

Our main result (Theorem 3) can be extended along the lines of Tourlakis ([26]) to show that $\mathrm{PrTime}(n)^{\mathrm{PrTime}(n)/n^{o(1)}} \not\subseteq \mathrm{PrTiSp}(n^{1+\gamma}, n^{\epsilon})/n^{o(1)}$. We omit the details of this extension.

2 Preliminaries

We assume that the reader is familiar with the standard notions of complexity theory as can be found in the standard textbooks (i.e. [5, 22, 11, 28]). In particular, the reader is assumed to be familiar with unbounded-error probabilistic oracle Turing machines.

In this paper $\mathrm{PrTime}^A(t)$ denotes the class of languages decidable by unbounded-error probabilistic Turing machines running in time t, with oracle A. If $\mathcal{C}$ is a class of languages, then $\mathrm{PrTime}^{\mathcal{C}}(t)$ is the union, over all $A \in \mathcal{C}$, of $\mathrm{PrTime}^A(t)$. We will use $\mathrm{PrTiSp}(t(n), s(n))$ to denote the class of languages decidable by an unbounded-error probabilistic Turing machine running in time $O(t(n))$ and space $O(s(n))$.

We define the levels of the linear time version of the counting hierarchy as follows.

Definition 1 *Given the first level of the linear time counting hierarchy relative to oracle A, $\mathrm{LCH}_1^A = \mathrm{PrTime}^A(n)$, we recursively define the higher levels as $\mathrm{LCH}_{i+1}^A = \mathrm{PrTime}(n)^{\mathrm{LCH}_i^A}$. The union over all levels gives the entire hierarchy $\mathrm{LCH}^A = \bigcup_i \mathrm{LCH}_i^A$. The unrelativized levels of this hierarchy are defined as usual by using the empty set as an oracle: $\mathrm{LCH}_i = \mathrm{LCH}_i^{\emptyset}, \mathrm{LCH} = \mathrm{LCH}^{\emptyset}$.*

It can be verified that LCH is a relatively robust class that can be defined equivalently using the linear-time analogs of #P or $\text{C}_=\text{P}$, etc.

In order to prove our main result we will need the following theorem, which can be proven by standard diagonalization techniques using the fact that a multi-tape probabilistic Turing machine running in time t can be simulated on a two-tape probabilistic Turing machine in time $O(t)$ [8, 2].

Theorem 2 (Hierarchy Theorem) *Let T and t be time-constructible functions. If $t \in o(T)$ then $\mathrm{PrTime}(t) \subsetneq \mathrm{PrTime}(T)$.*

As usual, TC^0 denotes the class of languages decidable by constant depth, polynomial-size threshold circuits, i.e., Boolean circuits that consist of unbounded fan-in AND, OR, NOT and MAJORITY gates. We will also use arithmetic circuits over the integers consisting of unbounded fan-in $+$ and $\times$ gates (usual arithmetic addition and multiplication).

Further recall that SAC^1 is the class of languages accepted by polynomial-size circuits of depth $O(\log n)$ having unbounded fan-in OR gates and bounded fan-in AND gates; it is equal to LogCFL: the class of problems logspace-reducible to a context-free language [27]. Since LogCFL is closed under complement [4], it can also be defined in terms of bounded fan-in OR gates and unbounded fan-in AND gates. Recall also that $\text{NC}^1 \subseteq \text{L} \subseteq \text{NL} \subseteq \text{SAC}^1$.

3 Main Result

Our main result extends the time-space tradeoff of [14], where instead of giving lower bounds for solving $\overline{\text{SAT}}$ with nondeterministic machines, we give lower bounds for solving MAJ·MAJSAT with probabilistic machines. We will give the following relation between the time and space required to solve problems from the second level of the linear time counting hierarchy. We show in Section 3.2 that this implies a lower bound on MAJ·MAJSAT.

Theorem 3 (Main Theorem) *For every constant ϵ such that $0 < \epsilon < 1$, there exists a constant $\gamma > 0$, such that*

$$\mathrm{PrTime}(n)^{\mathrm{PrTime}(n)} \not\subseteq \mathrm{PrTiSp}(n^{1+\gamma}, n^{\epsilon}).$$

More specifically, we show the next theorem, which implies our main result by setting $\gamma = \min\{a, d\} - 1$. It would be possible to determine more precisely the relationship among γ, a and d, by examining more closely certain factors such as the depth of uniform TC^0 circuits for iterated product. We do not attempt such an analysis here.

Theorem 4 *For every $0 < \epsilon < 1$, and every $a > 1$, there exists a constant $d > 1$ such that if*

$$\mathrm{PrTime}(n) \subseteq \mathrm{PrTiSp}(n^a, n^{\epsilon})$$

then

$$\mathrm{PrTime}(n)^{\mathrm{PrTime}(n)} \not\subseteq \mathrm{PrTime}(n^d).$$

The proof of this theorem is based on the following two lemmas. The first one is an analog of Nepomnjaščiǐ's Theorem [21]. We will prove it in Section 3.1.

