LABORATORY FOR
COMPUTER SCIENCE

MASSACHUSETTS
INSTITUTE OF
TECHNOLOGY

MIT/LCS/TM-205

CIRCUIT-SIZE LOWER BOUNDS AND NON-REDUCIBILITY TO SPARSE SETS

Ravindran Kannan

October 1981

545 TECHNOLOGY SQUARE, CAMBRIDGE, MASSACHUSETTS 02139

Circuit-size lower bounds and non-reducibility to sparse sets

R. Kannan*

Department of Mathematics
Massachusetts Institute of Technology

Abstract

As remarked in Cook (1980), we do not know any nonlinear lower bound on the circuit-size of a language in P or even in NP. The best known lower bound seems to be due to Paul (1975). In this paper we show that first for each nonnegative integer k, there is a language L_k in $\Sigma_2 \cap \pi_2$ (of Meyer and Stockmeyer (1972) hierarchy) which does not have $0(n^k)$-size circuits. Using the same techniques, one is able to prove several similar results. For example, we show that for each nonnegative integer k, there is a language L_k in NP that does not have $0(n^k)$-size uniform circuits. This follows as a corollary of a stronger result shown in the paper.

Finally, we note that existence of "small circuits" is in suitable contexts equivalent to being reducible to sparse sets. Using this, we are able to prove for example that for any time-constructible super-polynomial function f(n), NTIME(f(n)) contains a language which is not many-to-one p-time reducible to any sparse set.

---

* Supported by NSF Grants MCS-8105557 and MCS-77-09906

1. Introduction

A circuit for us is a Boolean circuit containing AND, OR and NEGATION gates with fan-in and fan-out of at most 2. (We remark that since all the results are proved for all polynomials, they are not changed if we allow arbitrary fan-out or allow other gates - for example EXCLUSIVE OR etc. because under these changes circuit-size as defined below changes by at most a polynomial (see Savage (1976)). The circuit-size $s(C)$ of a circuit C is the number of gates in the circuit. Clearly, there are constants k_0 and k such that every circuit of size s can be encoded into a 0,1 string $e(C)$ of length at most $k_0(s(C))^k$.

We let P as usual denote the class of languages accepted by a multitape Turing machine in polynomial-time. $\Sigma_i(\pi_i)$ is the class of languages accepted in polynomial-time by machines with i alternations beginning with an existential (universal) guess. (Kozen (1976) and Chandra and Stockmeyer (1976)).

A family of circuits $\{C_n\}_{n=1}^{\infty}$ is said to accept a language L over $\{0,1\}$ if for each n, C_n is an n-input circuit whose output is 1 for precisely the n-length strings of L(0 for other n-length strings). We say that a language L has $0(f(n))$ circuits if $\{C_n\}_{n=1}^{\infty}$ accepts L and there is a constant k such that $s(C_n) \leq kf(n)$ for all but finitely many n. A language has small circuits if it has $0(n^k)$-size circuits for some fixed k. A class of languages has small circuits if every member of it does. If one can show

that some language L in NP (or in fact in Σ_i for some i) does not have small circuits, then we would have proved that NP≠P. (Since P does have small circuits (Savage (1972)). At present however, no nonlinear lower bounds are known on the circuit-size of any language in P or NP or Σ_i for low i. The following lemma is proved by classical counting arguments.

Lemma 0: For any positive integers n and k, there is an n-input circuit of size n^{2k+2} which accepts a subset of the 2^n n-length strings not accepted by any n-input circuit of size at most n^k.

Proof: By our definition of circuit, since fan-out is 2 or less, there are at most

$$3^{n^k} \cdot (n^k)^{cn^k}$$ different circuits of size n^k or less (where c is some universal constant). This is asymptotically at most $(_2n^{2k})$. If $\{0,1\}^n = \{x_1,\ldots,x_{2^n}\}$, there are $2^{n^{2k+1}}$ distinct subsets of $\{x_1,x_2,\ldots,x_{n^{2k+1}}\}$. Thus one of these subsets, say, S cannot be accepted by any n-input circuit of size n^k or less. But S is accepted by an n-input circuit of size at most $2\cdot n^{2k+2}$, because each individual string can be accepted by a circuit of size (2n) and S is an "OR" of n^{2k+1} of these. Arguing more carefully, one can drop the factor of 2 and prove the lemma.

