Reductions in Circuit Complexity: An Isomorphism Theorem and a Gap Theorem*

Manindra Agrawal†
Department of Computer Science
Indian Institute of Technology
Kanpur 208016
India
manindra@iitk.ernet.in

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

Steven Rudich
Computer Science Department
Carnegie Mellon University
Pittsburgh, PA 15213
USA
rudich@cs.cmu.edu

Abstract

We show that all sets that are complete for NP under non-uniform AC^0 reductions are isomorphic under non-uniform AC^0-computable isomorphisms. Furthermore, these sets remain NP-complete even under non-uniform NC^0 reductions.

More generally, we show two theorems that hold for any complexity class $\mathcal{C}$ closed under (uniform) NC^1-computable many-one reductions. **Gap:** *The sets that are complete for $\mathcal{C}$ under AC^0 and NC^0 reducibility coincide.* **Isomorphism:** *The sets complete for $\mathcal{C}$ under AC^0 reductions are all isomorphic under isomorphisms computable and invertible by AC^0 circuits of depth three.*

Our Gap Theorem does not hold for strongly uniform reductions: we show that there are Dlogtime-uniform AC^0-complete sets for NC^1 that are not Dlogtime-uniform NC^0-complete.

1 Introduction

The notion of *complete sets* in complexity classes provides one of the most useful tools currently available for classifying the complexity of computational problems. Since the mid-1970's, one of the most durable conjectures about the nature of complete sets is the

*The material in this paper originally appeared in the papers [AA96] and [AAIPR97].

†This research was done while visiting the University of Ulm under an Alexander von Humboldt Fellowship.

‡Supported in part by NSF grant CCR-9509603.

Berman-Hartmanis conjecture [BH77], which states that all sets complete for NP (under polynomial-time many-one reductions) are p-isomorphic; essentially this conjecture states that the complete sets are all merely different encodings of the same set. (Two sets A and B are considered p-isomorphic if there is a 1-1 function from A onto B that is polynomial-time computable and polynomial-time invertible.) Although the isomorphism conjecture was originally stated for the NP-complete sets, subsequent work has considered whether the complete sets for other complexity classes $\mathcal{C}$ (and under other notions of reducibility) collapse to an isomorphism type. In this paper, we prove such an analogue of the Berman-Hartmanis conjecture in a very natural setting: all sets complete for NP under AC^0 reductions are isomorphic under AC^0-computable isomorphisms. (Most of the results in this paper concern non-uniform AC^0 and NC^0 reductions. In all instances, AC^0 and NC^0 refer to non-uniform circuits unless otherwise indicated.) One major ingredient of our proof of this result is a 'gap' theorem: we show that all sets complete for NP under AC^0 reductions are also complete under NC^0 reductions.

In full generality, the two main theorems of this paper can be stated as follows.

For any class $\mathcal{C}$ closed under uniform NC^1-computable many-one reductions:

Isomorphism Theorem: The sets complete for $\mathcal{C}$ under AC^0 reductions are all isomorphic under AC^0-computable isomorphisms.

Gap Theorem: The sets that are complete for $\mathcal{C}$ under AC^0 and NC^0 reducibility coincide.

The remainder of the introduction will present a more detailed description of these theorems and previous related work.

1.1 The Isomorphism Theorem

The Berman-Hartmanis conjecture has inspired a great deal of work in complexity theory, and we cannot review all of the previous work here. For an excellent survey, see [KMR90]. We do want to call attention to two general trends this work has taken, regarding (1) one-way functions, and (2) more restrictive reducibilities.

One-way functions (in the worst case sense) are functions that can be computed in polynomial-time, but whose inverse functions are not polynomial-time computable. Beginning with [JY85] (see also [KMR95, Se92, KLD86], among others) many authors have noticed that if worst case one-way functions exist, then the Berman-Hartmanis conjecture might not be true. In particular, if f is one-way, nobody has presented a general technique for constructing a p-isomorphism between SAT and f(SAT), even though f(SAT) is clearly NP-complete. (Rogers [Ro95] does show how to construct such isomorphisms relative to an oracle. However, the focus of our work is on non-relativized classes. Note also that it has been shown in [KMR88, JPY94] that there are (non-complete) degrees where isomorphisms can be constructed; however the focus of our work is on complete degrees.)

