On the Decomposability of NC and AC

Christopher B. Wilson

CIS-TR-88-17
November 29, 1988

DEPARTMENT OF COMPUTER AND INFORMATION SCIENCE
UNIVERSITY OF OREGON

On the Decomposability of NC and AC

Christopher B. Wilson*
Department of Computer and Information Science
University of Oregon
Eugene, OR 97403 USA

Abstract

It is shown for rationals $a, b \geq 1$ that $NC_a^{NC_b} = NC_{a+b-1}$. As a consequence, if, for some $k \geq 1$ and $\epsilon > 0$, $NC_k = NC_{k+\epsilon}$, then $NC_k = NC$. A similar development can be applied to circuits with unbounded fan-in. It is seen that $AC_a^{AC_b} = AC_{a+b}$, $AC_a^{NC_b} = NC_{a+b}$, and $NC_a^{AC_b} = AC_{a+b-1}$. This shows for $a \geq 0$ and $b \geq 1$ that $AC_a^{AC_b} = NC_{a+1}^{AC_b}$ and $AC_a^{NC_b} = NC_{a+1}^{NC_b}$. We then construct an oracle A for which $\forall k$, $NC_k^A \subset AC_k^A$ and, in fact, $AC_k^A - NC_{k+\epsilon}^A \neq \emptyset$ for any $\epsilon < 1$. Similarly, there is an A so that $\forall k$ and $\epsilon < 1$, $NC_{k+1}^A - AC_{k+\epsilon}^A \neq \emptyset$, and hence $AC_k^A \subset NC_{k+1}^A$. Combining, this yields an A such that, for all k and $0 < \epsilon < 1$, the classes AC_k^A and $NC_{k+\epsilon}^A$ are incomparable.

1 Introduction

In recent years the class NC has been established as one of the preferred characterizations of those problems with very fast parallel algorithms using a reasonable amount of hardware. It is a remarkably robust class, being invariant when defined on quite different models of computation. The most common models are PRAMs with shared memory and uniform Boolean circuit families (see [5,7]). As we are concerned here with some detailed structural theorems about the NC hierarchy, we will use the more basic model, that of circuits. The structural issue in question is whether it is possible to decompose any level of NC into components involving lower levels. We answer this affirmatively.

NC_k is defined to be the class of languages which are accepted by uniform families of circuits with bounded fan-in whose *size* grows polynomially with the length n of the input and whose *depth* grows proportional to $\log^k n$. The classes of particular interest are NC_a supplied with an oracle from NC_b. First we see that this is contained in NC_{a+b-1} so long as $b \geq 1$. Then for $a \geq 1$ and $b \geq 0$, NC_{a+b-1} can be expressed as NC_a relative to NC_b.

*Partially supported by NSF grant CCR-8810051

This provides characterizations of NC similar to those available for the polynomial time hierarchy PH. The polynomial hierarchy was originally defined in terms of one complexity class relative to another, and in [10] there was provided a characterization of each level in terms of polynomially bounded alternating quantifiers applied to a polynomial time computable predicate. Here NC is defined in terms of increasing depth, and we show that there is an equivalent characterization in terms of one level of NC relative to another.

Using the above development, we get as a simple corollary a result first proved in [4]: for integers $k \geq 1$ if $NC_k = NC_{k+1}$, then $NC_k = NC$. This provides a structural analogy with the polynomial hierarchy: if $\Sigma_k^P = \Sigma_{k+1}^P$, then $\Sigma_k^P = PH$. However, the NC hierarchy need not be discrete since we can surely examine NC_k for k rational or real (here rational for reasons of constructibility). As a corollary of the main result are able to show for rationals $k \geq 1$ and $\epsilon \geq 0$ that $NC_k = NC_{k+\epsilon}$ implies $NC_k = NC$.

The class AC_k is the same as NC_k except that unbounded fan-in gates are allowed. In section 6 these techniques above are applied to the AC_k classes. It is shown that $AC_a^{AC_b} = AC_{a+b}$, $AC_a^{NC_b} = NC_{a+b}$, and $NC_a^{AC_b} = AC_{a+b-1}$. This has several interesting consequences, one being that the AC hierarchy may collapse in the same way as the NC hierarchy: for rationals $k \geq 0$ and $\epsilon > 0$, $k + \epsilon \geq 1$, if $AC_k = AC_{k+\epsilon}$, then $AC_k = AC$. Another consequence is for rationals $a \geq 0$ and $b \geq 1$ that $AC_a^{AC_b} = NC_{a+1}^{AC_b}$ and $AC_a^{NC_b} = NC_{a+1}^{NC_b}$. That is, an NC_{k+1} reduction provides no more power than an AC_k reduction when acting on any level of NC or AC.

In [12] there was constructed an oracle A such that $\forall k$, $NC_k^A \subset NC_{k+1}^A$. In section 7 we take that development considerably further. The same oracle A shows that $\forall k$ and $\epsilon < 1$, $AC_k^A - NC_{k+\epsilon}^A \neq \emptyset$. So in particular, $NC_k^A \subset AC_k^A$. A different approach to oracle based languages allows us to construct an A for which $\forall k$ and $\epsilon < 1$, $NC_{k+1}^A - AC_{k+\epsilon}^A \neq \emptyset$, and hence $AC_k^A \subset NC_{k+1}^A$. The first construction allows an unbounded fan-in circuit to query a long sequence of large strings. The second construction takes advantage of the fact that a bounded fan-in circuit is charged less for a short query than for a long one, which is not true in the unbounded case. The two constructions can be combined to produce an A such that for any k and $0 < \epsilon < 1$, AC_k^A and $NC_{k+\epsilon}^A$ are incomparable.

2 Definitions and Notation

A *Boolean circuit* is an acyclic directed graph whose nodes are labelled with an operator. Nodes of indegree 0 are labelled as either *input* or *constant* gates, and those of outdegree 0 are *output* gates. Nodes of indegree 1 are *negation* or *identity* gates, while those of indegree 2 are *and* or *or* gates. No nodes in the graph will be allowed to have indegree greater than 2. A node may be both an output gate and perform an operation. Since we are primarily interested in deciding set membership, the circuits of use to us have only a single output gate. The circuit *accepts* an input string if the length of the string in binary is the same as the number of input gates of the circuit and the circuit outputs the value 1 when given the string on its input gates.