2. A Circuit-size lower bound

Our strategy is to prove the result first for $\Sigma_4 \cap \pi_4$ and then invoke the following very interesting theorem found in Karp and Lipton (1980):

Theorem 1: (Karp, Lipton and Sipser): If NP has small circuits then for all $k \geq 2$,

$$\Sigma_k = \Sigma_2 .$$

Lemma 1: For each integer k, there is a language L_k in $\Sigma_4 \cap \pi_4$ such that L_k does not have circuits of size $0(n^k)$.

Proof: The action of L_k is described by a first order formula with 4 alternations. The formula simulates a circuit C* of size n^{2k+4} which is not equivalent to any circuit of size n^{k+1} (cf. lemma 0). The first three statements of the formula below ensure this. But we also need to force L_k to simulate the same circuit C* on all inputs of length n. This is accomplished by the last 4 steps that choose the "minimum" C* with the necessary property. L_k is accepted by a Σ_4 machine M which functions as follows:

On input x of length n, M accepts iff

1) $\exists$ an encoding e(C*) of a circuit C* of size at most n^{2k+4}

2) [$\forall$ encodings e(C') of circuits of size at most n^{k+1}

3) $\exists$ a n-length 0,1 string y such that C* and C' differ on y]

and

4) [$\forall$ encodings e(C) of circuits with e(C) $\leq$ e(C*) (as binary integers)

5) $\exists$ an encoding $e(C_0)$ of a circuit of size at most n^{k+1} s.t.

6) $\forall$ strings Z of length n, C_0 agrees with C.]

and

7) C* accepts x.

For any standard choice of encoding, it is not difficult to see that given the encoding of a circuit and an input to the circuit, we can, in polynomial-time, find the action of the circuit on the input. (Savage (1976)). Thus L_k is in Σ_4. Clearly, the complement of L_k is accepted by a Σ_4 machine whose description is the same as that of L_k's except now step 7 reads "and C* rejects x". Thus L_k is in $\Sigma_4 \cap \pi_4$. It is obvious from the operation of L_k that it does not have n^{k+1} size circuits, hence does not have $0(n^k)$ size circuits.

Theorem 2: For any nonnegative integer k, there is a language L_k in $\Sigma_2 \cap \pi_2$ such that L_k does not have $0(n^k)$ size circuits.

Proof: The proof is made extremely simple because of Theorem 1. We just need to consider 2 cases:

Case 1: NP has small circuits: In this case, by Theorem 1, our language L_k of lemma 1 is in Σ_2 (because $\Sigma_4=\Sigma_2$) and so also its complement thus establishing the theorem.

Case 2: NP does not have small circuits: In this case, there is a language L in NP such that L does not have $0(n^k)$ size circuits for any k. L is of course in $\Sigma_2 \cap \pi_2$ and thus proves the theorem.

Remark 1: In case 2, it is not difficult to see that NP-complete language SAT given by:

SAT = {x| x is an encoding of a satisfiable Boolean formula}

does not have small circuits. The reason for this is that since P has small circuits and every language in NP is polynomial-time reducible to SAT (Cook 1971), if SAT had small circuits, so would every language in NP.

Thus we can assert that L_k of Theorem 2 is either SAT or L_k of lemma 1. But of course we will not know which it is unless we settle some really hard problems.

Remark 2: Theorem 2 leads to some curious observations, for example, we can assert that if NP had $0(n^k)$ circuits for some k fixed, then NP $\neq$ P. However, since the hypothesis of the above statement would seem to be rather more unlikely to be true then the conclusion, it is not terribly useful.

