INFORMATION AND CONTROL 55, 40–56 (1982)

Circuit-Size Lower Bounds and Non-reducibility to Sparse Sets

R. KANNAN*

Department of Mathematics, Massachusetts Institute of Technology, Cambridge, Massachusetts 02130

As remarked in Cook ("Towards a Complexity Theory of Synchronous Parallel Computation," Univ. of Toronto, 1980), a nonlinear lower bound on the circuit-size of a language in P or even in NP is not known. The best known published lower bound seems to be due to Paul ("Proceedings, 7th ACM Symposium on Theory of Computing," 1975). In this paper it is shown first that for each nonnegative integer k there is a language L_k in $\Sigma_2 \cap \pi_2$ (of the Meyer and Stockmeyer ("Proceedings, 13th IEEE Symposium on Switching and Automata Theory," 1972) hierarchy) which does not have $O(n^k)$-size circuits. Using the same techniques, one is able to prove several similar results. For example, it is shown that for each nonnegative integer k, there is a language L_k in NP that does not have $O(n^k)$-size uniform circuits. This follows as a corollary of a stronger result shown in the paper. This result like the others to follow is not provable by direct diagonalization. It thus points to the most interesting feature of the techniques used here—by using the polynomial-time hierarchy, they are able to prove results about NP that cannot seem to proved by direct diagonalization. Finally, it is noted that existence of "small circuits" is in suitable contexts equivalent to being reducible to sparse sets. Using this, one is 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.

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 with no feedback. (That is, if the circuit is represented by a directed graph with one vertex corresponding to each gate and an edge (u, v) corresponding to a wire running from the output of gate u to the input of gate v, then the resulting graph is acyclic.) We remark that since all the results are proved for all polynomials, they are still valid if we allow arbitrary fan-out or allow other gates (e.g., EXCLUSIVE OR, etc.), because under these changes circuit-size

* Supported by NSF Grants MCS–8105557 and MCS–77–09906. Some of the results of this paper appeared in the preliminary version "A circuit-size lower bound" (Kannan, 1981). The current paper is the final journal version of this presentation.

0019-9958/82 $2.00

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 C of size $s(C)$ 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 $(i-1)$ alternations beginning with an existential (universal) guess. (Chandra, Kozen, and Stockmeyer, 1981). Formally $\Sigma_i(\pi_i)$ is defined recurcively as follows:

$$\Sigma_0 = P; \qquad \Sigma_1 = NP \quad \text{and for } i \geqslant 1,$$

(1) $\pi_i = \{L : L \text{ is the complement of a language in } \Sigma_i\}$,

(2) $\Sigma_{i+1} = \{L : \exists$ a polynomial $p(\cdot)$ and a language L' in π_i: $L = \{x : \exists y, |y| \leqslant p(|x|), (x, y) \in L'\}\}$

(Meyer and Stockmeyer, 1972). Without loss of generality, we consider all languages to be over $\{0, 1\}$.

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). In the first case we say that C_n accepts the input, in the other it rejects it. We say that a language L *has* $O(f(n))$ *circuits if* $\{C_n\}_{n=1}^{\infty}$ accepts L and there is a constant k such that $s(C_n) \leqslant kf(n)$ for all but finitely many n. A *language has small circuits* if it has $O(n^k)$-size circuits for some fixed k. A class of languages has small circuits if every member of it does. These definitions are made in Karp and Lipton (1980).

Note that even though a language L over $\{0, 1\}$ may have small circuits $\{C_n\}_{n=1}^{\infty}$, the function $n \to C_n$ may not even be recursive. For this reason, the circuit-size of a language is referred to as a *nonuniform* measure of complexity (Borodin, 1977; Cook, 1980) as opposed to the time needed to accept the language on a Turing machine which is called a *uniform* measure. The terminology refers to the fact that in the case of Turing machines, there is one fixed device which uniformly accepts $L \cap \{0, 1\}^n$ for each n, and not so in the case of circuits. In spite of its nonuniformity, circuit-size is an interesting measure since one may prefabricate these devices for values of n ranging from 1 to some practically attained upper bound. Indeed in cryptographic applications, nonuniform rather than uniform complexity is accepted as a measure of how difficult a code is to crack. (Goldwaser and Micali, 1982 and Yao, 1982). In some contexts, however, it is useful to impose some uniformity conditions on circuit-size. A family of circuits $\{C_n\}_{n=1}^{\infty}$ is said to be uniform if the function $1^n \to e(C_n)$ is computable in space $O(\log n)$ (Ruzzo, 1979). In Section 4, we study uniform circuit-size and another measure defined there called provable circuit-size.