The *size* of a circuit is the number of nodes it contains and its *depth* is the length of the longest directed path in the graph. Intuitively, the size can be thought of as measuring the use of the hardware resource while the depth is a measure of parallel time. A *circuit family* $\{\alpha_n\}$ accepts a set L if, for all n, α_n has n input nodes and accepts only those strings in L of length n. The circuit family has size $s(n)$ and depth $d(n)$ if $size(\alpha_n) = O(s(n))$ and $depth(\alpha_n) = O(d(n))$.

Another requirement put on circuit families is that of uniformity. This can be described in several ways, as covered in [9].

Definition 1 *A circuit family $\{\alpha_n\}$ is $u(n)$-uniform if there is a deterministic Turing machine which on any input of length n outputs an encoding of α_n using $O(u(n))$ workspace.*

A circuit family is U_B uniform if it is $depth(\alpha_n)$-uniform, and it is U_{BC} uniform if it is $\log(size(\alpha_n))$-uniform. Other more complicated but technically more appealing notions of uniformity involve determining the structure of the circuit's connections on an alternating Turing machine [9]. We will adopt U_{BC} uniformity in the definitions below.

Definition 2 $NC_k = U(\log n)\text{-}DEPTH(\log^k n)$

Definition 3 $NC = \bigcup_{k=0}^{\infty} NC_k$

There is a slight deviation in notation here. NC_k is normally written as NC^k – here we will leave room for an oracle as superscript. Also notice that the size of the circuits is not explicitly restricted to be polynomial. This, however, follows automatically from the uniformity condition. A Turing machine operating in $O(\log n)$ space will only run for polynomial time (if it is going to halt), so will only be able to output descriptions of polynomial size circuits.

The notion of allowing an NC circuit access to an oracle has been addressed in [5,12]. The circuit is allowed *oracle gates* through which it can determine the membership of a string in the oracle set. An oracle gate which has k input bits (sufficient for determining membership in the oracle of any string of length k) is defined to have *size* k and *depth* $\lceil \log_2 k \rceil$. If X is a set, we denote NC_k relative to X by NC_k^X. If $\mathcal{C}$ is a class of sets, then $NC_k^{\mathcal{C}}$ is $\bigcup_{X \in \mathcal{C}} NC_k^X$. The uniformity condition will be unaffected by the presence of an oracle. The Turing machine which acts as a constructor of the circuit family will not have access to the oracle. This issue is discussed in greater detail in [12].

This is directly analogous to the notion of NC_1-reducibility defined in [5], where the reduction is from function to function.

Definition 4 *A is NC_1-reducible to B, $A \leq^{NC_1} B$, if and only if $A \in NC_1^B$.*

An unbounded fan-in circuit, which we will abbreviate hereafter as a UBF circuit, is a Boolean circuit as above but with no restriction on the indegree of any node. A class directly analogous to NC can be defined on this model.

Definition 5 $AC_k = U(\log n)\text{-}UBF\,DEPTH(\log^k n)$

Definition 6 $AC = \bigcup_{k=0}^{\infty} AC_k$

It is not too hard to see that for any $k \geq 0$, $NC_k \subseteq AC_k \subseteq NC_{k+1}$. A discussion of this class and related issues can be found in [5]. It is also known that the $PARITY$ function is not in AC_0 ([6]). This separates AC_0 from NC_1 and AC_1, but is the only known such separation, aside from the obvious $NC_0 \neq AC_0$.

To handle an oracle in AC_k, we will allow oracle gates as above. As in [3], however, the size and depth of this gate is 1. (This is similar also to [4].) Intuitively, this adheres to the spirit of unbounded fan-in, as we can also compute the *and* of k bits in depth 1. On a bounded fan-in model, the k-bit *and* requires depth $\log k$, as it does to determine membership of a k-bit string in an oracle.

All logarithms are to the base two. In fact, by $\log n$ we will mean the function $\max(1, \lceil \log_2 n \rceil)$.

3 NC Oracles

Initially we address the issue of allowing NC to have oracles from NC. That is, we want to know how complex a set in $NC_a^{NC_b}$ can be. This has been partially addressed in [5], where it is seen that NC_k is closed under $\leq^{NC_1}$, which is to say $NC_1^{NC_k} = NC_k$. Adapting the proof of that result yields $NC_a^{NC_b} \subseteq NC_{ab}$. This can be considerably sharpened. First, however, we need a technical lemma.

Lemma 1 *Given $b \geq 1$ and integers $x_1, x_2, \ldots, x_{N^a}$ such that $\forall i, 0 \leq x_i \leq N$ and $\sum_{i=1}^{N^a} x_i \leq N^a$, the maximum value $\sum_{i=1}^{N^a} x_i^b$ can obtain is $N^{a+b-1} + N^b$.*

proof

Suppose $b \geq 1$. We can see that $\sum_{i=1}^{k} x_i^b$ is maximized subject to the given constraints when $x_1 = \cdots = x_k = N$ and $x_{k+1} = \cdots = x_{N^a} = 0$, for appropriate k. That follows from

claim: if $x \geq y - \epsilon \geq 0$ and $\epsilon \geq 0$, then $(x+\epsilon)^b + (y-\epsilon)^b \geq x^b + y^b$.

The claim is seen by examining $f(z) = (z+\epsilon)^b - z^b$. A look at the derivative $f'(z) = b((z+\epsilon)^{b-1} - z^{b-1})$ indicates, for positive z, that f is an increasing function when $b \geq 1$. Thus, since $x \geq y - \epsilon \geq 0$, $f(x) \geq f(y-\epsilon)$ and the claim follows.