3. Related Results

A language $L \subseteq \{0,1\}^*$ is sparse if there is a polynomial $p(\cdot)$ such that $|L \cap \{0,1\}^n \leq p(n)$. Sparse languages obviously have small circuits (a disjunction of $p(n)$ conjuncts would do). Thus given k, it is of interest to produce a sparse language that does not have $0(n^k)$ circuits. If there was a sparse NP-complete language, then by using the ideas in the proofs of Theorems 2 and 3 (to follow), one could show that for each k, there is a sparse language in $\Sigma_2 \cap \pi_2$ which does not have $0(n^k)$ circuits. But unfortunately, there is no sparse NP-complete language unless NP=P (Mahaney (1980)) and we have to contend ourselves with working in $\Sigma_3 \cap \pi_3$.

Theorem 3: For each positive integer k, there is a sparse language L_k in $\Sigma_3 \cap \pi_3$ such that L_k does not have $0(n^k)$ size circuits.

Proof: First, we observe that in proving lemma 0, the set S accepted by a circuit of size at most n^{2k+4}, but not by any circuit of size n^{k+1} was "sparse"-- in fact it was a subset of $\{x_1, x_2, \ldots, x_{n^{2k+3}}\}$ where $\{0,1\}^n = \{x_1, x_2, \ldots, x_{2^n}\}$. We can thus restate lemma 0 as follows (with the notation that for any n-input circuit C and n-length string x, C(x) = 1 if C accepts x, else 0):

Lemma 2: There is a 0,1 string y of length n^{2k+3} say - $y = y^{(1)} y^{(2)} \ldots y^{(n^{2k+3})}$ such that for each n-input circuit C of

size at most n^{k+1}, there is a j, $1 \leq j \leq n^{2k+3}$ such that

$$y^{(j)} \neq C(x_j) \quad .$$

Consider the Σ_3 machine M that behaves as follows:

On input x of length n M accepts iff (we use the notation introduced so far):

0) $x = x_{j_0}$ for some j_0, $1 \leq j_0 \leq n^{2k+3}$ and

1) $\exists\, y = y^{(1)}y^{(2)}\ldots y^{(n^{2k+3})}$ such that

2) $[\forall e(C)$ - encodings of circuit C of size at most n^{k+1}

3) $\bigvee_{j=1}^{n^{2k+3}} (y_j \neq C(x_j))]$

and

4) $[\forall z = z^{(1)}\ldots z^{(n^{2k+3})}$,

$$\sum_{j=1}^{n^{2k+3}} z^{(j)}2^j < \sum_{j=1}^{n^{2k+3}} y^{(j)}2^j.$$

5) $\rightarrow \exists e(C')$ of a circuit C' of size at most n^{k+1}. s.t.

6) $\bigwedge_{j=1}^{n^{2k+3}} (z^{(j)} = C(x_j))]$

and

7) $y^{(j_0)} = 1.$

Here V stands for OR and Λ for AND. First of all, M has at most 3 alternations and is a Σ_3 machine. Secondly the disjunction and conjunction in steps 3 and 6 respectively are of polynomially

many simple predicates and thus can be checked in polynomial-time. The other steps are polynomial-time bounded as argued earlier. Thus the language is in Σ_3.

As before, steps 1 through 3 pick out a y that represents something different from all circuits of size at most n^{k+1}. The other steps ensure that we use the same such y for all n-length inputs. Thus the language above does not have $0(n^k)$ circuits.

Step 0) ensures that the language is sparse.

The complement language results when we change to step 0 to read "$x \neq x_{j_0}$ for any j_0, $j \leq j_0 \leq n^{2k+3}$" and we change step 7 to read "$y^{(j_0)} = 0$". Thus the complement is also in Σ_3 and we have proved the theorem.