If we 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 $NP \neq P$ (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, though it is conjectured by many that NP does not have small circuits. Such an assumption is often made—indeed it is assumed for cryptographic applications that problems like factoring and discrete logarithm (which are not known to be NP-complete) do not have small circuits. (See, e.g., Goldwasser and Micali, 1982; Yao, 1982). The important result of Karp and Lipton (1980) which states that if NP has small circuits, then the polynomial-time hierarchy collapses, may be construed as evidence that the assumption is perhaps valid. Stronger evidence is provided by Mahaney (1980) for the weaker conjecture that NP does not have small p-time constructible circuits. ($\{C_n\}_{n=1}^{\infty}$ is p-time constructible if 1^n—$e(C_n)$ is computable in time polynomial in n.) Mahaney's result essentially says that the weaker conjecture is equivalent to the statement $NP \neq P$. (See Section 5 where it is explained why Mahaney's result may be stated in this form.) On the other hand we now have also some evidence that the conjectures discussed above may not be easy to prove. Wilson (1983) has shown that there exist oracles relative to which every language in NP has *linear* size circuits. Further evidence is provided by the intricacy involved in proving even linear lower bounds on the circuit size for a particular language (Paul, 1975). A number of other papers prove nonlinear lower on the size of *monotone* Boolean circuits (i.e., circuits with only AND and OR gates—no NEGATIONs) needed to compute functions rather than for language acceptance. (Fischer, 1974; Lamagna, 1975; Lamagna and Savage, 1974; Melhorn and Galil, 1976; Paterson, 1975; Pippenger, 1980; Pippenger and Valiant, 1976; Stockmeyer, 1977; Tarjan, 1978; Wegner, 1979; Blum, 1981; Pratt, 1975; and Schnorr, 1974). All of these are fairly low polynomial lower bounds and involve fairly intricate combinatorial arguments. The results of this paper make a beginning towards providing nonlinear circuit-size lower bounds for languages.

An n-input circuit C_n that accepts the set L_n, of the n length strings in a language L, may be thought of as a representation of L_n. Thus circuit-size lower bounds are assertions about the noncompressibility of complete information about L_n in *such* representations. The stronger assertion would be that complete information about L_n may not be compressed into *any* representation of a certain size or less. This stronger assertion may be formalized by a generic statement of the form "L cannot be many-one or Turing reduced in a certain amount of time to any sparse set." In the last section, we show that all our circuit-size lower bounds may be translated into stronger assertions of this kind. Table 1 gives all the results about such reductions proved here as well as related known results.

To illustrate some of the "classical" counting lower-bound techniques (initiated by Shannon (see Savage, 1976), we prove Lemma 0 which is used later.

LEMMA 0. *Given any positive integer k, for all but finitely many n, there is an n-input circuit of size $(3 \cdot n^{2k+2})$ 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. We count the maximum number of n^k size circuits. There are at most 3^{n^k} choices for the at most n^k gates. Each gate has its output connected to at most 2 others, thus there are at most (n^{2k}) ways of connecting up its output. Hence there are at most $3^{n^k} \cdot n^{2k \cdot n^k}$ different circuits of size n^k. This is asymptotically at most $2^{n^{2k}}$. If $\{0,1\}^n = \{x_1, x_2, ..., x_{2^n}\}$, there are $2^{n^{2k+1}}$ distinct subsets of $\{x_1, x_2, ..., x_{n^{2k+1}}\}$. 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 $3 \cdot n^{2k+2}$, because each individual string can be accepted by a circuit of size $(2n)$ and S is an "OR" of n^{2k+1} of these. (The OR of l variables takes l gates to compute.) ∎

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 $k \geqslant 2$,*

$$\Sigma_k = \Sigma_2.$$

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 $O(n^k)$.*

Proof. L_k is described by a first order formula with 3 alternations. The formula simulates a circuit C^* of size n^{2k+5} 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 steps (4)–(6) 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 of lenght n, M accepts iff