In particular, the claim says for integers $x \geq y \geq 1$ that $(x+1)^b + (y-1)^b$ is no less than $x^b + y^b$. The sum is maximized when we first make x_1 as large as possible, then x_2, and so on until we would violate the constraint $\sum_{i=1}^{k} x_i \leq N^a$. Since the maximum value obtainable by an x_i is N, this will happen when $k = \lfloor \frac{N^a}{N} \rfloor$. Since a may not be an integer, there may be some remainder: $x_{k+1} = \lfloor N^a \rfloor - kN$. The x_{k+2} through x_{N^a} will all be zero. When the x_i are chosen as above to maximize the sum,

$$\begin{aligned}\sum_{i=1}^{k+1} x_i^b &= kN^b + (\lfloor N^a \rfloor - kN)^b \\ &\leq \frac{N^a}{N} N^b + (N^a - (\frac{N^a}{N} - 1)N)^b \\ &= N^{a+b-1} + N^b.\end{aligned}$$

□

Theorem 2 *For $b \geq 1$, $NC_a^{NC_b} \subseteq NC_k$, where $k = \max(a+b-1, b)$.*

proof

Consider an $O(\log^a n)$ depth circuit with queries made to a language in NC_b. Look at any path from an input to the output. It has length $O(\log^a n)$, so the series of queries $q_1, \ldots, q_k$ made on that path must satisfy $\sum_{i=1}^k \log |q_i| \leq c \log^a n$. Each query q_i has length at most polynomial in n, so $\log |q_i| \leq d \log n$. (Assume here that $c = d^a$: if this is not true, either c or d can be increased to ensure it.) We will replace each query with an NC_b circuit. In the new circuit, after replacement, the length of the path will be the maximum of $O(\log^a n)$ (from the part of the path excluding the queries) and $\sum_{i=1}^k e \log^b |q_i|$ (caused by replacing the query by an NC_b circuit). The latter value has maximum value $e\,(d \log n)^{a+b-1} + e\,(d \log n)^b$, seen by letting $x_i = \log |q_i|$ and $N = d \log n$ and applying the previous lemma (notice that $N^a = c \log^a n$). Since $b \geq 1$, we have $a + b - 1 \geq a$, so the depth of the entire path becomes $O((\log n)^{a+b-1} + (\log n)^b)$.
□

4 Splitting Circuits

To show a containment in the other direction, we must take a uniform circuit family and be able to divide each circuit into arbitrarily sized uniform subcircuits.

Theorem 3 $NC_{a+b-1} \subseteq NC_a^{NC_b}$ *for rationals $a \geq 1$ and $b \geq 0$.*

proof

Let us assume that $b \geq 1$. Otherwise, if $0 \leq b < 1$, then trivially $NC_{a+b-1} \subseteq NC_a \subseteq NC_a^{NC_b}$.

Given a language $S \in NC_{a+b-1}$, look at the family $\{\alpha_n\}$ of circuits accepting it. The α_n are generable by a Turing machine M_α in space $O(\log n)$ from any input of length n, and each α_n has depth $c(\log n)^{a+b-1}$ for some constant c. Suppose that M_α on an input of size n outputs a sequence of p tuples of the form (i, l_i, r_i, t_i), where l_i and r_i are the inputs to gate i and t_i indicates the type of gate i. In a fixed α_n, let *dist(i)* be the maximum distance of gate i from an input gate.

We will demonstrate the existence of an NC_a circuit family using an oracle from NC_b and behaving the same as $\{\alpha_n\}$. This will show that $S \in NC_a^{NC_b}$. The idea is to divide each α_n into $c\log^{a-1} n$ levels each of depth $\log^b n$. We then provide the computation of each level as an oracle. Since each query will have at most polynomial length, the depth of each query will be $O(\log n)$. There are at most $c\log^{a-1} n$ of them in any series, so the resulting relativized circuit has depth $O(\log^a n)$. This NC_a circuit family is easily seen to be $\log n$-uniform.

A problem arises with the NC_b subcircuits. It is not at all clear that they are uniform since generating them requires determining the depth of each node in α_n. This seems to require transitive closure, a problem complete for NL and not known to be solvable in $O(\log n)$ deterministic space. Instead we use a technique found in theorem 12 of [1] to construct from α_n an equivalent circuit β_n with a very regular structure which can be easily subdivided.

Let $K = c(\log n)^{a+b-1}$ be an upper bound to the depth of α_n. Also let $p = p(n)$ bound the number of gates in α_n. The circuit β_n consists of K levels, each of p gates, one for each gate of α_n. Each level takes all its inputs from the previous level. Each level L_j consists of gates $g_{j,1}, g_{j,2}, \ldots, g_{j,p}$.

In L_0, if gate i is an input node, then gate $g_{0,i}$ will be a single gate, labelled as an input gate. Otherwise, gate $g_{0,i}$ will be a constant 0 gate.

At level L_j, for $j \geq 1$, each gate $g_{j,i}$ will take as inputs g_{j-1,l_i} and g_{j-1,r_i}. That gate will compute $g_{j-1,l_i} \diamond g_{j-1,r_i}$, where $\diamond$ is the operation determined by t_i. Other gates of indegree 1 and 0 can be handled in an entirely similar manner. However, if i is an input gate in α_n, then $g_{j,i}$ will perform the *identity* operation on input $g_{j-1,i}$. This way, the original input values are passed from level to level.

Figure 1(a) illustrates a sample circuit. Figure 1(b) shows the transformation of that circuit. The circled nodes indicate which nodes are relevant in the simulation of the original circuit.

The fact that β_n performs the same as α_n can be shown by induction on the depth of the gates of α_n. This follows from

claim: If *dist(i)* $\leq j$, then gate $g_{j,i}$ of β_n outputs the same value as gate i of α_n.

The claim is seen by induction on j. If $dist(i) = 0$, then i is an input and gate $g_{0,i}$ provides that input. Suppose then that $dist(i) \leq j$ for $j \geq 1$. If i is an input gate in α_n, then by definition $g_{j,i}$ provides that value. For i not an input, look at its left l_i and right r_i predecessors. Since $dist(l_i) \leq j-1$ and $dist(r_i) \leq j-1$, by the inductive hypothesis g_{j-1,l_i} and g_{j-1,r_i} yield the same values as l_i and r_i in α_n. Gate $g_{j,i}$ computes the same operation on those values as gate i in α_n does.