We can use Theorem 2 to produce languages which do not have small circuits. For a function f(n), $\Sigma_i^{f(n)}$ ($\pi_i^{f(n)}$) denotes the class of languages accepted by a Σ_i(π_i) machine in time 0(f(n)). Thus

$$\Sigma_i = \Sigma_i^p = \bigcup_{k=1}^{\infty} \Sigma_i^{n^k} .$$

A function f(n) is said to be super-polynomial if for each $k \geq 1$ integer, $\lim_{n\to\infty} n^k/f(n) = 0$. It is known that for any "nice" "super-polynomial" f(n), SPACE (f(n)) - the class of languages accepted

by a deterministic $0(f(n))$ space bounded TM contains a language which does not have small circuits. The proof is by what might be called the "voting strategy" which is an elaboration of the counting argument of lemma 0.

Definition: We say that a function $f(n) \geq n$ is time-constructible if a deterministic multitape $0(f(n))$-time bounded TM computes $f(n)$ (in binary) on each input of length n.

Lemma 3: If $f(n)$ is a super polynomial time-constructible function, then SPACE $(f(n))$ contains a language which does not have small circuits.

Sketch of Proof: On inputs of length n, we diagonalise over all n-input circuits encodable in $\sqrt{f(n)}$ or less, $\sqrt{f(n)}$ also being super-polynomial this suffices. Care has to be exerted in the diagonalization -- there are only 2^n inputs of length n, but $2^{\sqrt{f(n)}}$ circuits to deal with. The "voting strategy" works as follows: Assume without loss of generality that $\sqrt{f(n)} < 2^{n/2}$. Suppose $\{0,1\}^n = \{x_1,\ldots,x_{2^n}\}$ and we have decided what to do on inputs $x_1,\ldots,x_i$. On input x_{i+1}, our machine finds out first, what it decided to do on $x_1,\ldots,x_i$. Next, it enumerates all circuits of encoding size $\sqrt{f(n)}$ or less. Among the circuits that agree with our machine on $x_1,\ldots,x_i$, a majority must either accept or reject x_{i+1}. Our machine will do the opposite of the majority. Thus for any circuit with encoding length at most

$\sqrt{f(n)}$, on at least one input, among $\{x_1,\ldots,x_{\sqrt{f(n)}}\}$, our machine will differ from the circuit. It is not difficult to reckon the space requirements to complete the proof.

It is not difficult to see that the time taken by the above machine is at least $2^{\sqrt{f(n)}}$. Here, we show the stronger result:

Lemma 4: If f(n) is any increasing time -- constructible super-polynomial function, then there is a language L in $\Sigma_2^{f(n)} \cap \pi_2^{f(n)}$ that does not have small circuits.

Proof: If SAT does not have small circuits, then of course SAT = L would do. Thus we may assume that it does and hence $\Sigma_k^p = \Sigma_2^p$ for all k. Suppose the language L_1 of Theorem 3 is in $\Sigma_2^{n^\ell} \cap \pi_2^{n^\ell}$. For any integer N, we assume that $\{0,1\}^N = \{x_1,\ldots,x_{2^N}\}$ where $x_1 < x_2 < \ldots < x_{2^N}$. (< reads lexicographically greater than). We further assume $f(n) \leq 2^{n/20}$, else we replace f(n) by $\min(f(n), 2^{n/20})$. Now, the machine M accepting L behaves as follows:

1) On input x of length n, compute

$f(n); \lceil (f(n))^{1/\ell} \rceil = m$, (say.)

2) Accept x iff $0^m x$ is in L_1 (of Theorem 3).

Since f(n) is time-constructible and once, f(n) is found, m can be found in at most $0((\log f(n))^s)$ steps for some s, step 1 takes time 0(f(n)). Since L_1 is in $\Sigma_2^{n^\ell}$, step 2 can also be done in 0(f(n)) time. Thus the language L defined by 1) and 2) is certainly in $\Sigma_2^{f(n)} \cap \pi_2^{f(n)}$. (The latter because the complement of