Lemma 5 *For any $\alpha > 1$, and any $0 < \beta < 1$, there is a constant $\tau(\alpha, \beta) > 0$ such that:*

$$\mathrm{PrTiSp}(n^{\alpha}, n^{1-\beta}) \in \mathrm{LCH}_{\tau(\alpha,\beta)}.$$

Lemma 6 *For any rational $d > 1$, and any integer constant $k \geq 1$, if*

$$\mathrm{PrTime}(n)^{\mathrm{PrTime}(n)} \subseteq \mathrm{PrTime}(n^d)$$

then

$$\mathrm{LCH}_k \subseteq \mathrm{PrTime}(n^{d^{k-1}}).$$

Proof of Lemma 6: For $k < 3$, the lemma is trivial, so assume that $k \geq 3$. The assumption $\mathrm{PrTime}(n)^{\mathrm{PrTime}(n)} \subseteq \mathrm{PrTime}(n^d)$ implies by a padding argument on the oracle that $\mathrm{PrTime}(n)^{\mathrm{PrTime}(n^{d^i})} \subseteq \mathrm{PrTime}(n^{d^{i+1}})$, for any $i \geq 1$. Therefore, for any $0 \leq i < k-2$:

$$\begin{aligned} \mathrm{LCH}_{k-i-1}^{\mathrm{PrTime}(n^{d^i})} &= \mathrm{LCH}_{k-(i+1)-1}^{\mathrm{PrTime}(n)^{\mathrm{PrTime}(n^{d^i})}} \\ &\subseteq \mathrm{LCH}_{k-(i+1)-1}^{\mathrm{PrTime}(n^{d^{i+1}})}. \end{aligned}$$

Hence, $\mathrm{LCH}_k = \mathrm{LCH}_{k-1}^{\mathrm{PrTime}(n^{d^0})} \subseteq \cdots \subseteq \mathrm{LCH}_1^{\mathrm{PrTime}(n^{d^{k-2}})} \subseteq \mathrm{PrTime}(n^{d^{k-1}})$. □

We can use these two lemmas to prove Theorem 4.

Proof of Theorem 4: Let $a > 1$, and $\epsilon > 0$ be given, and assume that

$$\mathrm{PrTime}(n) \subseteq \mathrm{PrTiSp}(n^a, n^\epsilon).$$

Let $b > 1$ be such that $b\epsilon < 1$. Let $\tau(ba, 1-b\epsilon)$ be the constant from Lemma 5, and let $d < b^{1/(\tau(ba,1-b\epsilon)-1)}$. We claim that

$$\mathrm{PrTime}(n)^{\mathrm{PrTime}(n)} \not\subseteq \mathrm{PrTime}(n^d).$$

Suppose on the contrary that

$$\mathrm{PrTime}(n)^{\mathrm{PrTime}(n)} \subseteq \mathrm{PrTime}(n^d).$$

Then

$$\begin{aligned} \mathrm{PrTime}(n^b) &\subseteq \mathrm{PrTiSp}(n^{ba}, n^{b\epsilon}) \\ &\subseteq \mathrm{LCH}_{\tau(ba,1-b\epsilon)} \\ &\subseteq \mathrm{PrTime}(n^{d^{\tau(ba,1-b\epsilon)-1}}), \end{aligned}$$

where the first inclusion holds by a padding argument, the second follows from Lemma 5, and the third from Lemma 6. This contradicts the Hierarchy Theorem. □

3.1 Proof of Lemma 5

We want to show that any computation running in probabilistic time n^α and space $n^{1-\beta}$ for some constants α and β, lies in some level of the linear time counting hierarchy.

We will prove this lemma in three steps. First, we will build a family of exponentially large constant depth arithmetic circuits computing the number of paths from the start to the accepting configuration in the configuration graph of a $\mathrm{PrTiSp}(n^\alpha, n^{1-\beta})$-machine running on an input x. Second, we will convert this arithmetic circuit into a Boolean threshold circuit of roughly the same size and depth. Third, we will show that the sets recognized by those large (but very uniform) threshold circuits are contained in some level of the linear time counting hierarchy.