An even stronger notion than one-way functions is that of *average case* one-way functions: these are polynomial-time computable functions whose inverse can be efficiently computed *only on a negligible fraction* of the range. Advances in the theoretical foundations of cryptography have shown that *average case* one-way functions can be used to construct "pseudo-random" functions that are computable in polynomial-time [HILL90]. (For an excellent distillation of the most important results in this area we recommend [Lu96].) Intuitively, if f is a polynomial-time, random-like function, $f(\mathrm{SAT})$ is NP-complete, but has no apparent structure to facilitate the construction of an isomorphism to SAT. Kurtz, Mahaney, and Royer [KMR95] are able to make this intuition technically precise in the random oracle setting. They show that when f is a truly random function, $f(\mathrm{SAT})$ is not isomorphic to SAT even when f is given as an oracle. This suggests that a pseudo-random f might similarly guarantee that no isomorphism to $f(\mathrm{SAT})$ is possible. As we argue below, the results in this paper greatly undermine our confidence in this approach to resolving the Berman-Hartmanis conjecture.

Our Isomorphism Theorem *negates* the above intuition in two important special cases. Firstly, it is easy to observe that *there are worst-case one-way functions in uniform* NC^0 if there are any one-way functions at all. Thus, if the worst-case one-way functions are sufficiently easy to compute, the intuition that worst-case one-way functions cause the isomorphism conjecture to fail is incorrect.

Secondly, we prove that all sets complete for NP under AC^0 reductions are complete under reductions that are computable via *depth two* AC^0 circuits, and these sets are all isomorphic to SAT under isomorphisms computable and invertible by AC^0 circuits of *depth three*. But it is known that there are functions computable in AC^0 that are average-case hard to compute for AC^0 circuits of depth three, and that this allows one to produce output that appears pseudorandom to AC^0 circuits of depth three [Ni92]. Using these tools, one can construct one-one functions f, many of whose output bits look random to depth three circuits. Although one might believe that for such a function f, $f(\mathrm{SAT})$ should appear random to AC^0 circuits of depth three, there is nonetheless a reduction from SAT to $f(\mathrm{SAT})$ computable in depth two, and an isomorphism computable and invertible in depth three.

Nevertheless, we refrain from concluding that our results indicate that the isomorphism conjecture is true. What is true in the AC^0 settings need not hold for the much more general polynomial-time settings.

There has been previous work on versions of the isomorphism conjecture for more restrictive reducibilities (see e.g., [Ag94, ABI93]). In fact, in [Ag94] a class of reductions, 1-NL, was presented such that the 1-NL-complete sets in natural (unrelativized) complexity classes are all 1-NL-isomorphic. However, all such reductions can be inverted in polynomial time. (For instance, the 1-L and 1-NL reductions in [Ag94, Ag95] and earlier work, and the first-order projections considered in [ABI93] have this property.) A possible exception is the so-called "1-omL reducibility" considered in [Ag94], which shares the non-invertibility property of NC^0 and AC^0 reductions considered here. However 1-omL reducibility is a rather contrived reducibility invented solely for the purpose of proving the "collapse" result in [Ag94], and the proof of that result relies heavily on the invertibility of the related 1-L and 1-NL reductions. That is *not* the case with the results presented in this paper. Additionally, we show that the sets that are complete under the reducibilities considered in [Ag94] are in fact complete under NLOG-uniform projections, and hence are also complete under NC^0 reductions. Thus the complete sets considered in [Ag94] are a subclass of the sets for which we present isomorphisms.

One of the major goals of the work presented here is to correct a shortcoming of the results presented in [ABI93]. In [ABI93], it is shown that, for essentially any natural complexity class $\mathcal{C}$, the sets complete for $\mathcal{C}$ under first-order projections are all isomorphic under first-order isomorphisms. The shortcoming of [ABI93] to which we refer is this: the complete degree under first-order projections is *properly contained in* the isomorphism type of the complete sets. In order to improve the result in [ABI93] to obtain a true analog of the Berman-Hartmanis conjecture [BH77] it would be necessary to show that the complete degree under first-order reductions coincides exactly with the first-order isomorphism type of the complete sets. Since first-order reductions are precisely the functions computable by uniform families of AC^0 circuits [BIS90], the result we present here can be seen as correcting this defect in [ABI93], except for the question of *uniformity*.