L_1 is also in $\Sigma_2^{n^\ell}$). We now wish to claim that L does not have $0((f(n))^{1/\ell})$ - size circuits. For suppose it did. Then, suppose N is any integer such that there exists an n with $N = \lceil (f(n))^{1/\ell} \rceil + n$. Consider $L_1 \cap \{0,1\}^N$. This is a subset of the N^5 lexicographically least strings of $\{0,1\}^N$. By our assumption of f(n), n>s log N and thus every string of $L_1 \cap \{0,1\}^N$ has at least m 0's on the left. Let C be a circuit accepting $L \cap \{0,1\}^n$. Then C' which is C plus a device to check whether there are enough leading o's accepts $L_1 \cap \{0,1\}^N$. Hence, if L has $k.f(n)^{1/\ell}$ size circuits, then L_1 has 0(n) circuits for infinitely many n. Going back to the construction of L_1 in Theorem 3, we see that this is impossible. (Note: This is a stronger statement than the theorem which only said that L_1 does not have 0(n) circuits.) Since f(n) is superpolynomial, so is $(f(n))^{1/\ell}$ and hence we have proved the lemma.

What we have is actually the following:

Theorem 4: There is a universal constant ℓ such that for any time -- constructible function f(•) satisfying $n^\ell \leq f(n) \leq 2^{n/20} \forall n$, there is a language in $\Sigma_2^{f(n)} \cap \pi_2^{f(n)}$ that does not have $0((f(n))^{1/\ell})$-size circuits.

This implies Theorem 2 and in fact points out another way of establishing Theorem 2 -- first establish the existence of L_1, then apply "padding arguments".

4. Other Nonuniform Measures

A $\Sigma_k(\pi_k)$ formula is a quantified Boolean formula with k alternations beginning with an existential (universal) quantifier. The size of a Σ_k formula is the total number of quantified variables plus Boolean connectives. (We could have included the length of subscripts of variables etc. -- but these do not change the length by more than a polynomial and thus do not affect our results). We can then define the concept of a language or a family of languages "having $0(f(n))$ size $\Sigma_k(\pi_k)$ formulas" just as in the case of circuits. By a development quite similar to that of Theorem 1, one can then show:

If Σ_{k+1} and π_{k+1} have small Σ_k formulas, then $\Sigma_j = \Sigma_{k+2}$ $j \geq k+3$.

Using this and the method of Theorem 2, it is easy to show that:

For all $j \geq 1$ integer, there is a language L_j in $(\Sigma_{k+2} \cap \pi_{k+2})$ such that L_j does not have $0(n^j)$ - size Σ_k-formulas.

Similar theorems can be proved about the nonuniform versions of first-order expressibility of Immerman (1980).

We will now focus attention on a quasi-uniform measure which we call "provable circuit-size". On the one hand we have the measure "circuit-size" which is totally nonuniform, in the sense that even if $\{C_n\}_{n=1}^{\infty}$ accepts language L, the function $n \to C_n$ may not even be recursive. At the other extreme are definitions of uniform circuit-size (Cook (1980)) where the above function is required to be computable in log space. Provable circuit size is in between.

Definition: A family of circuits $\{C_n\}_{n=1}^{\infty}$ provably accepts L if C_n accepts $L \cap \{0,1\}^n$ for all n and the language $\{1^n \# e(C_n) \mid n=1,2,\ldots,\}$ is in NP, where e is some natural encoding.

The reason for the terminology is that the latter language being in NP would imply that there is a "short" proof that C_n is in fact the right circuit. Note that we only require the proof to be polynomially long in $e(C_n)$.

A language L has provable $0(f(n))$-size circuits if $\{C_n\}_{n=1}^{\infty}$ provably accepts L and size $(C_n) \leq f(n)$ for all but finitely many n. We then define a language/a class of languages have probable small circuits, etc. as before. On the lines of Karp, Lipton and Sipser, one can show:

Theorem 5: If NP has provable small circuits, then $\Sigma_k = NP$ for all k.

The proof is quite simple -- under the hypothesis that NP has provable small circuits, we can show that π_1 does too (because circuits are simple deterministic devices). The last statement then implies that $\pi_1 \subseteq NP$, thus proving our theorem. We leave the details to the reader. Using Theorem 5 and arguing as for Theorem 2, one shows:

Theorem 6: For each $k \geq 1$ integer, there is a language L_k in NP such that L_k does not have provable $0(n^k)$-size circuits.