Step 1 - Construction of the arithmetic circuit

A probabilistic machine M running in space $n^{1-\beta}$ and time n^α has at most $2^{O(n^{1-\beta})}$ different configurations if it is run on input x of length n. W.l.o.g., we may assume that each configuration is time-stamped; thus the configuration graph of machine M is layered and acyclic. We want to count the number of paths from the start configuration c_{start} to the accepting configuration c_{accept} in the graph. All these paths are of length n^α

Let V_n be the set of configurations of M. We can count the number of paths from c_{start} to c_{accept} in the configuration graph as follows. First we pick $n^\beta - 1$ intermediate checkpoints $c_1, c_2, \ldots, c_{n^\beta-1} \in V_n$, such that configuration c_i is at level $in^{\alpha-\beta}$ in the configuration graph; next we count the number of paths from c_{start} to c_{accept} going through all these checkpoints; finally we take the sum over all possible choices of the checkpoints:

$$\#\mathrm{path}(c_0, c_{n^\beta}, n^\alpha) =$$

$$\sum_{c_1,\ldots,c_{n^\beta-1} \in V_n} \prod_{i=0}^{n^\beta-1} \#\mathrm{path}(c_i, c_{i+1}, n^{\alpha-\beta}),$$

where $c_0 = c_{start}$, and $c_{n^\beta} = c_{accept}$.

We can easily compute the above expression with a depth 2 arithmetic circuit consisting of one $+$ gate and many $\times$ gates, provided that we know the number of paths of length $n^{\alpha-\beta}$ between all the pairs of nodes c_i and c_{i+1}. Since we take the sum over $|V_n|^{n^\beta} = (2^{O(n^{1-\beta})})^{n^\beta} = 2^{O(n)}$ different combinations of check points, the $+$ gate has fan-in $2^{O(n)}$, whereas the $\times$ gates have fan-in $n^\beta + 1$.

The preceding two paragraphs give a reduction: the problem of computing the number of paths of length n^α from c to c' is reduced to the problem of counting paths of length $n^{\alpha-\beta}$. By iterating this reduction α/β times, we obtain an arithmetic circuit of depth $2\alpha/\beta$ that has as its inputs the number of paths of length $n^\alpha/(n^\beta)^{\alpha/\beta} = 1$ between any two configurations. The number of paths of length 1 between two configurations can be directly computed from input x by considering the transition relation of machine M.[1]

Note that the output of the arithmetic circuit is a number with polynomially many bits in terms of n, since probabilistic machine M has running time n^α.

[1] We omit here usual technical details arising from the possible non-integrality of n^α, n^β, etc.

It is straightforward to see that the total size of the circuit is $2^{O(n)}$. We can uniquely label a $+$ gate at level k, that corresponds to the start- and endpoints c_0 and c_{n^β}, by $\langle +, k, c_0, c_{n^\beta}\rangle$. A $\times$ gate at level k, that corresponds to the start- and endpoints c_0 and c_{n^β}, and the checkpoints $c_1, \ldots, c_{n^\beta-1}$, can be labeled by $\langle \times, k, c_0, \ldots, c_{n^\beta}\rangle$. Since there are at most $2^{O(n)}$ such labels, the size of the circuit is exponential in n.

Further, the circuit is highly uniform. It is easy to determine if two gates g and h are connected or not. This can even be done in linear time in terms of the length of the description of g and h, i.e., in time linear in n.

Step 2 - Conversion of the arithmetic circuit into a Boolean threshold circuit

Now we can use the recent result that iterated multiplication is in logspace uniform TC^0 ([9], see also [3, 17]) along with the fact that iterated addition is in Dlogtime-uniform TC^0 ([6]) to convert the arithmetic circuit to a constant-depth Boolean threshold circuit.

Let us assume that the output of the arithmetic circuit has n^s bits. We replace each $+$ gate with the appropriate constant-depth threshold circuit of size $2^{O(n)}$ with n^s outputs computing the iterated sum. Similarly we replace each $\times$ gate with a subcircuit of polynomial size in n computing the iterated product of the n^β inputs with n^s bits. The resulting Boolean threshold circuit is also of size $2^{O(n)}$ and it also computes the number of s-t paths in the graph.

This Boolean circuit is not necessarily linear time uniform in terms of n, but we show that we can determine whether two gates are connected or not in linear time with access to an oracle from the linear time counting hierarchy. We can label each gate g in the Boolean circuit with a label $\langle l_a, l_s\rangle$, where l_a is the label of the substituted $+$ or $\times$ gate in the arithmetic circuit and l_s is the relative label g has in the addition/multiplication subcircuit.

We can reduce the question of whether two gates are connected in the Boolean circuit to the connectivity question for the subcircuits. Two gates g and h are connected either because they belong to the same subcircuit and are adjacent within that subcircuit, or because they are in two adjacent subcircuits (a question we can answer in linear time in terms of n), and one is an output gate and the other is a matching input gate.