Note that, since first-order projections are a very restricted sort of NC^0 reduction, our result showing that the sets complete under AC^0 reductions are all AC^0-isomorphic would be a strict improvement of [ABI93] if not for the question of uniformity. ([ABI93] works in the Dlogtime-uniform setting; our results are known to hold only in the less-restrictive P-uniform setting (in some cases) and in the non-uniform setting.) We believe that the result for non-uniform reducibilities is interesting in its own right, and that the technical aspects of the argument lend additional interest to this work.

1.2 The Gap Theorem

A curious, often observed fact is that all sets known to be NP-complete under polynomial-time, many-one reductions remain NP-complete under many-one, AC^0 reductions (or, even weaker reductions). This is interesting because AC^0 is known to compute a much smaller class of functions than polynomial-time. Until recently ([AAIPR97]), it was not even known if each polynomial-time reduction to an NP-complete set could always be replaced by an AC^0 reduction. In this paper, we prove that such a gap in the power of reductions does exist between AC^0 and NC^0: any set that is NP-complete under non-uniform AC^0 reductions remains NP-complete under non-uniform NC^0 reductions. As is the case for polynomial-time versus AC^0, the difference in computational power between AC^0 and NC^0 is enormous. For example, each output of an NC^0 computable function can depend on only *finitely* many inputs. Thus, NC^0 can't even compute an AND of all its inputs (in contrast, the unbounded fan-in AND is an AC^0 function). Nonetheless, our theorem shows that AC^0 and NC^0 are equivalent from the point of view of performing NP-completeness reductions. It follows that all known NP-complete problems are complete under NC^0 reductions. The fact that in an NC^0 reduction each output bit depends on only finitely many of the input bits means that NC^0 reductions are local and simple by nature. Intuitively, NC^0 reductions correspond to a radically simple form of "gadget reduction."

It is instructive to consider the standard reduction from 3SAT to CLIQUE. For each clause i, for each of its literals L, we create a node (i, L). Place an edge between nodes (i, L) and (j, M) if and only if $i \neq j$ and L is not the negation of M. The resulting graph contains a clique of one third its size if and only if the original 3SAT formula is satisfiable.

We fix a particular choice of encoding a 3SAT formula with n clauses. Each literal will be encoded as a $1 + \lceil \log 3n \rceil$ bit string. The first bit indicates whether the literal is positive or negative; the remaining bits index the name of the variable. (Since there are n clauses there will be at most $3n$ different variable names.) The encoding for a given 3SAT instance will be the concatenation of the encodings of each of its literals.

This is a very simple reduction to compute; it is in AC^0. The reducing circuit takes a

3SAT formula with n clauses and produces $\binom{3n}{2}$ outputs, one for each edge of the $3n$ node graph produced by the reduction. Each output bit that is associated with two identical clauses is set to 0. Each other output bit depends on two of the input literals; it should output 1 if and only if the literals are not negations of each other. This is clearly in AC^0. *However, this is not computable in NC^0.* Why not? To check if two literals are negations of one another is not possible without considering all of the relevant $2(1+\lceil\log 3n\rceil)$ bits of input. But an NC^0 reduction can only consider a *constant* number of the relevant input bits.

It is an exercise to find an NC^0 reduction from 3SAT to CLIQUE. Using our results, any such exercise is a corollary of our Gap Theorem.

As stated above, we prove that for any class $\mathcal{C}$ closed under NC^1-computable many-one reductions the sets that are complete for $\mathcal{C}$ under AC^0 and NC^0 reducibility coincide. This is a *gap* theorem in the sense that there is a big difference (gap) in the computational power of NC^0 and AC^0 functions, but no difference in their power to perform completeness reductions. This is analogous to the Borodin-Trakhtenbrot Gap Theorem [Bo72, Tr64]. Some other similar gap theorems are presented in [Ag94, Ag96], and in [Ag95] it is shown that sometimes, the hypothesis that a gap exists for two reducibilities can have interesting consequences.

We do not know if our Isomorphism Theorem holds for Dlogtime-uniform AC^0 isomorphisms (also known as first-order isomorphisms). In fact, we show that our Gap Theorem fails in the Dlogtime-uniform setting, i.e., there are Dlogtime-uniform AC^0-complete sets for NC^1 (or any other natural class) that are *not* Dlogtime-uniform NC^0-complete. This implies that with our approach one cannot hope to prove the Isomorphism Theorem for Dlogtime-uniform AC^0 isomorphisms.

1.3 Section Organization

Section 2 presents definitions for the classes of reductions considered in this paper.