The circuit family $\{\beta_n\}$ thus accepts S. Each β_n is split into subcircuits B_0, B_1, ..., $B_{c\log^{a-1} n}$. B_0 is simply L_0. Subcircuit B_l consists of levels $L_{(l-1)\log^b n+1}$ through $L_{l\log^b n}$. Each of these B_l is now $\log n$-uniform, so each computes a function f_l in NC_b. Here we

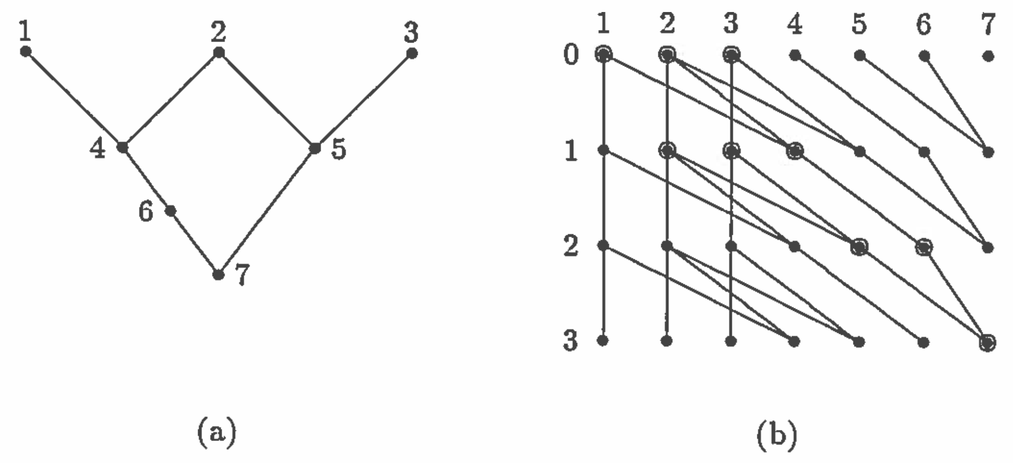

Figure 1: A circuit and its transformation

see the need for a and b to be rational, for otherwise those levels could not necessarily be determined. The language

$$F = \{ < y, i, l > \mid \text{the } i^{th} \text{ bit of } f_l(y) \text{ is } 1 \}$$

is also in NC_b, since $b \geq 1$. Using F as an oracle, there will be an NC_a circuit accepting S.
□

In the previous proof, notice that we split up the circuit and provide each level as an NC_b oracle. Unfortunately, each of these correspond to separate languages, but the circuit can query only one language. As in [4], we provide information about all the languages encoded into the language F. To perform the selection, we need at least $O(\log n)$ depth, which is why $b \geq 1$ was necessary in the construction above. This problem also arises below in theorem 12(b,c) when dealing with AC.

5 Main Theorem and Applications

The main theorem follows directly from theorems 2 and 3.

Theorem 4 *For rationals $a \geq 1$ and $b \geq 1$, $NC_{a+b-1} = NC_a^{NC_b}$.*

As corollaries to theorem 4, we get fairly simple proofs of some other structural properties.

Corollary 5 *([5]) NC_k is closed under $\leq^{NC_1}$ for $k \geq 1$.*

proof

$NC_1^{NC_k} = NC_{1+k-1} = NC_k$

□

Corollary 6 *([4]) Let $k \geq 1$ be an integer. If $NC_k = NC_{k+1}$, then $NC_k = NC$.*

proof

First note that

$$NC_{k+2} = NC_2^{NC_{k+1}} = NC_2^{NC_k} = NC_{k+1} = NC_k.$$

The fact that $NC_k = NC$ follows by induction via this process.

□

The proof of the corollary 6 provided in [4] works only when k is an integer. As a minor improvement, our characterizations provide for a more general case.

Corollary 7 *Let $k \geq 1$ be rational. If $NC_k = NC_{k+1}$, then $NC_k = NC$.*

proof

The proof is identical to that for corollary 6.

□

We can also show under weakened assumptions that NC will still collapse. Consider for example the assumption that $NC_{3.5} = NC_4$. Then

$$NC_{4.5} = NC_{1.5}^{NC_4} = NC_{1.5}^{NC_{3.5}} = NC_4 = NC_{3.5},$$

so by corollary 7, $NC_{3.5} = NC$. This can be generalized.

Theorem 8 *Let $k \geq 1$ and $\epsilon > 0$ be rationals. If $NC_k = NC_{k+\epsilon}$, then $NC_k = NC$.*

proof

We will start to decompose NC_{k+1}:

$$\begin{aligned} NC_{k+1} &= NC_{2-\epsilon}^{NC_{k+\epsilon}} = NC_{2-\epsilon}^{NC_k} = NC_{k+1-\epsilon} \\ &= NC_{2-2\epsilon}^{NC_{k+\epsilon}} = NC_{2-2\epsilon}^{NC_k} = NC_{k+1-2\epsilon} \\ &= \cdots = NC_{k+1-c\epsilon} \end{aligned}$$

after c iterations of this process. We must choose an integer c so that $2 - c\epsilon \geq 1$ (recall that theorem 4 needs $a \geq 1$) and $k + 1 - c\epsilon \leq k + \epsilon$. Choosing c in the range $\frac{1}{\epsilon} - 1 \leq c \leq \frac{1}{\epsilon}$ will work. Therefore

$$NC_{k+1} = NC_{k+1-c\epsilon} \subseteq NC_{k+\epsilon} = NC_k.$$

Corollary 7 now yields the fact that $NC_k = NC$.

□

The powers need not always be rational. The theorems above would hold for reals satisfying certain space-constructibility constraints (such as requiring a sufficient number of bits, dependent on the length of the input, to be computable in $O(\log n)$ space). The previous theorem also holds for arbitrary reals.

Corollary 9 *Let $\alpha \geq 1$ and $\delta > 0$ be real numbers. If $NC_\alpha = NC_{\alpha+\delta}$, then $NC_\alpha = NC$.*

proof

If $\alpha = 1$, then it is obviously rational. Pick rational $\epsilon \in (0, \delta]$ and apply theorem 8. If $\alpha > 1$, then choose rationals k and ϵ satisfying $k \geq \alpha$, $\epsilon > 0$, and $k + \epsilon \leq \alpha + \delta$. This can be achieved by picking $k \in [\alpha, \alpha + \frac{\delta}{2}]$ and $\epsilon \in (0, \frac{\delta}{2}]$. Note that $k + \epsilon \leq \alpha + \frac{\delta}{2} + \frac{\delta}{2} = \alpha + \delta$.

Then

$$NC_{k+\epsilon} \subseteq NC_{\alpha+\delta} = NC_\alpha \subseteq NC_k.$$

Because $NC_k = NC_{k+\epsilon}$ and $k \geq 1$, we can apply theorem 8 and see that

$$NC = NC_k = NC_{k+\epsilon} \subseteq NC_{\alpha+\delta} = NC_\alpha.$$

□

Theorem 8 can, under a weaker assumption, show a collapse of NC below NC_1.

Corollary 10 *Let $k \geq 0$ and $\epsilon > 0$ be rationals, $k + \epsilon > 1$. If $NC_k = NC_{k+\epsilon}$, then $NC_k = NC$.*

proof

If $k \geq 1$, then theorem 8 applies. If $k < 1$, then $NC_{k+\epsilon} = NC_k \subseteq NC_1$. In other words, $NC_1 = NC_{1+\xi}$, where $\xi = k + \epsilon - 1 > 0$. Theorem 8 shows that $NC = NC_1 \subseteq NC_{k+\epsilon} = NC_k$.