Since the addition circuits are Dlogtime uniform, they maintain the uniformity of the Boolean circuit. The multiplication circuits are only logspace uniform. But Nepomnjaščiĭ's theorem ([21]) yields as a corollary that every set $A \in \mathrm{L}$ is contained in $\Sigma_k^{O(n)} \subseteq \mathrm{LCH}_k$ for some k. Therefore we can decide the connectivity language for the multiplication subcircuits (and thus the entire circuit) in linear time in terms of n as long as we have access to an oracle from the k-th level of the linear time counting hierarchy. (Alternatively, one can use the recently-proved theorem of Hesse [17], which improves [9] by showing that iterated product lies in Dlogtime-uniform TC^0.)

We have now constructed a family of Boolean threshold circuits $\{C_n\}$ of size $2^{O(n)}$ and depth $O(\frac{2\alpha}{\beta})$, for which the set $\{(x, g, h) \ : \ h \text{ is an input gate of } g \text{ in } C_{|x|}\}$ is in LCH_k, for a suitable constant k. This constant depends on the exact logspace uniformity of the TC^0 circuits for iterated multiplication of n^β inputs of size n^α.

Step 3 - Putting the arithmetic circuit into LCH

Finally, we will show that the language decided by the circuit, that was obtained in the previous step, lies in LCH. More precisely we will show by induction on j that for any threshold circuit of depth j with the above mentioned size and uniformity conditions, the set $\{(x, g) : \text{gate } g \text{ evaluates to 1 in the circuit } C_{|x|} \text{ on input } x\}$ is in LCH_{j+k}.

Without loss of generality we may assume that the threshold circuit only contains majority and negation gates.

We may inductively assume that we can decide if a gate g in a depth $j-1$ threshold circuit evaluates to 1, by a $\mathrm{LCH}_{(j-1)+k}$ machine. So let us consider some gate g on input x for a circuit $C_{|x|}$ of depth j. We can now randomly guess a gate h and query a LCH_k oracle to find out if h is an input to g. If not, then we make one more random choice, depending on which we accept or reject. If h is an input to g, then h belongs to a depth $j-1$ subcircuit and thus by our induction hypothesis we can query a $\mathrm{LCH}_{(j-1)+k}$ oracle to determine whether h evaluates to 1 or not. If g is a majority gate then we accept if and only if h evaluates to 1; if it is a negation gate, we accept if and only if h evaluates to 0.

It is fairly straightforward to see that this probabilistic oracle Turing machine runs in linear time and accepts (x, g) if and only if gate g evaluates to 1 on input x. Since the running time is linear in n and we require access to a $\mathrm{LCH}_{(j-1)+k}$ oracle we can evaluate a threshold circuit of depth j and LCH_k uniformity in $\mathrm{PrTime}(n)^{\mathrm{LCH}_{(j-1)+k}} = \mathrm{LCH}_{j+k}$.

So we can finally conclude that for every α and β

$$\mathrm{PrTiSp}(n^\alpha, n^{1-\beta}) \subseteq \mathrm{LCH}_{\tau(\alpha,\beta)},$$

where $\tau(\alpha, \beta) = O(\frac{\alpha}{\beta}) + k$ is a positive integer.

□

3.2 Lower Bound for MAJ·MAJSAT

The lower bound that is obtained in the Main Theorem also implies a time–space tradeoff for MAJ·MAJSAT.

Theorem 7 *For every constant* ϵ *such that* $0 < \epsilon < 1$, *there exists a constant* $\gamma' > 0$, *such that*

$$\textit{MAJ·MAJSAT} \not\subseteq \mathrm{PrTiSp}(n^{1+\gamma'}, n^\epsilon).$$

Remark: The statement of Theorem 7 gives essentially the same lower bound for MAJ·MAJSAT as Theorem 3 gives for LCH_2. If one were to compute, for a specific ϵ, the precise value of γ that one obtains in Theorem 3, one would find that the conclusion of Theorem 7 holds for any $\gamma' < \gamma$, but it is not clear that it would hold for $\gamma' = \gamma$.

To show this we will prove that MAJ·MAJSAT is complete for the second level of the linear time counting hierarchy under a fairly restrictive reduction. Namely we will map instances of any language from LCH_2 to instances of MAJ·MAJSAT of size $O(n \log^{O(1)} n)$ via a reduction such that the i-th bit of the formula can be computed in time $O(\log^{O(1)} n)$ and logarithmic space (exactly as in the proof of Theorem 2.4 of [15]). Such a reduction immediately implies Theorem 7.