(1) $\exists$ an encoding $e(C^*)$ of a circuit C^* of size at most n^{2k+5}

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

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

and

(4) ($\forall$ encodings $e(C)$ of circuits with $e(C) \leqslant e(C^*)$ (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 that for each n sufficiently large, $L_k \cap \{0, 1\}^n$ cannot be accepted by a circuit of size n^{k+1} or less. Hence it does not have $O(n^k)$ 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 $O(n^k)$ 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 is 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 $O(n^k)$ size circuits for any k. L is of course in $\Sigma_2 \cap \pi_2$ and thus $L = L_k$ proves the theorem. ∎

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

$$\text{SAT} = \{x \mid x \text{ is an encoding of a satisfiable Boolean formula}\}$$

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 $O(n^k)$ circuits for some k fixed, then $NP \neq P$. However, since the hypothesis of the above statement would seem to be rather more unlikely than the conclusion, it is not terribly useful.

3. Related Results

A language $L \in \{0, 1\}^*$ is sparse if there is a polynomial $p(\cdot)$ such that $|L \cap \{0, 1\}^n| \leqslant p(n)$. 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 $O(n^k)$ circuits. If there were 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 $O(n^k)$ circuits. But unfortunately, there is no sparse *NP*-complete language unless $NP = P$ (Mahaney, 1980) and we have to content 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 $O(n^k)$ 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, $i \leqslant j \leqslant n^{2k+3}$ such that*

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

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, $1 \leqslant j_0 \leqslant n^{2k+3}$ and

(1) $\exists y = y^{(1)}y^{(2)} \cdots y^{(n^{2k+3})}$ 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) $(\forall z = z^{(1)} \cdots z^{(n^{2k+3})}$,

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

(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 $\vee$ stands for OR and $\wedge$ for AND. First, M has at most 2 alternations and is a Σ_3 machine. Second, 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)–(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 $O(n^k)$ circuits.

Step (0) ensures that the language is sparse.

The complement language results when we change step (0) to read "$x \neq x_{j_0}$ for any j_0, $1 \leqslant j_0 \leqslant n^{2k+3}$ or" and we change step 7 to read "$y^{(j_0)} = 0$." Thus the complement is also in Σ_3. ∎

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 $\Sigma_i(\pi_i)$ machine in time $O(f(n))$. 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 $k \geqslant 1$ 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 $O(f(n))$ 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. This is outlined below.

DEFINITION. We say that a function $f(n) \geqslant n$ is time-constructible if a deterministic multitape $O(f(n))$-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 diagonalize over all n-input circuits encodable by a string of length $\sqrt{f(n)}$ or less, $\sqrt{f(n)}$ also being superpolynomial 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 $\{0,1\}^n = \{x_1,...,x_{2^n}\}$ and we have decided what to do on inputs $x_1,...,x_i$. On input x_{i+1}, our machine finds out first, what it decided to do on $x_1,...,x_i$. Next, it enumerates all circuits of encoding size $\sqrt{f(n)}$ or less. Among the circuits that agree with our machine on $x_1,...,x_i$, 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 $\{x_1,...,x_{\sqrt{f(n)}}\}$, our machine will differ from the circuit. It is not difficult to reckon the space requirements to complet 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 $\Sigma_2^{n^l} \cap \pi_2^{n^l}$. For any integer N, we assume that $\{0,1\}^N = \{x_1,...,x_{2^N}\}$, where $x_1 < x_2 < \cdots < x_{2^N}$ ($<$ reads lexicographically less than). We further assume $f(n) \leqslant 2^{n/20}$, 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

$$f(n); \lceil (f(n))^{1/l} \rceil = m \quad \text{(say)}.$$

(2) Accept x iff $O^m x$ 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 $O((\log f(n))^s)$ steps for some s, step 1 takes time $O(f(n))$. Since L_1 is in $\Sigma_2^{n^l}$, step 2 can also be done in $\Sigma_2^{f(n)}$ 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 $\Sigma_2^{n^l}$.) We now wish to claim that L does not have $O((f(n))^{1/l})$-size circuits. For suppose it did. Then, suppose N is any integer such that there exists an n with $N = \lceil (f(n))^{1/l} \rceil + n$. 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 that $f(n) < 2^{n/20}$, $n > 5 \log N$ 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 0's accepts $L_1 \cap \{0,1\}^N$. Hence, if L has $k \cdot f(n)^{1/l}$ size circuits, then L_1 has $O(n)$ 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 $O(n)$ circuits.) Since $f(n)$ is superpolynomial, so is $(f(n))^{1/l}$ and hence we have proved the lemma. ∎