Section 3 presents our results about sets complete under NC^0 reductions, the gap theorem, the isomorphism theorem, and that it cannot be improved to Dlogtime-uniform reductions using our approach.

Section 4 presents some concluding remarks.

2 Basic Definitions and Preliminaries

We assume familiarity with the basic notions of many-one reducibility as presented, for example, in [BDG88]. In this paper, **only many-one reductions will be considered**.

A *circuit family* is a set $\{C_n : n \in \mathbf{N}\}$ where each C_n is an acyclic circuit with n Boolean inputs $x_1, \ldots, x_n$ (as well as the constants 0 and 1 allowed as inputs) and some number of output gates $y_1, \ldots, y_r$. $\{C_n\}$ has *size* $s(n)$ if each circuit C_n has at most $s(n)$ gates; it has *depth* $d(n)$ if the length of the longest path from input to output in C_n is at most $d(n)$. A family $\{C_n\}$ is *uniform* if the function $n \mapsto C_n$ is easy to compute in some sense. In this paper, we will consider only Dlogtime-uniformity [BIS90] and P-uniformity [Al89] (in addition to non-uniform circuit families).

A function f is said to be in AC^0 if there is a circuit family $\{C_n\}$ of size $n^{O(1)}$ and depth $O(1)$ consisting of unbounded fan-in AND and OR and NOT gates such that for each input x of length n, the output of C_n on input x is $f(x)$. Note that, according to this strict definition, a function f in AC^0 must satisfy the restriction that $|x| = |y| \Longrightarrow |f(x)| = |f(y)|$. However, the imposition of this restriction is an unintentional artifact of the circuit-based definition given above, and it has the effect of disallowing any interesting results about the class of sets isomorphic to SAT (or other complete sets), since there could be no AC^0-isomorphism between a set containing only even length strings and a set containing only odd length strings – and it is precisely this sort of indifference to encoding details that motivates much of the study of isomorphisms of complete sets. Thus we allow AC^0-computable functions to consist of functions computed by circuits of this sort, where some simple convention is used to encode inputs of different lengths (for example, "00" denotes zero, "01" denotes one, and "11" denotes end-of-string; other reasonable conventions yield exactly the same class of functions). For technical reasons, we will adopt the following specific convention: each C_n will have $n^k + k\log(n)$ output bits (for some k). The last $k \log n$ output bits will be viewed as a binary number r, and the output produced by the circuit will be the binary string contained in the first r output bits. It is easy to verify that this convention is AC^0-equivalent to the other convention mentioned above, and for us it has the advantage that only $O(\log n)$ output bits are used to encode the length. It is worth noting that, with this definition, the class of Dlogtime-uniform AC^0-computable functions admits many alternative characterizations, including expressibility in first-order with $\{+, \times, \leq\}$ [Li94, BIS90][1], the logspace-rudimentary reductions of Jones [Jo75, AG91], logarithmic-time alternating Turing machines with $O(1)$ alternations [BIS90] and others. This lends additional weight to our choice of this definition.

NC^1 and NC^0 are the classes of functions computed in this way by circuit families of size $n^{O(1)}$ and depth $O(\log n)$ (or $O(1)$, respectively), consisting of fan-in two AND and OR and NOT gates. Note that for any NC^0 circuit family, there is some constant c such that each output bit depends on at most c different input bits. An NC^0 function is a *projection* if its circuit family contains no AND or OR gates. The class of functions in NC^0 was considered previously in [Hå87]. The class of projections is clearly a subclass of NC^0 and has been studied by many authors; consult the references in [ABI93].

For technical reasons and for simplicity of exposition, we do *not* allow an NC^0 circuit C_n to produce outputs of different lengths for different inputs of length n, although we *do* allow AC^0 and NC^1 circuits to do this by following the convention mentioned above. That is, if f is computed by NC^0 circuit family $\{C_n\}$ where each C_n has $s(n)$ output bits, then for all inputs x of length n, $|f(x)| = s(n)$. Our chief justification for imposing this restriction is that Theorem 10 shows that any set hard for NP (or other complexity classes) under AC^0 reductions (using the less-restrictive convention allowing outputs of different lengths) is in fact hard under NC^0 reductions (using the more-restrictive convention). Thus we are able to obtain our corollaries about sets complete under AC^0 reductions without dealing with the technical complications caused by allowing NC^0 reductions to output strings of different lengths. Also note that, even with this restriction, the NC^0 reductions we consider

[1] Lindell [Li94] shows only that this coincides with first-order expressibility in first order with $\{+, \times, \leq, \exp\}$, where "exp" denotes exponentiation. However, personal communication from K. Regan and S. Lindell shows that exponentiation can be eliminated.

are still more general than the first-order projections considered in [ABI93].