Remark 3: Neither Theorem 6 nor the following corollary of it is provable by direct diagonilisation:

For each $k \geq 1$, there is a language L_k in NP that does not have uniform circuits of size $0(n^k)$.

For example, in the case of uniform circuits, since the log space machine can take $0(n^\ell)$ time for any fixed ℓ, one ND polynomial-time bounded TM cannot seem to simulate all the log space machines to diagonilize over them.

Remark 4: Theorem 6 can be interpreted as saying that there is a language L in NP such that the information about the 2^n or less strings in $L \cap \{0,1\}^n$ cannot be compressed into length $0(n^k)$ by any non-deterministic polynomial-time machine in a certain fashion. This remark is related to Theorem 9 to come that talks about reductions to sparse sets.

5. Reductions to sparse sets

As observed by Meyer (Berman and Hartmanis, 1977), the existence of small circuits for NP is equivalent to the existence of a sparse oracle for NP. (i.e., each language in NP being Turing reducible in p-time to a sparse set). To see the equivalence, first note that the second statement obviously implies the first. Now suppose NP has small circuits. Say L is in NP and has circuits $\{C_n\}_{n=1}^{\infty}$ such that $|e(C_n)| \leq n^k$. Suppose $\{0,1\}^n = \{x_1, \ldots, x_{2^n}\}$. The sparse language S contains x_i iff $i \leq n^k$ and the ith digit of $e(C_n)$, the encoding of C_n is a 1. Clearly L is p-time Turing reducible to S. We will say that a

family $\{C_n\}_{n=1}^{\infty}$ of circuits is p-time constructible if a p-time deterministic multitape Turing machine on input 1^n produces output $e(C_n)$. Mahaney (1980) has shown that if every language in NP is many-one p-time reducible to a sparse set then NP = P. Using this it is not difficult to show that NP having small p-time constructible circuits is equivalent to NP being many-one reducible to a sparse set.

Finally, we defined the property of a language "having provable small circuits". This property is also equivalent, in the case of NP, to being reducible to a sparse set. To make this more precise, we make the following definition:

We say that a language S over an alphabet Σ is an <u>exact sparse set</u> if there is a time-constructible function p(n) which is bounded above by a polynomial such that $|S \cap \Sigma^n| = p(n)$

<u>Lemma 5</u>: NP has small provable circuits iff NP is p-time Turing reducible to an exact sparse set which is itself in NP.

<u>Proof</u>: Suppose $L \in NP$ has small provable circuits $\{C_n\}$. Let $\{1^n \# e(C_n) | n=1,\ldots,\} = L'$. $L' \in NP$ and suppose $|e(C_n)| \leq n^k \forall n$. Consider the sparse language S (again $\{0,1\}^n = \{x_1,\ldots,x_{2^n}\}$) defined by $S \cap \{0,1\}^{n+1} = \{x_i z | 1 \leq i \leq n^k$, z=0 or 1 and the ith digit of $e(C_n)$ is z$\}$. It is an exact sparse language. S is in NP because on input x of length (n+1), we guess $e(C_n)$, use the fact that L' is in NP. Clearly L is reducible to S.

Conversely, suppose language L in NP is p-time Turing reducible to an exact sparse set S in NP. Then surely, L has small circuits $\{C_n\}$ (C_n simulates the p-time reduction remembering- "in memory"- the sparse set S.) These are provable small circuits because S is in NP and is exact.

We observe that the stipulation that the exact sparse set be in NP is crucial. By modifying Meyer's construction - say - as in the proof of the lemma above one can easily show that NP has small circuits iff NP is p-time Turing reducible to an exact sparse set. The conditions in the above lemma are stronger because the sparse set is required to be in NP.

Corresponding to the three equivalences we have above (between particular reducibility of NP to a sparse set and NP having a particular type of small circuits), our techniques yield 3 theorems.