What we have shown is actually the following:

THEOREM 4. *There is a universal constant l such that for any time-constructible function $f(\cdot)$ satisfying $n^l \leqslant f(n) \leqslant 2^{n/20} \forall n$, there is a language in $\Sigma_2^{f(n)} \cap \pi_2^{f(n)}$ that does not have $O((f(n))^{1/l})$-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 $(k-1)$ 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 $O(f(n))$ 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} or π_{k+1} has small Σ_k formulas, then $\Sigma_j = \Sigma_{k+2} j \geqslant k+3$.

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

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

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

We will 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 though $\{C_n\}_{n=1}^{\infty}$ may accept a language L, the function $1^n \to e(C_n)$ 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 $O(\log n)$ 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 $L' = \{1^n \# e(C_n) \mid n = 1, 2, \ldots\}$ is in NP, where e is some natural encoding.

The reason for the terminology is as follows: the language L' being in NP implies that we may easily prove that C_n is the correct circuit by checking that $1^n \# C_n$ is in L'. Uniformity and p-time constructibility defined in Section 5 are stronger notions; there the function $1^n \to C_n$ is required to be log space and polynomial-time computable, respectively. Note that the concept of provable circuit-size has the following practical use: If we have prefabricated provable circuits $\{C_n\}$ that accept a language L, we may also include (as documentation) a short certificate that C_n performs the correct function.

A language L has provable $O(f(n))$-size circuits if $\{C_n\}_{n=1}^{\infty}$ provably accepts L and size $(C_n) \leqslant cf(n)$, c a constant for all but finitely many n. We then define a language (a class) of languages having provable small circuits, etc., as before. Along the lines of Karp and Lipton, one can show

THEOREM 5. *If NP has provable small circuits, then* $\Sigma_k = NP$ *for all* $k \geqslant 1$.

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 integer* $k \geqslant 1$, *there is a language* L_k *in NP such that* L_k *does not have probable* $O(n^k)$*-size circuits.*

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

Remark 3. Neither Theorem 6 nor the corollary of it is probable by direct diagonalization: For example, in the case of uniform circuits, since the log space machine can take $O(n^l)$ time for any fixed l, one nondeterministic polynomial-time bounded TM cannot seem to simulate all the log space machines to diagonalize 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 in a certain fashion into length $O(n^k)$ by *any* non-deterministic polynomial-time machine. This remark is related to Theorem 9, where reductions to sparse sets are discussed.

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 suppse *NP* has small circuits. Say L is in *NP* and has circuits $\{C_n\}_{n=1}^{\infty}$ such that $|e(C_n)| \leqslant n^k$. Suppose $\{0,1\}^n = \{x_1,\ldots,x_{2^n}\}$. Define a sparse set S as follows: S contains x_i if and only if $i \leqslant n^k$ 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 $e(C_n)$. It is well known (Savage, 1976) that P has small *p*-time constructible circuits. 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 with special properties. To make this more precise, we make the following definition:

We say that a language S over an alphabet Σ is an *exact sparse set* if there is a time-constructible function $p(n)$ which is bounded above by a polynomial such that $|S \cap \Sigma^n| = p(n)$. In the terminology of Hartmanis and Mahaney (1980), S is a sparse set with an easily computable census.