For a complexity class $\mathcal{C}$, a $\mathcal{C}$-isomorphism is a bijection f such that both f and f^{-1} are in $\mathcal{C}$. Since only many-one reductions are considered in this paper, a "$\mathcal{C}$-reduction" is simply a function in $\mathcal{C}$.

(A *language* is in a complexity class $\mathcal{C}$ if its characteristic function is in $\mathcal{C}$. This convention allows us to avoid introducing additional notation such as FAC^0, FNC^1, etc. to distinguish between classes of languages and classes of functions.)

The theorems we prove in this paper hold for most complexity classes that are of interest to theoreticians; we require only closure under certain easily-computable reductions. To make this precise, we say that a class of languages $\mathcal{C}$ is a *proper complexity class* if $\mathcal{C}$ is closed under Dlogtime-uniform NC^1 reductions. (That is, if A is in $\mathcal{C}$, and B is reducible to A via a many-one reduction computable in NC^1, then B is in $\mathcal{C}$.) Note that most complexity classes, such as NP, P, PSPACE, BPP, etc. are proper complexity classes.

In fact, inspection of our proofs shows that our results hold even for any class $\mathcal{C}$ that is closed under reductions computed by uniform threshold circuits of depth five. (The number five can probably be reduced.) We do not know how to weaken the assumption to closure under reductions computed in ACC^0; it is easy to see that our results do *not* hold for some classes closed under AC^0 reductions. For instance, the sets $\{1\}$ and $\{1, 11\}$ are both hard for AC^0 under AC^0 reductions, but they are not isomorphic, and they are not hard under NC^0 reductions.

A function is *length-nondecreasing* (*length-increasing*, *length-squaring*) if, for all x, $|x| \leq |f(x)|$ ($|x| < |f(x)|$, $|x|^2 \leq |f(x)|$); it is $\mathcal{C}$-invertible if there is a function $g \in \mathcal{C}$ such that for all $x, f(g(f(x))) = f(x)$.

The following proposition is well-known:

Proposition 1 *P=NP iff every length-increasing Dlogtime-uniform NC^0 function is P-invertible.*

Proof. The forward direction is obvious. For the converse, assume that 3SAT is not in P. Consider the following encoding of a 3CNF formula with variables $v_1, \ldots, v_n$. Note that there are only $8n^3$ clauses on these variables that can possibly appear in any 3CNF formula. A formula can thus be encoded as a sequence of $8n^3$ bits, with each bit denoting the presence or absence of the corresponding clause. Now consider the function f defined by $f(\phi, \vec{v}) = (\phi, x)$ where x is the string of length $8n^3$ such that the ith bit of x is 1 iff (the ith bit of ϕ is 0 **or** the ith bit of ϕ is 1 and the corresponding clause evaluates to 1 when the variables are set according to the assignment $\vec{v}$). It is clear that f is length-increasing, and it is not hard to see that f is computed by a Dlogtime-uniform family of NC^0 circuits. Note that ϕ is in 3SAT iff $(\phi, 1^{|\phi|})$ is in the range of f, which can be detected in polynomial time if f is P-invertible. ∎

3 Main Results

3.1 Superprojections

Definition 1 *An NC^0 reduction $\{C_n\}$ is a* superprojection *if the circuit that results by deleting zero or more of the output bits in each C_n is a projection wherein each input bit (or* *its negation) is mapped to some output. (Stated another way, it is a superprojection if, for each input bit x_i, there is an output bit whose value is completely determined by x_i. That is, this output bit is either x_i or $\neg x_i$.)*

Note that every superprojection has an inverse that is computable in AC^0: On input y, we want to determine if there is an x such that $f(x) = y$. The AC^0 circuit will have a subcircuit for each $n \leq |y|$ (since a superprojection is by definition one-one and length-nondecreasing) checking to see if there is such an x of length n. This subcircuit will find the n output bits that completely determine what x must be (if such an x exists), and then will check to see if $f(x) = y$.