□

6 Unbounded Fan-In Circuits

In this section we apply the previous development to the class AC. First, we will point out that the standard containment of AC_k in NC_{k+1} holds for any oracle.

Lemma 11 *For any oracle A and $k \geq 0$, $NC_k^A \subseteq AC_k^A \subseteq NC_{k+1}^A$.*

proof

The first containment is trivial. To see that $AC_k^A \subseteq NC_{k+1}^A$, we can expand the *and* and *or* gates into a bounded fan-in tree. This will increase the depth by a factor of $O(\log n)$. Charging $\log k$ rather than 1 for the depth of an oracle gate of size k will likewise add a factor of $O(\log n)$. In both instances, recall that the size of the AC_k circuit is polynomial in n.

□

Similar to theorems 2 and 3, we can see what happens when we give NC_b and AC_b to NC_a and AC_a as oracles.

Theorem 12 **(a)** *$AC_a^{AC_b} \subseteq AC_{a+b}$ for $a \geq 0$ and $b \geq 0$.*

(b) $AC_{a+b} \subseteq AC_a^{AC_b}$ *for rationals* $a \geq 0$ *and* $b \geq 1$.

(c) $NC_{a+b} \subseteq AC_a^{NC_b}$ *for rationals* $a \geq 0$ *and* $b \geq 1$.

(d) $NC_a^{AC_b} \subseteq AC_{a+b-1}$ *for* $a \geq 1$ *and* $b \geq 1$.

proof

(a) In a UBF circuit of depth $O(\log^a n)$, look at the series of queries to AC_b made on a path from an input to the output. There are at most $O(\log^a n)$ of them, since each has depth 1. At worst, they are polynomial in size, so we will replace each by a UBF circuit of depth $O(\log^b n)$. The resulting UBF circuit is $\log n$-uniform and has depth $O((\log n)^{a+b})$.
(b) The uniform levelling technique in theorem 3 applies just as well to UBF circuits. In the same way, we can take a circuit of depth $O((\log n)^{a+b})$ and split it into $O(\log^a n)$ levels (each $\log n$-uniform) of depth $\log^b n$. Each computes a function f_l $(1 \leq l \leq O(\log^a n))$ in AC_b. The set

$$F = \{ \langle y, i, l \rangle \,|\text{the } i^{th} \text{ bit of } f_l(y) \text{ is } 1 \}$$

is also in AC_b so long as $b \geq 1$. The queries to F have depth 1, so the set accepted by the original circuit family is in AC_a^F.
(c) The same levelling technique used in theorem 3 and in the previous paragraph applies here. The $O(\log^a n)$ queries to NC_b are of polynomial size but are made by a UBF circuit in this context. Each will have depth 1.
(d) Here we replace a series of queries in the NC_a circuit by AC_b circuits. The result will be a UBF circuit. Its depth will be $O((\log n)^{a+b-1})$ by an appeal to lemma 1 as in theorem 2.
□

Now we are able to provide other characterizations of NC_k and AC_k as in theorem 4.

Theorem 13 *Let a and b be rationals.*

(a) $AC_a^{AC_b} = AC_{a+b}$ *for* $a \geq 0$ *and* $b \geq 1$.

(b) $AC_a^{NC_b} = NC_{a+b}$ *for* $a \geq 0$ *and* $b \geq 1$.

(c) $NC_a^{AC_b} = AC_{a+b-1}$ *for* $a \geq 1$ *and* $b \geq 1$.

proof

Part (a) follows from theorem 12(a,b). Part (b) in one direction follows from theorem 12(c). To get containment in the other direction, we notice that

$$AC_a^{NC_b} \subseteq NC_{a+1}^{NC_b} \subseteq NC_{a+b}$$

by lemma 11 and theorem 2. Part (c) is obtained by use of theorem 12(d) and the fact that

$$AC_{a+b-1} \subseteq AC_{a-1}^{AC_b} \subseteq NC_a^{AC_b}$$

by theorem 12(b) and lemma 11.
$\square$

The AC hierarchy collapses like the NC hierarchy. Modifying the proof of theorem 8 suffices to show this.

Corollary 14 *Let $k \geq 0$ and $\epsilon > 0$ be rationals, $k + \epsilon \geq 1$. If $AC_k = AC_{k+\epsilon}$, then $AC_k = AC$.*

An interesting point that theorem 13 shows is that AC_a is equal to NC_{a+1} if oracles from AC_b or NC_b are used.

Corollary 15 *Let $a \geq 0$ and $b \geq 1$ be rationals.*

(a) $AC_a^{AC_b} = NC_{a+1}^{AC_b}$

(b) $AC_a^{NC_b} = NC_{a+1}^{NC_b}$

proof

By theorems 4 and 13, $AC_a^{AC_b} = AC_{a+b} = NC_{a+1}^{AC_b}$ and $AC_a^{NC_b} = NC_{a+b} = NC_{a+1}^{NC_b}$.
$\square$

We see that if oracles are chosen from AC_b or NC_b for $b \geq 1$, then AC_a offers no less power than NC_{a+1}. This seems counter-intuitive in light of the fact that AC_0 is properly contained in NC_1. We also see that $AC_k^{NC_1} = NC_{k+1}^{NC_1} = NC_{k+1}$. Similarly, $NC_k^{AC_1} = AC_k$. Note that this does not say that any NC_{a+1} reduction can be replaced with an AC_a reduction on sets from AC_b or NC_b. It does say that if A NC_{a+1}-reduces to $B \in NC_b$, then there is a $C \in NC_b$ such that A AC_a-reduces to C. An interesting question is the relationship between B and C.

At lower levels of the NC and AC hierarchies, we are especially interested in the containment $NC_1 \subseteq AC_1 \subseteq NC_2$. Theorem 13 shows that $AC_1 = NC_1^{AC_1}$ and $NC_2 = AC_1^{NC_1}$. We could view this as evidence that AC_1 is "closer to" NC_2 than NC_1 since it needs a weaker oracle. However, AC_1 has more power in accessing the oracle than does NC_1, so we must be careful making such interpretations.