Lemma 5. *NP has small provable circuits iff NP is p-time Turing reducible to an exact sparse set which is itself in NP.*

Proof. Suppose $L \in NP$ has small provable circuits $\{C_n\}$. Let $\{1^n \# e(C_n) \mid n = 1,\ldots,\} = L'$. L' is in *NP*. We may assume after padding, if necessary, that $|e(C_n)| = c(n)$, where $c(n)$ is time constructible. Consider the sparse language S (again $\{0,1\}^n = \{x_1,\ldots,x_{2^n}\}$) defined by $S \cap \{0,1\}^{n+1} = \{x_i z \mid 1 \leqslant i \leqslant c(n),\ z = 0 \text{ or } 1 \text{ and the } i\text{th digit of } e(C_n) \text{ is } z\}$. 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 a language L in *NP* is *p*-time reducible to an exact sparse set S in *NP* with $|S \cap \{0,1\}^n| = c(n)$, c a time constructible

polynomial bounded function. Let M be a $p(n)$ time bounded deterministic *TM* (p a polynomial) which using an oracle for S accepts L. M clearly consults the oracle only on strings of length $p(n)$ or less on an input of length n. There are precisely $\sum_{i=1}^{p(n)} c(i) = P(n)$ such strings and $P(n)$ is bounded by a polynomial. We construct a circuit C_n: On any input of length n, C_n simulates M (Savage, 1976) and whenever an oracle question is asked, it tests membership of the queried string in S by "table-look-up." Since S is sparse, C_n has size bounded by a polynomial in n. Further the construction of C_n can be made canonical given the $P(n)$ strings of S and M, i.e., if S is known, $L' = \{1^n \# e(C_n) : n = 1, 2, ...,\}$ is in P. A NDTM recognizes L' as follows: first, it checks the format. Then guesses the $P(n)$ strings in S of length n or less and checks by running the polynomial-time bounded NDTM accepting S that all these strings are in S. Finally, it checks that C_n is correctly constructed using M and the guessed strings. If all of these hold, it accepts. Clearly, this runs in polynomial-time and hence, L' is in NP proving that L has provable small circuits. ∎

We observe that the stipulation that the exact sparse set be in NP is crucial. By modifying Meyer's construction, 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 suh that L_k cannot be many–one reduced in $O(n^k)$ deterministic time to a language of sparcity $O(n^k)$.*

Remark. Note again that direct diagonalization 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 $\Sigma_2 = NP$. Thus for each l, NP has a language L_l that does not have $O(n^l)$-size circuits (by Theorem 2). This would be contradicted if every language in NP were reducible in $O(n^k)$ time to a language of sparsity $O(n^k)$ for a fixed k (because then for any language L in NP, there is a $O(n^k)$-deterministic time reduction f of L to the language S of sparsity $O(n^k)$. Thus any string x belongs to L iff $f(x) \in S$. $|f(x)| = O(|x|^k)$. There is a $O(|x|^{k^5})$-size circuit that by "table-look-up" checks if $f(x)$ is in S. There is a $O(|x|^{3k})$ size circuit that on input x produces output $f(x)$. Thus L has $O(n^{k^5+3k})$-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 $O(n^k)$ to a language of sparsity $O(n^k)$.*

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 $O(n^k)$ 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 $O(n^l)$ for a fixed l as argued earlier in this section.

We can also use "padding" arguments to prove results about super-polynomial 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.*

As remarked earlier, 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 superpolynomial time.

Proof. Suppose the theorem is false. Then consider the "universal" language U for NTIME($f(n)/n$):

$$U = \{M \# x \mid M \text{ is a 2 tape NDTM and } M \text{ accepts } x \text{ in time } f(n)/n\}.$$

U is accepted by a machine that constructs $f(n)/n$ in time $O(f(n))$ and then runs M on x for $f(n)/n$ steps, accepts x iff M does. Simulating M for 1 step takes time at most $|M|$ (to look-up the next move) and hence U runs in time $O(f(n))$. Since $f(n)$ is super-polynomial, $f(n)/n$ is too. The language L_k of Theorem 7 is accepted by a 2-tape NDTM in time $O(n^k)$ (Book, Greibach, and Wegbreit, 1970) and thus is linear (deterministic) time many–one reducible to U. By hypothesis there exists a many–one polynomial-time reduction g of U to a sparse set S. Suppose l is such that S is of sparsity $O(n^l)$ and g can be computed in time $O(n^l)$. Then $L_{l+1}, L_{l+2}, \ldots$, can all be