Theorem 7: For each integer k, there is a language L_k in NP such that L_k cannot be many-one reduced in $0(n^k)$ deterministic time to a language of sparsity $0(n^k)$.

Remark: Note again that direct diagonilisation cannot hope to prove this result since we are not restricting the sparse set to any complexity.

Proof: We consider as usual two cases:

Case 1: NP is many-one p-time reducible to a sparse language. Then by Mahaney (1980), NP = P and thus Σ_2 = NP. Thus for each ℓ, NP has a language L_ℓ that does not have $0(n^\ell)$-size circuits. This would be contradicted if every language in NP were reducible in $0(n^k)$ time to a language of sparsity $0(n^k)$ for a fixed k (because then for any language L in NP, there is a $0(n^k)$-det. time reduction f of L to the language S of sparsity $0(n^k)$. Thus any string x belongs to L iff $f(x) \in S$. $|f(x)| = 0(|x|^k)$. There is a $0(|x|^{k^5})$-size circuit that by "table-look-up" checks if f(x) is in S. There is a $0(|x|^{3k})$ size circuit that on input x produces output f(x). Thus L has $0(n^{k^5+3k})$-size circuits.)

Case 2: NP is not many-one p-time reducible to a sparse language. Then, of course the theorem is obvious.

Theorem 8: For each k, there is a language L_k in $\Sigma_2^P \cap \pi_2^P$ such that L_k cannot be Turing reduced in deterministic time $0(n^k)$ to a language of sparsity $0(n^k)$.

Proof: If not, we would contradict Theorem 2.

Theorem 9: For each k, there is a language L_k in NP such that L_k cannot be Turing reduced in deterministic time $0(n^k)$ to any language L satisfying:

(i) $L \cap \{0,1\}^n = n^k \ \forall n$.

(ii) $L \in NP$.

Proof: If not, all of NP will have provable circuits of size $0(n^\ell)$ for a fixed ℓ as argued earlier in this section.

We can also use "padding" arguments to prove results about superpolynomial functions. We state only one of the results that we can prove:

Theorem 10: Suppose f(n) is a super-polynomial time-constructible increasing function. Then there is a language L in NTIME (f(n)) that cannot be many-one p-time reduced to a sparse language.

We remark that Mahaney (1980) shows that if NP $\neq$ P then NP itself has a language that cannot be many-one p-time reduced to a sparse language. The above theorem assumes no such hypothesis (as NP $\neq$ P), but requires super-polynomial time.

Proof: Suppose the theorem is false. Then for following "universal" language U for NTIME (f(n)) there is a sparse set S and a p-time function g such that $x \in U$ iff $g(x) \in S$:

$$U = \{M \# x \mid M \text{ is a NDTM and } M \text{ accepts } x \text{ in time } f(n)\}$$

Let k be such that g is computable in det. time $0(n^k)$ and S is of sparsity $0(n^k)$. Since f(n) is super-polynomial, the languages L_{k+1}, L_{k+2} ..., of Theorem 7 are "sublanguages" of U. The above reduction g of U to S then contradicts Theorem 7.

To summarize the results, we introduce some notation. In keeping with usual notation found in the literature we denote by P^{SP} the class of

languages accepted in det. polynomial time with a sparse oracle - i.e., the class of languages that are p-time Turing reducible to sparse set. The class P may be replaced by other classes, example DTIME (n^k) = class of languages accepted in det.time $0(n^k)$ etc. We may add conditions on the sparse set as follows:

$$P^{SP};\ SP \in NP,\quad |SP| = n^k$$

denotes the class of languages Turing reducible in p-time to a sparse set S over the alphabet Σ in NP s.t.

$$A \overset{m}{\underset{n\ell\text{-time}}{\leq}} |S \cap \Sigma^n| = n^k.$$

B denotes of course that the language A is many-to one reducible in time $0(n^{\ell})$ to the language B.