In [4] there is a characterization of NC as a hierarchy similar to the polynomial hierarchy: $\Sigma_1^{NC} = NC_1$ and $\Sigma_{k+1}^{NC} = AC_1^{\Sigma_k^{NC}}$. It is shown that for all integers $k \geq 1$ that $\Sigma_k^{NC} = NC_k$. This follows from theorem 13(b) as well. Also, AC_1 can be replaced by NC_2. Similarly, each AC_k could be defined in this manner: $\Sigma_1^{AC} = AC_1$ and $\Sigma_{k+1}^{AC} = AC_1^{\Sigma_k^{AC}}$ $(= NC_2^{\Sigma_k^{AC}})$.

We have seen that if $NC_k = NC_j$ ($AC_k = AC_j$) for $k < j$, then NC (AC) collapses. A natural question is what happens if either $NC_k = AC_k$ or $AC_k = NC_{k+1}$. We cannot exhibit a hierarchy collapse, but the equalities translate upwards.

Corollary 16 *Let $k \geq 1$ be rational.*

(a) *If $NC_k = AC_k$, then, for all rational $j \geq k$, $NC_j = AC_j$.*

(b) *If $AC_k = NC_{k+1}$, then, for all rational $j \geq k$, $AC_j = NC_{j+1}$.*

proof

For part (a), assume that $NC_k = AC_k$. For any rational $\delta \geq 0$,

$$AC_{k+\delta} = AC_\delta^{AC_k} = AC_\delta^{NC_k} = NC_{k+\delta}.$$

Similarly for part (b), assume that $AC_k = NC_{k+1}$. For any rational $\delta \geq 0$,

$$NC_{k+1+\delta} = NC_{1+\delta}^{NC_{k+1}} = NC_{1+\delta}^{AC_k} = AC_{k+\delta}.$$

These follow directly from theorems 4 and 13.

$\square$

7 Separation with Oracles

In [12] there is exhibited an oracle A so that, for all k, $NC_k^A \neq NC_{k+1}^A$. Here we will apply the same method to separate NC_k from AC_k and AC_k from NC_{k+1}.

A language $L_{k+1}(A)$ was introduced in [12] having the property that $\forall A$, $L_{k+1}(A) \in NC_{k+1}^A$. A specific oracle A was then constructed so that $\forall k$, $L_{k+1}(A) \notin NC_k^A$. It turns out for any k and oracle A that $L_{k+1}(A) \in AC_k^A$. This suffices to give a relativized separation of NC_k and AC_k.

Theorem 17 *There exists a recursive oracle A so that for any rational $k \geq 0$ and $0 \leq \epsilon < 1$, $AC_k^A - NC_{k+\epsilon}^A \neq \emptyset$.*

proof

We will describe the language $L_{k+1}(A)$ and then point out why it is in AC_k^A. $L_{k+1}(A)$ is the set accepted by the following procedure.

```
input $x$, $|x| = n$
    $K \leftarrow \lceil \log^{k+1} n \rceil$
    for $i \leftarrow 1$ to $K - 1$ do
        if $x0^{n-i}1b_{i-1} \ldots b_1 \in A$ then $b_i \leftarrow 1$
                                             else $b_i \leftarrow 0$
    if $x0^{n-K}1b_{K-1} \ldots b_1 \in A$
      then accept $x$
      else reject $x$
```

A cursory examination of this algorithm would seem to indicate that $L_{k+1}(A)$ is in AC^A_{k+1}. We can do better if we are careful. A UBF circuit can determine $\log n$ bits b_i at a time, so the maximum depth need only be $O(\log^k n)$. To see that $\log n$ bits can be determined in constant depth and polynomial size, let us illustrate how to find the first $\log n$ bits b_i. Let binary strings γ have length $\log n$: $\gamma = \gamma_{\log n} \cdots \gamma_1$. Define $f_\gamma = \bigwedge_{i=1}^{\log n}(x0^{K-i}\gamma_i \cdots \gamma_1 \in A)$. Then $b_i = \bigvee_{\gamma, \gamma_i = 1} f_\gamma$. A similar process can be repeated to find the next $\log n$ bits in constant depth and polynomial size, and so on. Thus, for all k and A, $L_{k+1}(A) \in AC^A_k$.

Let $\langle \cdot, \cdot \rangle$ be a standard pairwise encoding function on integers. This can be extended to handle more integers by composition. The i^{th} circuit family is the one constructed by machine M_i, where $M_1, M_2, \ldots$ is an enumeration of $O(\log n)$ space transducers. At stage $e = \langle i, c, p, q, s, t \rangle$, letting $k = \frac{p}{q}$ and $\epsilon = \frac{s}{t}$ we ensure that if M_i constructs a circuit family of depth at most $c \log^{k+\epsilon} n$, then that family cannot accept $L_{k+1}(A)$. This is done be choosing n appropriately, ensuring on an input of length n that M_i constructs an α of depth at most $c\log^{k+\epsilon} n$, and, if so, diagonalizing across the behavior of α. Initially, A will be empty, and strings will be added to it. Once added, no string will ever be removed.

For the moment we are only concerned about queries of length $2n$, those which are relevant to membership in $L_{k+1}(A)$. Given a circuit α, we will break it up into *independent query levels.* Level 1 consists of those queries which depend on no other query (that is, no other query lies on a directed path ending at that query). Level j consists of those queries which depend on some query from level $j-1$. If α has depth $c\log^{k+\epsilon} n$, then it can have at most $\frac{c\log^{k+\epsilon} n}{\log 2n} \leq c\log^{k+\epsilon-1} n$ such levels.

Construction of A stage $e = \langle i, c, p, q, s, t \rangle$

Check that $q \neq 0$, $t \neq 0$, and $\epsilon = \frac{s}{t} < 1$. If not, skip this stage. Let $k = \frac{p}{q}$.

Choose n large enough to satisfy the following constraints:

- $2^{\log^{2-\epsilon} n/c}$ is larger than the polynomial which bounds the size of the circuit constructed by M_i on 0^n and

- $2n$ is larger than anything queried or added to A at any previous stage.