Theorem 2 *For every proper complexity class $\mathcal{C}$, every set hard for $\mathcal{C}$ under P-uniform NC^0 reductions is hard under P-uniform one-one, length-squaring superprojections.*

Proof. Let A be hard for $\mathcal{C}$ under NC^0 reductions. We shall show A to be hard under one-one length-squaring superprojections in two stages.

Stage 1: In the first stage, we show that A is also hard under length-nondecreasing superprojections.

Take any set B in $\mathcal{C}$. We define a new set C, TC^0-reducible to B, that is accepted by the following procedure:

> On input y, let $y = 1^k 0 z$. If k does not divide $|z|$, then reject. Otherwise, break z into blocks of k consecutive bits each. Let these be $u_1 u_2 u_3 \ldots u_p$. Accept if there is an i, $1 \leq i \leq p$, such that $u_i = 1^k$. Otherwise, reject if there is an i, $1 \leq i \leq p$, such that $u_i = 0^k$. Otherwise, for each i, $1 \leq i \leq p$, label u_i as *null* if the number of ones in it is less than $k/2$; as *zero* if the number of ones in it are between $k/2$ and $3k/4$; and as *one* otherwise. Let $v_i = \epsilon$ if u_i is null, 0 if u_i is zero, and 1 otherwise. Let $x = v_1 v_2 \cdots v_p$, and accept iff $x \in B$.

It is straightforward to see that C reduces to B via a Dlogtime-uniform NC^1 reduction. Therefore, $C \in \mathcal{C}$ by the closure properties of $\mathcal{C}$. Since A is NC^0-hard for $\mathcal{C}$, there exists an NC^0 reduction of C to A. Let this be given by the family of circuits $\{D_n\}$. In what follows, we will use the circuits D_n (reducing C to A) to construct a projection reducing B to C with the property that the composition of these two reductions is a superprojection from B to A.

Let the depth of circuits in the family $\{D_n\}$ be bounded by the constant d and let $c = 2^{2^{2^d}}$. The projection from B to C will map strings of size n to strings of size $4c+1+4cm$ where $m = O(n^c)$ (the exact value of m will be given later). It will map string x, $|x| = n$, to the string $1^{4c} 0 u_1 u_2 \ldots u_m$ where each u_i is of size $4c$, and where the string formed out of these u_is (as described in the procedure defining C) is x. We show below how the values of the u_is are computed. In the discussion below, we refer to the u_is as *blocks*.

Consider the circuit $D_{4c+1+4cm}$. Set the first $4c+1$ bits of the circuit to $1^{4c}0$ and consider the reduced circuit with $4cm$ unset input bits. Each output bit of this circuit depends on at most 2^d input bits. Let O be a *maximal* set of output bits satisfying the property that (1) each output bit in O depends on at least one input bit (so there are no constant output bits in O), and (2) no two output bits in O depend on the same set of input bits. We first show that $|O| \geq m/2^d$. Suppose not. Since any output bit of the circuit depends on at most 2^d input bits, the bits in O depend on at most $|O| \cdot 2^d < m$ input bits. Also, any output bit of the circuit that is not in O can depend only on these input bits since otherwise it would have been included in O. Thus, there are fewer than m input bits, out of $4cm$, on which the output of the circuit depends. This implies that there is one whole block of input bits, say u_i, that does not affect the output. Set all the blocks except u_i to zero (i.e., to a value with number of ones between $2c$ and $3c$, e.g., $1^{2c}0^{2c}$). Now, setting u_i to 1^{4c} or 0^{4c} keeps the output of the circuit identical, which is a contradiction since $D_{4c+1+4cm}$ reduces strings of C to A.

Therefore, the size of O is r, for some

$$r \geq m/2^d. \tag{1}$$

We will start with this set O of output bits, and possibly remove some bits from O as the construction proceeds.

The remaining input bits are denoted as usual with $x_1, \ldots, x_{4cm}$. We view each output bit as the outcome of a (bounded) truth-table evaluation on the input bits on which it depends. (We need to be fairly precise here about how we associate a truth table to each output bit. Consider one of the output bits in O, and consider the fan-in two circuit of depth d that computes this output bit. Order these input bits according to the index, with x_i coming before x_j if $i < j$. If one of these $\leq 2^d$ input bits actually has no effect on the value of the output, then remove that input bit and simplify the circuit computing the output bit accordingly, and let the number of input bits remaining be $d' \leq 2^d$. The "truth table" for this output bit has variables $v_1, \ldots, v_{d'}$. The value of the output bit is obtained by plugging in the appropriate input bit for each v_i.) Note that there are at most $2^{2^{2^d}} = c$ different such truth-tables. We choose some truth-table, say α, that is associated with a maximal number of output bits (i.e., at least as many as is any other truth table).