TABLE 1

Reduction to sparse sets (A)	$NP \subseteq P^{SP}$	$NP \subseteq P^{SP}$; $SP \in NP$, SP exact	$NP \leqslant^{m}_{p\text{-time}} SP$
Equivalent circuit Property (B)	NP has small circuits (Meyer)	NP has provable small circuits (Lemma 5)	NP has p-time constructible small circuits (Mahaney)
Hierarchy collapses to under (A) or (B)	Σ_2 (Karp, Lipton, and Sipser)	NP (Theorem 5)	P (Mahaney)
Result on non-reducibility to sparse sets	$\forall k$, $\Sigma_2 \cap \pi_2 \not\subseteq \mathrm{DTIME}(n^k)^{SP}$ $\lvert SP\rvert \in O(n^k)$ (Theorem 8)	$\forall k$, $NP \not\subseteq \mathrm{DTIME}(n^k)^{SP}$ $SP \in NP$; $\lvert SP\rvert = n^k$	$\forall k$, $NP \not\leqslant^{m}_{\text{time } O(n^k)} SP$ $\lvert SP\rvert \in O(n^k)$ (Theorem 7)
Circuit-size lower bound	$\forall k$, $\Sigma_2 \cap \pi_2$ does not have $O(n^k)$-size circuits (Theorem 2)	$\forall k$, NP does not have $O(n^k)$ provable circuits (Theorem 6)	
Non-reducibility to sparse sets (superpoly. classes)	$\forall$ super-poly., constructible $f(\cdot)$ $\Sigma_2^{\mathrm{TIME}(f(n))} \cap \pi_2^{\mathrm{TIME}(f(n))} \not\subseteq P^{SP}$		$\forall$ super-poly constructible $f(n)$ $\exists L \in \mathrm{NTIME}(f(n))$ s.t. $L \not\leqslant^{m}_{p\text{-time}} S$ for any sparse set S (Theorem 10)

reduced in time $O(n^l)$ to S contradicting Theorem 7. This proves Theorem 10.

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

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

denotes the class of languages Turing reducible in p-time to some sparse set S over the alphabet Σ in NP s.t. $|S \cap \Sigma^n| = n^k$.

$$A \leqslant^m_{n^l\text{-time}} B$$

denotes, of course, that the language A is many-to-one reducible in time $O(n^l)$ to the language B.

Finally we remind the reader that a sparse set S over Σ is said to be an exact sparse set if there is a time-constructible polynomial-bounded function $p(n)$ such that $|S \cap \Sigma^n| = p(n)$. Table 1 uses this notation.

6. Relativization

All the results in this paper relativize to arbitrary oracles using the relativized circuit model introduced by Wilson (1983). The results of that paper along with our results then point to the difficulty of strengthening some of the theorems in this paper. For a full discussion of this, the reader is referred to Wilson's paper.

Acknowledgments

I would like to thank Professor J. Hartmanis for a helpful discussion on sparse sets. I thank the referees for many valuable suggestions.

References

1. Adelman, L. (1978), Two theorems on random polynomial-time, *in* "Proceedings of the 19th IEEE Symp. on Foundations of Computer Science."
2. Berman, L., and Hartmanis, J. (1977), On isomorphisms and density of *NP* and other complete sets, *SIAM J. Comput.* **6**, 305–322.