Finally we remind the reader that a sparse set S over is said to be an exact sparse set if there is a time-constructible polynomial-bounded function p(n) such that $|S \cap \Sigma^n| = p(n)$.

Ackwoledgment: I would like to thank Prof. Hartmanis for a helpful discussion on sparse sets.

Reduction to Sparse Sets (A)	$NP \subseteq P^{SP}$	$NP \subseteq P^{SP}$; $SP \varepsilon NP$, SP exact	$NP \leq^{m}_{p\text{-time}} SP$
Equivalent circuit Property (B)	NP has small circuits (Meyer)	NP has provable small circuits	NP has p-time constructible small circuits. (Mahaney)
Hiearchy collapses to under (A) or (B)	Σ_2 (Karp, Lipton & Sipser)	NP	P (Maheney)
Result on non-reducibility to sparse sets	$\forall k$, $\Sigma_2 \cap \pi_2 \not\subseteq DTIME(n^k)^{SP}$ $\lvert SP \rvert \varepsilon 0(n^k)$	$\forall k$, $NP \not\subseteq DTIME(n^k)^{SP}$; $SP \varepsilon NP$; $\lvert SP \rvert = n^k$	$\forall k$, $NP \not\leq^{m}_{\text{time } 0(n^k)} SP$ $\lvert SP \rvert \varepsilon 0(n^k)$
Circuit-size lower bound	$\forall k$, $\Sigma_2 \cap \pi_2$ does not have $0(n^k)$-size circuits	$\forall k$, NP does not have $0(n^k)$ provable circuits	
Non-reducibility to sparse sets (super-poly. classes)	$\forall$ super-poly., constructible $f(\cdot)$, $\Sigma_2^{TIME(f(n))} \cap \pi_2^{TIME(n)} \not\subseteq P^{SP}$		(Same as col. 1) $NTIME(f(n)) \not\leq^{m}_{p\text{-time}} SP$

References

1. Adelman, L., "Two theorems on random polynomial-time", Proc. of the 19th IEEE Symp. on Foundations of Computer Science (1978).
2. Chandra, A. K., and L. J. Stockmeyer, "Alternation", Proc. of the 17th IEEE Symposium on Foundations of Computer Science (1976).
3. Cook, S. A., "The complexity of theorem proving procedures", Proc. of 3rd ACM Symp. on Theory of Computing (1971).
4. Cook, "Towards a complexity theory of synchronous parallel computation", TR#141/80, Dept. of Computer Science, Univ. of Toronto (1980).
5. Immerman, N., "Upper and lower bounds for first-order expressibility", 21st Annual Symp. on Foundation of Computer Science (1980).
6. Karp, R. M., and Lipton, R. J., "Some connections between nonuniform and uniform complexity classes", Proc. of the 12th ACM Symposium on Theory of Computing (1980).
7. Kozen, D., "On parallelism in Turing machines", Proc. IEEE Symposium on Foundations of Computer Science (1976).
8. Krapchenko, V.M., "Complexity of realization of a linear function in the class of II-circuits", in Math. Notes Acad. Sci. USSR (1971) pp. 21-23 (English Translation).

9. Mahaney, S., "Sparse complete sets for NP: Solution of a conjecture of Berman and Hartmanis", 21st FOCS (1980).

10. Meyer, A.R. and L. J. Stockmeyer, "The equivalence problem for regular expressions with squaring requires exponential space", Proc. 13th IEEE Symposium on Switching and Automata Theory (1972).

11. Paul, W.J., "A2.5 N-lower bound on the combinatorial complexity of Boolean functions", Proc. 7th ACM Symposium on Theory of Computing (1975).

12. Ruzzo, W.L., "On uniform circuit complexity" 20th FOCS (1979).

13. Savage, S.E., "Computational work and time on finite machines", JACM 19(4), pp. 660-674 (1972).

14. Savage, J.E., "The complexity of computing", John Wiley and Sons (1976).

15. Berman, L. and Hartmanis, J., "On isomorphisms and density of NP and other complete sets", SIAM J. Computing 6(1977) pp. 305-322.