Let us first define MAJ·MAJSAT. Given two disjoint sets x and y of Boolean variables, consider the problems

$$\begin{aligned} \text{MAJSAT} &= \{\varphi(x) : \varphi(x) \text{ is true for} \geq \tfrac{1}{2} \text{ of} \\ &\qquad \text{the assignments to } x\} \\ \text{MAJ}\cdot\text{MAJSAT} &= \{\varphi(x,y) : \varphi(x,\overline{y}) \in \text{MAJSAT} \\ &\qquad \text{for} \geq \tfrac{1}{2} \text{ of the assignments } \overline{y} \text{ to } y\}. \end{aligned}$$

We will present the reduction of a set $L \in \mathrm{LCH}_2$ to MAJ·MAJSAT in two steps. First we will show that L can be accepted in probabilistic linear time with very restricted oracle access. Then we will show how to represent such a computation by a MAJ·MAJSAT formula.

Consider a set $L \in \mathrm{PrTime}(n)^{\mathrm{PrTime}(n)}$ that is accepted by a linear time probabilistic oracle machine M^A, where $A \in \mathrm{PrTime}(n)$ via a machine M_1. Since $\mathrm{PrTime}(n)$ is closed under complement, there is a linear time probabilistic machine M_0 deciding $\overline{A}$. By standard techniques (see [13]) we can assume that for every $a \in \{0,1\}$ and for every string x, the probability that M_a accepts on input x is not equal to $\frac{1}{2}$; that is, the probability of accepting is always greater than one half, or less than one half. We show how to accept L on a probabilistic linear time machine N with an oracle $C \in \mathrm{PrTime}(n)$, such that N queries C exactly once on each computation path, and where N accepts along the path if and only if the query is answered *YES*. Our construction follows a proof by Torán [25].

The oracle C is defined as follows:

$$\begin{aligned} C \;=\; \big\{&(q_1,a_1,w_1),(q_2,a_2,w_2),\ldots : \forall i\, M_{a_i} \text{ accepts} \\ &\text{input } q_i \text{ along computation path } w_i \text{ and} \\ &|\{v : v < w_i \wedge M_{a_i} \text{ accepts } q_i \text{ along } v\}| = 2^{|q_i|-1}\big\}. \end{aligned}$$

It will be convenient for us to know a string $x_1 \in C$ and a string $x_0 \notin C$, for us to refer to later.

Note that if a_i is the correct answer to query q_i then M_{a_i} accepts q_i on more than $2^{|q_i|-1}$ computation paths. Thus for exactly one computation path w_i, the machine M_{a_i} accepts q_i along w_i and also on $2^{|q_i|-1}$ other computation paths lexicographically less than w_i. So for every sequence of oracle queries $q_1,\ldots,q_m$ there is exactly one correct guess of answers $a_1,\ldots,a_m$ and witnesses $w_1,\ldots,w_m$ such that $(q_1,a_1,w_1),\ldots,(q_m,a_m,w_m) \in B$.

The machine N works as follows. It guesses a sequence of coin flips p and a sequence of queries, answers and corresponding witnesses $q = (q_1,a_1,w_1),\ldots,(q_m,a_m,w_m)$ of length $O(n)$, and then it simulates machine M along path p using q to answer the oracle queries.

If the coin flips p do not correspond to a computation path of M generating queries corresponding to q, or if the coin flips p correspond to a rejecting path of M, then N rejects (by asking if $x_0 \in C$).

Otherwise, the coin flips p do correspond to an accepting computation path of M that generates the queries appearing in q (if the given oracle answers are supplied). In this case, N accepts if and only if $q \in C$.

It is easy to see that the number of accepting computation paths of $N^C(x)$ is equal to the number of accepting computation paths of $M^A(x)$, since for every accepting path p of M there is exactly one corresponding sequence q for which N^C will accept. Thus $x \in L$ if and only if the number of accepting paths of N^C is greater than or equal to $2^{|p|-1}$. Now via standard techniques (see e.g. [25]) we can modify N so that the accepting probability threshold is one-half.

Furthermore, note that N runs in linear time and queries the oracle exactly once at the end of each computation, accepting if and only if the query answer is *YES*. It remains to be shown that $C \in \mathrm{PrTime}(n)$.

Actually we show that $C \in \mathrm{C_{=}Time}(n) \subseteq \mathrm{PrTime}(n)$, where $\mathrm{C_{=}Time}(n)$ is defined as follows:

$$\begin{aligned} \mathrm{C_{=}Time}(n) \;=\; \big\{&A : x \in A \text{ iff } \exists \text{ a prob. linear time} \\ &\text{machine } M \text{ s.t. } \Pr\big[x \in L(M)\big] = \tfrac{1}{2}\big\}. \end{aligned}$$

It is straightforward to verify that language $\big\{(q,a,w) : M_a$ accepts input q along computation path w and $|\{v : v < w \wedge M_a \text{ accepts } q \text{ along } v\}| = 2^{|q|-1}\big\}$ is in $\mathrm{C_{=}Time}(n)$. Results of Fenner *et al.* [13] show that $\mathrm{C_{=}Time}(n)$ is closed under conjunctive reductions, and thus $C \in \mathrm{C_{=}Time}(n)$.