Let α be the circuit constructed by M_i on 0^n. If the depth of α exceeds $c\log^{k+\epsilon} n$, then skip this stage. We now proceed in steps, and at each step fix $\frac{\log^{k+1} n}{c\log^{k+\epsilon-1} n} = \frac{\log^{2-\epsilon} n}{c}$ bits b_i. The steps will be numbered 1 through $c\log^{k+\epsilon-1} n$. Fix $x = 0^n$ as the input to α.

step m: Let $j = \frac{(m-1)}{c}\log^{2-\epsilon} n$

[Invariant: No string of the form $xzb_j \cdots b_1$, $|z| = n - j$, has been queried by α at levels 1 through $m-1$.] There are $2^{\log^{2-\epsilon} n/c}$ strings y of length $\frac{\log^{2-\epsilon} n}{c}$. By the first constraint in the choice of n there must be some y for which no string of the form $xzyb_j \cdots b_1$, $|z| = n - |y| - j$, has been queried at this or any previous level. Pick such a y and for $l = 1$ to $\frac{\log^{2-\epsilon} n}{c}$ put $x0^{n-l-j}1y_{l-1} \cdots y_1 b_j \cdots b_1$ into A if and only if y_l=1. (*end step m*)

Now that we have dealt with all levels of α, we can add strings of the form $xzb_{K-1} \cdots b_1$, $K = \log^{k+1} n$ and $|z| = n - K + 1$, to A without affecting the behavior of α on x. The final step is to add $x0^{n-K}1b_{K-1}\cdots b_1$ to A if and only if α with oracle A rejects x. **end construction**

Adding the final string to A cannot affect the behavior of α on x due to the invariance condition.

The first constraint in the choice of n provides further assurance that α will be unable to accept $L_{k+1}(A)$. Since it is certainly true that $2^{\log^{2-\epsilon} n/d} > n$, we must have $\frac{\log^{2-\epsilon} n}{d} > \log n$ or $d < \log^{1-\epsilon} n$. This implies that $d \log^{k+\epsilon} n < \log^{k+1} n$, the latter value being the depth within which an NC circuit could simulate the AC_k circuit accepting $L_{k+1}(A)$ described above.

□

Corollary 18 *There exists an oracle A so that for any rational $k \geq 0$, $NC_k^A \subset AC_k^A$ (where "$\subset$" denotes proper containment).*

An oracle was able to witness a separation between NC_k and AC_k since an AC_k circuit is able to ask a dependent series of $O(\log^k n)$ questions each of length $O(n)$. An NC_k circuit is not always able to do this. If we want to separate AC_k and NC_{k+1}, we will have to look at some advantage NC_{k+1} has over AC_k. One advantage is that it can ask a series of $O(\log^{k+1} / \log\log n)$ questions each of length $O(\log^2 n)$. A bounded fan-in circuit benefits from asking shorter questions, while an unbounded fan-in circuit has no easy way to do this.

Theorem 19 *There exists a recursive oracle A so that, for any rational $k \geq 0$ and $0 \leq \epsilon < 1$, $NC_{k+1}^A - AC_{k+\epsilon}^A \neq \emptyset$.*

proof

Consider the language $S_{k+1}(A)$ described by the following algorithm:

```
input $x$, $|x| = n$
    $x_0 \leftarrow 0^{\log^2 n}$
    $K \leftarrow \log^2 n$, $L \leftarrow \lceil \frac{\log^{k+1} n}{\log\log n} \rceil$
    for $i \leftarrow 1$ to $L$ do begin
        for every $1 \leq j \leq K$ do in parallel
            if $x_{i-1}0^{K-j}10^{j-1} \in A$ then $b_j \leftarrow 1$
                                                  else $b_j \leftarrow 0$
            end parallel
        $x_i \leftarrow b_K b_{K-1} \cdots b_1$
        end
    if $x_L 0^K \in A$
        then accept $x$
        else reject $x$
```

On an NC circuit, the depth to determine membership in $S_{k+1}(A)$ is

$$(\frac{\log^{k+1} n}{\log\log n} + 1) \cdot \log(2\log^2 n) = O(\log^{k+1} n).$$

So for any A, $S_{k+1}(A) \in NC^A_{k+1}$. The obvious UBF circuit for $S_{k+1}(A)$ has depth $\frac{\log^{k+1} n}{\log\log n}$. We will show how to construct an oracle A such that, for all k, $S_{k+1}(A) \notin AC^A_{k+\epsilon}$.

Similar to the previous construction, at stage $e = \langle i, c, p, q, s, t\rangle$ if M_i constructs a circuit family of depth no more than $c\log^{k+\epsilon} n$ where $k = \frac{p}{q}$ and $\epsilon = \frac{s}{t}$, then we will ensure that this family will not accept $S_{k+1}(A)$. Initially, A will be empty, and strings will be added to it. Once added, no string will ever be removed.

Given a circuit α, we will break it up into independent query levels, as above. The number of independent query levels in a UBF circuit is clearly bounded above by its depth.

Construction of A stage $e = \langle i, c, p, q, s, t\rangle$

Check that $q \neq 0$, $t \neq 0$, and $\epsilon = \frac{s}{t} < 1$. If not, skip this stage. Let $k = \frac{p}{q}$.

Choose n large enough to satisfy the following constraints:

- $\frac{\log^{k+1} n}{\log\log n} > c\log^{k+\epsilon} n$
- $2^{\log^2 n}$ exceeds $c\log^{k+\epsilon} n$ plus the size (a polynomial) of the circuit constructed by M_i on 0^n.
- $2\log^2 n$ is larger than anything queried or added to A at stage $m-1$.

Let α be the UBF circuit constructed by M_i on 0^n. If the depth of α is larger than $c\log^{k+\epsilon} n$, then skip the rest of this stage. Fix 0^n as the input to α. This stage proceeds in steps 1 through $\frac{\log^{k+1} n}{\log\log n}$. Initially, let $x_0 = 0^{\log^2 n}$, $K = \log^2 n$, and $L = \frac{\log^{k+1} n}{\log\log n}$

step i: Find an x_i of length $\log^2 n$ and $\forall j < i$, $x_i \neq x_j$ so that for no z of length $\log^2 n$ is $x_i z$ queried at levels 1 through i of α. This must exist since $2^{\log^2 n}$ is larger than the size of the circuit plus the number of its query levels. Where $x_i = b_K \cdots b_1$, add $x_{i-1}0^{K-j}10^{j-1}$ to A for each j satisfying $b_j = 1$. (*end step i*)

Finally, if α rejects 0^n, then add $x_L 0^K$ to A. If α accepts, do not add it to A. **end construction**

Note that L is larger than the number of independent query levels of α, so by construction α cannot have queried $x_L 0^K$. For the A described by the construction, it is the case for every rational k and $\epsilon < 1$ that any $AC^A_{k+\epsilon}$ circuit family cannot accept $S_{k+1}(A)$.
$\square$

Corollary 20 *There exists an oracle A so that, for any rational $k \geq 0$, $AC^A_k \subset NC^A_{k+1}$ (where "$\subset$" denotes proper containment).*

In fact, we have shown that $NC_{k+1}^A - AC_{k+\epsilon}^A$ contains a tally set. Another fact worth noting is that the proofs of the two previous theorems could be interleaved to construct an A where for all k, $NC_k^A \subset AC_k^A \subset NC_{k+1}^A$. As an even stronger result, we can get the following.