At this point, remove from O all output bits that do *not* have α as their truth table, and let s be the number of output bits that now remain in O. Clearly,

$$s \geq r/c. \tag{2}$$

Consider the ith output bit remaining in O. Let the ordered set of input bits on which it depends be $\{x_{i,1}, x_{i,2}, \ldots, x_{i,d'}\}$ where $d' \leq c$. Consider the family of sets $\mathcal{F} = \{S_i\}$ where $S_i = \{(j, x_{i,j}) : 1 \leq j \leq d'\}$. All of the sets in $\mathcal{F}$ have size at most c, and note that, by the way we have constructed our set O, we have that if $i \neq i'$, then $S_i \neq S_{i'}$. Thus by the Sunflower Lemma of [ER60] (see also [BS90, Lemma 4.1]), the collection of sets $\mathcal{F}$ contains a sunflower of size at least

$$t \geq (\lfloor s/(c!) \rfloor)^{(1/c)}. \tag{3}$$

(This "Sunflower" is a collection of t sets from $\mathcal{F}$ with the property that, for all pairwise distinct sets S_1, S_2, S_3, S_4 in the sunflower, $S_1 \cap S_2 = S_3 \cap S_4$. The set that one obtains by intersecting any two elements of the sunflower is called the "core" of the sunflower.) We now remove from O all of those output bits i such that S_i is not in this sunflower. Thus $|O| = t$ now.

Consider any two bits i and j that remain in O. S_i and S_j record the input bits on which output bits i and j depend, and note that the input bits in $S_i \cap S_j$ correspond to exactly the same variables in the truth table α that determines how i and j depend on these inputs. Now, set the bits in the core of the sunflower to 0-1 values such that the truth-table α does not become a constant (this can always be done, because the truth table α depends on *all* of its d' input bits). So now each output bit in O depends only on the input bits that are in the corresponding petal of the sunflower. We will process each petal of the sunflower in turn.

Consider the first petal (corresponding to output bit i_1). Since setting the bits in the core did not make α a constant, there is some bit z_{i_1} in this petal and some assignment of $\{0,1\}$ values to the other bits in the petal such that the output bit i_1 depends only on the value of z_{i_1}. Moreover, since the truth-table relating the output bits to petals is identical for all petals, we obtain a corresponding bit z_i in *each* petal, along with an identical assignment to the remaining bits in each petal. (There is a subtle point here. Although the sets in our sunflower $\mathcal{F}$ have pairwise intersection equal to the core of the sunflower, and thus the petals are each pairwise disjoint, this says only that a tuple $(j, x_{k,j})$ can appear in at most one petal, but it does *not* say that a given input variable can appear in at most one petal (although it does follow that a given input variable can appear in at most $d' \leq c$ different petals, each time paired with a different number j). In particular, it is certainly possible that the "identical assignments to the remaining bits in each petal" referred to above will conflict with each other. We will show below how to deal with this.) Call the bit z_i the *identified bit* for the petal i.

Recall that our goal is to map a string x of length n to a string of the form $1^{4c}0u_1u_2\ldots u_m$ where each u_i is of size $4c$, and where each u_i is either "null" or represents a single bit of x. Our overall approach is to map input bits of x to the identified bits z_i in the petals of the sunflower. When we try to do this we have to assign values to the bits in the core of the sunflower and to the other bits in the petals of the sunflower; this will cause us to make some of the blocks u_i "null", and it will cause us to remove some of the petals from O. We will succeed if we can show how to make this assignment and still end up with enough petals to encode all of the bits of x.