3. BLUM, N. (1981), A $\Omega(n^{4/3})$ on the monotone network complexity of nth degree convolution, *in* "Foundations of Computer Science."
4. BOOK, R. V., GREIBACH, S. A., AND WEGBREIT, B. (1970), Time and tape bounded Turing acceptors and AFL's, *J. Comput. System Sci.* **4**.
5. BORODIN, A. (1977), On relating time and space to size and depth, *SIAM J. Comput.* 733–744.
6. CHANDRA, A. K., KOZEN, D., AND STOCKMEYER, L. J. (1981), Alternation, *J. Assoc. Comput. Mach.* **28**.
7. COOK, S. A. (1971), The complexity of theorem proving procedures, *in* "Proceedings, 3rd ACM Symp. on Theory of Computing."
8. COOK, S. A. (1980), "Towards a Complexity Theory of Synchronous Parallel Computation," TR No. 141/80, Dept. of Computer Sci., Univ. of Toronto.
9. FISCHER, M. J. (1974), The complexity of negation-limited networks, *in* "*GI*-Fachtagung Automatentheorie und Formale Sprachen, Kaiserslautern," Springer Lecture Notes, No. 33, pp. 71–82.
10. GOLDWASSER, S., AND MICALI, S. (1982), Probabilistic encryption and how to play mental poker keeping secret all partial information, *in* "Proceedings of the 14th ACM Symposium on Theory of Computing."
11. HARTMANIS, J., AND MAHANEY, S. R. (1980), "On Census Complexity and Sparseness of *NP*-Complete Sets," Cornell Univ., Computer Sci. Dept., TR 80–416, April.
12. IMMERMAN, N. (1980), Upper and lower bounds for first-order expressibility, *in* "Proceedings, 21st Annual Symp. on Foundation of Computer Science."
13. KANNAN, R. (1981), A circuit-size lower bound, *in* "Proceedings, 22nd IEEE Symposium on Foundations of Computer Science."
14. KARP, R. M., AND LIPTON, R. J. (1980), Some connections between nonuniform and uniform complexity classes, *in* "Proceedings of the 12th ACM Symposium on Theory of Computing."
15. KRAPCHENKO, V. M. (1971), Complexity of realization of a linear function in the class of II-circuits, *Math. Notes* 21–23. [English]
16. LAMAGNA, E. A. (1975), "The Complexity of Monotone Functions," Ph.D. thesis, Computer Science Program, Brown Univ., Providence, R. I.
17. LAMAGNA, E. A., AND SAVAGE, J. E. (1974), Combinatorial complexity of some monotone functions, *in* "Proceedings, 15th SWAT Conference," New Orleans, pp. 140–144.
18. MAHANEY, S. (1980), Sparse complete sets for *NP*: Solution of a conjecture of Berman and Hartmanis, *in* "Proceedings, 21st FOCS."
19. MEHLHORN, K., AND GALIL, Z. (1976), Monotone switching circuits and Boolean matrix product, *Computing* **16**, 99–111.
20. MEYER, A. R., AND STOCKMEYER, L. J. (1972), The equivalence problem for regular expressions with squaring requires exponential space, *in* "Proceedings, 13th IEEE Symposium on Switching and Automata Theory."
21. PATERSON, M. S. (1975), Complexity of monotone networks for Boolean matrix products, *Theoret. Comput. Sci.* **1**, 13–20.
22. PAUL, W. J. (1975), A 2.5 N-lower bound on the combinatorial complexity of Boolean functions, *in* "Proceedings, 7th ACM Symposium on Theory of Computing."
23. PIPPENGER, N. (1980), On another Boolean matrix, *Theoret. Comput. Sci.* **11**, 45–56.
24. PIPPENGER, N., AND VALIANT, L. G. (1976), Shifting graphs and their applications, *J. Assoc. Comput. Mach.* **23**, 423–432.
25. PRATT, V. R. (1975), The power of negative thinking in multiplying Boolean matrices, *SIAM J. Comput.* **4**, 326–330.
26. RUZZO, W. L. (1979), On uniform circuit complexity, *in* "Proceedings, 20th FOCS."

27. SAVAGE, J. E. (1972), Computational work and time on finite machines, *J. Assoc. Comput. Mach.* **19** (4), 660–674.
28. SAVAGE, J. E. (1976), "The Complexity of Computing," Wiley, New York.
29. SCHNORR, C. P. (1974), Zwei lineare untere Schranken für die Komplexität Boolescher Funktionen, *Computing* **13**, 155–171.
30. STOCKMEYER, L. J. (1977), On the combinational complexity of certain symmetric Boolean functions," *Math. Systems Theory* **10**, 323–336.
31. TARJAN, R. E. (1978), Complexity of monotone networks for computing conjunctions, *Ann. Discrete Math.* **2**, 121–133.
32. WEGENER, I. (1979), Switching functions whose monotone complexity is nearly quadratic, *Theoret. Comput. Sci.* **9**, 83–97.
33. WILSON, C. B. (1983), Relativized circuit complexity, preprint, Computer Science, Univ. of Toronto.
34. YAO, A. (1982), Theory and applications of trapdoor functions, *in* "Proceedings, 23rd Symp. on Foundations of Computer Science."