Next we will show how to reduce computation of the machine N^C to a formula from MAJ·MAJSAT. We know that for any $C \in \mathrm{PrTime}(n)$ there is a deterministic machine M_C taking two inputs x and y such that $x \in C$ iff M_C accepts (x,y) for at least half of the possible values for y of length $c|x|$ for some c. By the Fischer-Pippenger construction [12] we can assume that M_C is an oblivious machine (i.e., the position of the worktape heads depend only on the length of the input), running in time $n \log n$.

In [10] Cook gives a construction that reduces M_C to a formula $\varphi_{C,n}(x,y,z)$ of size $n \log^{O(1)} n$, with the property that M_C accepts (x, y) if and only if there is an assignment to the variables z such that $\varphi_{C,n}(x, y, z)$ is true. Furthermore, for any (x, y), there is at most one such z. Let $\psi_{C,n}(x, y, z, v) = (\varphi_{C,n}(x, y, z) \wedge v) \vee (\neg v \wedge (y_1 \vee \bigvee_i z_i))$, where v is a new Boolean variable. Then for any string x of length n, $x \in C$ iff $\psi_{C,n}(x, y, z, v) \in$ MAJSAT, (where we are making the obvious correspondence between the string $x \in \{0, 1\}^*$ and an assignment to the x variables in $\psi_{C,n}$).

Now consider the computation of the LCH$_2$-machine N^C. Since C is reducible in $n \log^{O(1)} n$ time to MAJSAT, we can simulate N^C by a probabilistic machine N' running in time $n \log^{O(1)} n$ with MAJSAT as an oracle. Furthermore, since MAJSAT is easily seen to be paddable, we can assume that all queries made by N' have the same length. By application of the reduction outlined above, all queries of the form "Is y in MAJSAT" made by N' on input x can be expressed by asking if the string $\psi_{A,m}(y, w)$ is in MAJSAT, where $A =$ MAJSAT, $m = |y|$ depends only on the length of x, and w is a sequence of variables (as outlined above, with some renaming).

Note that the set $D = \{(x, p, y) : N'$ on input x with coin flip sequence p makes query y to the oracle$\}$ is accepted by a deterministic machine running in time $n \log^{O(1)} n$. Thus, as above, we can use Cook's reduction to construct efficiently a formula $\varphi_{D,m'}(x, p, y, z)$ such that (x, p, y) is in D if and only if $\varphi_{D,m'}(x, p, y, z)$ is in SAT.

Now observe that x is accepted by N^C if and only if, for at least half of the paths p, the query y asked by N' along path p is in MAJSAT. In turn, this happens if and only if for at least half of the assignments to the variables in p, for the unique (y, z) such that N' queries y along path p on input x (satisfying the formula $\varphi_{D,m'}(x, p, y, z)$), the formula $\psi_{A,m}(y, w)$ is in MAJSAT. This happens if and only if the formula $\Phi(x, (p, y, z, v), w) = (v \wedge (\varphi_{D,m'}(x, p, y, z) \wedge \psi_{A,m}(y, w))) \vee (\neg v \wedge (p_1 \vee \bigvee_i y_i \vee \bigvee_i z_i) \wedge (w_1 \vee \neg w_1))$ is in MAJ·MAJSAT. $\Box$

4 SAC^1 Lower Bounds

In this section, we extend another result of Fortnow [14]. Fortnow shows that SAT cannot be recognized by very small logspace-uniform NC^1 circuits. Here, we prove similar results about SAC^1 circuits – although we have to be somewhat particular about how "size" is defined.

Theorem 8 *Neither SAT nor $\overline{\text{SAT}}$ can be solved by logspace-uniform SAC^1 circuits having $n^{1+o(1)}$ wires.*

Theorem 9 *$\overline{\text{SAT}}$ cannot be solved by Dlogtime-uniform SAC^1 circuits having $n^{1+o(1)}$ gates.*

For both of these theorems, it will be convenient to make use of the following technical result:

Theorem 10 *[16] Every language that is accepted by a nondeterministic random-access machine using t(n) time is also accepted by a nondeterministic multitape Turing machine using time $t(n) \log^{O(1)} t(n)$ time. This result also generalizes to show that any random-access alternating Turing machine using time $t(n)$ can be simulated in time $t(n) \log^{O(1)} t(n)$ on a multitape alternating Turing machine using the same number of alternations.*

The proof of Theorem 8 follows very closely the proof of Theorem 5.1 in [14] (attributed there to Harry Buhrman). Recall that QBF$_i$ denotes the set of true quantified Boolean formulas with i alternating blocks of quantifiers, starting with existential quantifiers. Throughout this proof we adopt the notation of [14], where $\Sigma_{a(n)}^{t(n)}$ ($\Pi_{a(n)}^{t(n)}$) denotes the class of languages accepted by alternating machines making $a(n) - 1$ alternations, running for time $t(n)$, where the start configuration is existential (universal).