Corollary 21 *There exists an oracle A such that for all rationals $k \geq 0$ and $0 < \epsilon < 1$, AC_k^A and $NC_{k+\epsilon}^A$ are incomparable.*

In [2,13] there is introduced a notion of relativized space which is a reasonable measure to compare with relativized depth. We refer the reader to the original papers for details, but essentially the oracle Turing machine can put partially constructed queries into some storage mechanism, say a stack. In this way we define, for an oracle A, the classes sL^A, stack log-space relative to A, and sNL^A, the nondeterministic version (an important consideration here is that the nondeterministic machine must act deterministically while the stack is not empty). In [13] it is shown, for any A, that $NC_1^A \subseteq sL^A$ and $sNL^A \subseteq NC_3^A$. The proof of the latter containment can easily be modified to show that $sNL^A \subseteq AC_2^A$. Compare these to the unrelativized $NC_1 \subseteq L \subseteq NL \subseteq AC_1 \subseteq NC_2$.

Corollary 21 now shows that there is an oracle A so that, for any $\epsilon < 1$, $sL^A - AC_\epsilon^A$ is not empty, because NC_1^A and AC_ϵ^A are incomparable. This indicates that it may be difficult if not unlikely to improve upon the containment $NL \subseteq AC_1$. By improvement, we mean in terms of depth, as it is known that NL is contained in the semi-unbounded class SAC_1 [11]. Similarly, for any $\epsilon < 1$, $NC_{2+\epsilon}^A - sNL^A$ is not empty.

8 Conclusion

The NC and AC hierarchies provide an interesting structural contrast to other hierarchies. In many respects they behave like the polynomial hierarchy. This is especially true when considering that for these hierarchies a collapse at one level spreads upward, and this collapse can be shown by a decomposition of the higher levels. Unlike the polynomial hierarchy, the NC and AC hierarchies are dense. In this, they act like the classical space/time complexity classes. This is reasonable: NC and AC are defined by allowing progressively more and more parallel time. For NC and AC however, no separation results are known (aside from $AC_0 \neq NC_1$ [6]). A statement about the NC hierarchy which combines features of both the other hierarchies is the following:

> for any two rationals $r < t$ there exists an s such that $NC_r \subseteq NC_s \subseteq NC_t$ and if NC_s is equal to either NC_r or NC_t, then NC collapses at least to NC_s.

An interesting open question raised by Corollary 15 is the relationship between AC_a and NC_{a+1} reducibilities. For example, are they the same on L or NL: is $AC_a^L = NC_{a+1}^L$? Under what circumstances can we say that $A \leq^{NC_{a+1}} B$ implies that $A \leq^{AC_a} B$? Answering these questions should help us pinpoint the relationship of AC_a to NC_{a+1}.

In section 7 we saw oracles A so that, for any rational k, $NC_k^A \subset AC_k^A$ and $AC_k^A \subset NC_{k+1}^A$. We would like to see oracles B and C where, for any k, $NC_k^B \subset AC_k^B = NC_{k+1}^B$ and $NC_k^C = AC_k^C \subset NC_{k+1}^C$. This would raise an intriguing possibility.

We conclude this paper by pointing out that although the method presented here may seem a natural way to provide NC and AC with an oracle, the corresponding problem for space bounded classes has been much more difficult ([2,8,13]). This has been especially true when comparing relativized NC to relativized space.

Acknowledgements

Several people deserve my thanks for helping improve this paper. In particular, I have benefited from discussions with Ron Book, Gene Luks, Larry Ruzzo, and Osamu Watanabe.

References

[1] A. Borodin, S. A. Cook, P. W. Dymond, W. L. Ruzzo, and M. Tompa, "Two Applications of Complementation via Inductive Counting," *Proceedings of the 3rd Structure in Complexity Theory Conference*, 1988, pp 116-125

[2] J. Buss, "Relativized Alternation," *Proceedings of the 1st Structure in Complexity Theory Conference*, 1986, pp 66-76

[3] A. Chandra, L. Stockmeyer, and U. Vishkin, "Constant Depth Reducibility," *SIAM Journal on Computing*, vol. 13, no. 2, 1984, pp 423-439

[4] Jian-er Chen, "Logarithmic Depth Reducibility and the NC Hierarchy," *manuscript*, 1987

[5] S. Cook, "A Taxonomy of Problems with Fast Parallel Algorithms," *Information and Control*, vol. 64, no. 1, 1985, pp 2-22

[6] M. Furst, J. Saxe, and M. Sipser, "Parity, Circuits, and the Polynomial-Time Hierarchy," *Mathematical Systems Theory*, vol. 27, no. 1, 1984, pp 13-27

[7] N. Pippenger, "On Simultaneous Resource Bounds (Preliminary Version)," *Proceedings of the 20th FOCS* (1979), pp 307-311

[8] W. Ruzzo, J. Simon, and M. Tompa, "Space-bounded Hierarchies and Probabilistic Computations," *Journal of Computer and System Sciences*, vol. 28, 1984, pp 216-230

[9] W. Ruzzo, "On Uniform Circuit Complexity," *Journal of Computer and System Sciences*, vol. 22, no. 3, 1981, pp 365-383

[10] L. Stockmeyer and A. Meyer, "Word Problems Requiring Exponential Time," *Proceedings of the 5th STOC* (1973), pp 1-9

[11] H. Venkateswaran, "Properties that Characterize LOGCFL," *Proceedings of the 19th STOC*, 1987, pp 141-150

[12] C. Wilson, "Relativized NC," *Mathematical Systems Theory*, vol. 20, no. 1, 1987, pp 13-29

[13] C. Wilson, "A Measure of Relativized Space Which is Faithful with Respect to Depth," *Journal of Computer and System Sciences*, vol. 36, no. 3, 1988, pp 303-312