Proof of Theorem 8:

We begin by recalling the following fact from [14][Corollary 5.3]:

> If SAT has circuits of size $n^{1+o(1)}$, then QBF$_2 \in \Pi_2^{n^{1+o(1)}}$.

The proof of this follows directly from [18] – but it is important for the proof that the circuits under consideration have *bounded* fan-in (if the size of the circuit is measured by the number of gates). We do not know if this implication holds when unbounded fan-in circuits are considered. On the other hand, it is a simple matter for a random-access machine to simulate a circuit in time bounded roughly by the number of wires in the circuit. Thus we obtain the following.

Observation: If SAT has (unbounded fan-in) circuits with $n^{1+o(1)}$ wires, then QBF$_2 \in \Pi_2^{n^{1+o(1)}}$.

Note that this is true, regardless of whether the circuits are uniform or not.

Next, we observe that the proof of Theorem 5.1 of [14] (without modification) shows that, if SAT has (unbounded fan-in) circuits with $n^{1+o(1)}$ wires, then for a slowly-growing (but unbounded) function $s(n)$,

$$\Sigma_{s(n)}^{n \log n} \subseteq \Sigma_2^p.$$

Now assume that SAT can be solved by logspace-uniform SAC^1 circuits having $n^{1+o(1)}$ wires. Under this assumption, we can see that NP = SAC^1 $= \Sigma_2^p$. Since SAC$^1 \subseteq$ NTiSp$(n^{O(1)}, \log^2 n)$ [23], and since this latter class is contained in $\bigcup_k \Sigma_k^{O(n)}$ [21], we now have the sequence of inclusions:

$$\Sigma_{s(n)}^{n \log n} \subseteq \Sigma_2^p = \text{SAC}^1 \subseteq \bigcup_k \Sigma_k^{O(n)}.$$

Now, we observe that this contradicts a simple diagonalization argument that shows

$$\Sigma_{s(n)}^{n \log n} \not\subseteq \bigcup_k \Sigma_k^{O(n)}.$$

A similar argument shows that the same bound holds for $\overline{\text{SAT}}$. □

Recently van Melkebeek has observed that the uniformity condition in Theorem 8 can be relaxed significantly, from logspace to $\text{NTiSp}(n^{O(1)}, n^{1-\epsilon})$ [20].

To prove our other lower bound for SAC^1 circuits, we first need to observe that they are easy to evaluate.

Lemma 11 *If a set A has Dlogtime-uniform SAC^1 circuits with $n^{1+o(1)}$ gates then $A \in Ntime(n^{1+o(1)})$.*

Proof: Assume such a set A has a Dlogtime-uniform SAC^1 circuit family $\{C_n\}_{n=0}^{\infty}$. Given input x of size n, our nondeterministic machine for A wants to verify that $C_n(x) = 1$. It does that as follows.

Observe, $C_n(x) = 1$ iff there is a subcircuit of C_n that evaluates to one and which consists of AND gates of bounded fan-in and OR gates of fan-in 1. Given the input, we may guess such a subcircuit and verify that it really evaluates to one.

A description of the subcircuit is list of pairs $\langle g, h\rangle$, where g and h are gates of the subcircuit and g is an input gate to h. The length of the description is $O(n^{1+o(1)})$, hence it may be guessed in time $O(n^{1+o(1)})$. Using Dlogtime-uniformity of C_n, it can be verified in time $O(n^{1+o(1)})$ that the description is indeed a subcircuit of C_n.

The subcircuit can be evaluated in bottom-up fashion on a deterministic random-access machine in time $O(n^{1+o(1)})$, given that the description was guessed in topologically sorted form. Hence by Theorem 10, it can be evaluated in non-deterministic time $O(n^{1+o(1)})$. If the subcircuit evaluates to one, we accept input x, otherwise we reject it. □

Remark: Although $\text{SAC}^1 \subseteq \text{NTiSp}(n^{O(1)}, \log^2 n)$ [23], it is not known how to evaluate a nearly-linear-size SAC^1 circuit in nearly-linear nondeterministic time, using less than linear space. Similarly, although SAC^1 is closed under complement [4], known constructions involve squaring the circuit size.

A simple corollary to the previous lemma is the following statement that is similar in flavor to (but much easier than) the version of the Karp-Lipton collapse that we used in the proof of Theorem 8. Note that the uniformity condition is essential here.

Corollary 12 *If $\overline{SAT}$ has Dlogtime-uniform SAC^1 circuits of size $n^{1+o(1)}$ then $QBF_1 \in \Pi_1^{n^{1+o(1)}}$.*

Proof of Theorem 9: Assume that $\overline{\text{SAT}}$ is solved by Dlogtime-uniform SAC^1 circuits having $n^{1+o(1)}$ gates. By Corollary 12, $\text{QBF}_1 \in \Pi_1^{n^{1+o(1)}}$. As in [14], it follows that for a slowly-growing (but unbounded) function $s(n)$,

$$\Sigma_{s(n)}^{n \log n} \subseteq \Sigma_1^p.$$

The rest of the proof follows along the lines of the proof of Theorem 8. □

References

[1] M. Ajtai. A non-linear time lower bound for Boolean branching programs. In *Proc. IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 60–70, 1999.

[2] E. Allender. The permanent requires large uniform threshold circuits. *Chicago Journal of Theoretical Computer Science*, article 7, 1999.

[3] E. Allender, D. Mix Barrington, and W. Hesse. Uniform circuits for division: Consequences and problems. These proceedings.

[4] A. Borodin, S. A. Cook, P. W. Dymond, W. L. Ruzzo, and M. Tompa. Two applications of inductive counting for complementation problems. *SIAM Journal on Computing*, 18:559–578, 1989.

[5] J. Balcázar, J. Díaz, and J. Gabarró. *Structural Complexity Theory I and II.* Springer-Verlag, 1995,1990.

[6] D. A. M. Barrington, N. Immerman, and H. Straubing. On uniformity within NC^1. *Journal of Computer and System Sciences*, 41:274–306, 1990.

[7] P. Beame, M. Saks, X. Sun, and E. Vee. Super-linear time-space tradeoff lower bounds for randomized computation. In *Proc. IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 169–179, 2000.

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

[9] A. Chiu, G. Davida, and B. Litow. NC^1 Division. Preliminary version, 2000. Available from the website `http://www.cs.jcu.edu.au/~bruce` as `/papers/crr00_3.ps.gz`.

[10] S. A. Cook. Short propositional formulas represent nondeterministic computations. *Information Processing Letters*, 26:269–270, 1988.

[11] D.-Z. Du and K.-I. Ko. *Theory of Computational Complexity.* Wiley-Interscience, New York, 2000.

[12] M. Fischer and N. J. Pippenger. Relations among complexity measures. *Journal of the ACM*, 26:361–381, 1979.

[13] S. Fenner, L. Fortnow, and S. Kurtz. Gap-definable counting classes. In *Proceedings of the 6th IEEE Structure in Complexity Theory Conference*, pages 30–42, 1991.

[14] L. Fortnow. Time-space tradeoffs for satisfiability. *Journal of Computer and System Sciences*, 60:336–353, 2000.

[15] L. Fortnow and D. van Melkebeek. Time-space tradeoffs for nondeterministic computation. In *Proc. IEEE Conference on Computational Complexity*, pages 2–13, 2000.

[16] Y. Gurevich and S. Shelah. Nearly-linear time. In *Proc. Logic at Botik*, number 363 in Lecture Notes in Computer Science, pages 108–118. Springer, 1989.

[17] W. Hesse. Division is in uniform TC^0. In *ICALP 2001: Twenty-Eighth International Colloquium on Automata, Languages and Programming* (July 2001), to appear.

[18] R. Karp and R. Lipton. Turing machines that take advice. *L'Ensignement Mathématique*, 28:191–210, 1982.

[19] R. Lipton and A. Viglas. On the complexity of SAT. In *Proc. IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 459–464, 1999.

[20] D. van Melkebeek. Time-space lower bounds for satisfiability. *Bulletin of the European Association for Theoretical Computer Science*, 73:57-77, 2001.

[21] V. A. Nepomnjaščiĭ. Rudimentary predicates and Turing calculations. *Soviet Math. Dokl*, 11:1462–1465, 1970.

[22] C. Papadimitriou. *Computational Complexity*. Addison-Wesley, New York, 1994.

[23] Walter L. Ruzzo. Tree-size bounded alternation. *Journal of Computer and System Sciences*, 21:218–235, 1980.

[24] S. Toda. PP is as hard as the polynomial time hierarchy. *SIAM Journal on Computing*, 20:865–877, 1991.

[25] J. Torán. Complexity classes defined by counting quantifiers. *Journal of the ACM*, 38:753–774, 1991.

[26] I. Tourlakis. Time-space lower bounds for SAT on uniform and non-uniform machines. In *Proc. IEEE Conference on Computational Complexity*, pages 22–33, 2000.

[27] H. Venkateswaran. Properties that characterize LOGCFL. *Journal of Computer and System Sciences*, 43:380–404, 1991.

[28] H. Vollmer. *Introduction to Circuit Complexity*. Springer-Verlag, 1999.