Process each output bit $i \in O$ in turn. Consider the unset bits in S_i. (Initially, *none* of the bits in S_i are set. When we process the first bit i in O we will set all of the bits in S_i except for z_i, including all of the bits in the core. When we process the other bits in O only the bits in the petal will be unset.) For each of these bits *other* than z_i, set this bit to the value (discussed above) so that output bit i depends only on the value of z_i. (This causes at most $c-1$ bits to be set.) Each of these bits is in some block u_j. Consider any such block u_j that contains one of these bits that has just been set but *does not* contain z_i. Set the rest of the bits in such a block u_j to zero. Note that this has the effect of making block u_j *null*, since the length of u_j is $4c$ and we are setting at least $3c+1 > 2c$ variables in this block to zero. We have now set all blocks containing variables in S_i except for the block u_{j_i} containing z_i. This block contains at most $c-1$ variables that have been set. Set the rest of the inputs in block u_{j_i} (that is, set the variables in u_{j_i} other than z_i) so that there are exactly $3c$ ones and $c-1$ zeroes in the block. (This has the effect of making the block depend on the identified bit: it is zero when the identified bit is zero and one otherwise.) Thus far in processing petal i, we have set fewer than $4c^2$ input bits (at most $4c$ for each bit other than z_i, and at most $4c-1$ for z_i). Some of the input bits we have set (including some of the bits in the petal just processed) may be elements of other petals in our sunflower. Remove from O any output j such that its petal contains a bit that has been set in this way; remove also any j such that its petal contains the bit z_i just processed. This causes O to lose fewer than $4c^3$ output bits (since each of the $< 4c^2$ bits can appear in at most c petals), and in the remaining sunflower, none of the bits in any petal has been set. Note that the end result of processing this element $i \in O$ is that we have obtained an input bit z_i such that the output bit i is a projection of z_i. Now repeat the process described above for the next bit remaining in O.

We repeat this process $|x|$ times to obtain $|x|$ such bits z_i. In order for this to be possible, it is sufficient for t to be at least $4c^3 \cdot |x|$. This gives us a bound on m: $m \leq 2^d \cdot r$ (by (1)) $\leq 2^d \cdot c \cdot s$ (by (2)) $\leq 2^d \cdot c \cdot (t^c + 1) \cdot c!$ (by (3)), and thus if we pick t to be $4c^3 \cdot |x|$, it follows that it is sufficient to choose m to be $c'' \cdot |x|^c$ for some constant c'' depending on d. (Recall that c depends on d.)

So our reduction of B to C will, on input x of length n, identify bits $z_1, \ldots, z_n$ and map x_i to bit z_i, where the other bits of circuit $D_{4c+1+4cm}$ are set according to the procedure listed above (or if there are any remaining bits left unset by this procedure, we set those bits to zero, having the effect of nullifying all remaining blocks not containing one of the z_is). This reduction of B to C is just a projection from B, since every output bit depends on at most one input bit. It maps a string of size n to one of size $4c + 1 + 4cm$ with $m = c'' \cdot n^c$.

If we now consider the reduction from B to A that results by composing the projection from B to C with the reduction $D_{4c+1+4cm}$, we note that the n bits that are determined by the z_i are merely the projections of the input x, and the other bits are either fixed or correspond to output bits that were deleted from O by the foregoing procedure, but nonetheless are still computed by the NC^0 circuit. Thus the reduction is a superprojection.

It is somewhat tedious to verify that this reduction can be made P-uniform. First observe that α can be found in logspace. Then observe that there are at most n^c sets that could possibly be the core of the desired sunflower; exhaustively trying each such possible core in turn, and then using a greedy algorithm to find a maximal collection of sets containing the core and with pairwise disjoint "petals" will eventually uncover a sunflower of the desired size. (The proof of the Sunflower Lemma given in [BS90] shows that this approach will succeed.) Finding the desired setting of the bits in the core and petals is easy. Then sequentially deleting the bits from the petals is straightforward.

Stage 2: It is clear at this point that the reduction of B to A described above is length-nondecreasing and also 1-1 at least on strings of the same length. However, it may map strings of two different lengths to the same string. To take care of this problem, we add another stage of the construction.

Once more, we take any set B in $\mathcal{C}$. Once more, we define a new set E AC^0-reducible to B. The definition of E is straightforward:

$$E = \{x10^k \mid x \in B \quad k \in \mathbf{N}\}.$$

E is clearly in $\mathcal{C}$ and therefore there exists an NC^0 reduction of E to A that is a length-increasing superprojection. Let this reduction be given by the function f. We know that for all x: $|x| \leq |f(x)| \leq p(|x|)$ for some polynomial p. Define a function r as follows: $r(0) = 1$, and $r(t+1) = p(r(t)) + 1$. And now define a reduction g of B to E as: $g(x) = x10^k$ where k is the smallest number such that: $k \geq |x|^2$ and $|x| + 1 + k = r(t)$ for some t. Function g can clearly be computed by a projection circuit, and so $f \circ g$ is a superprojection reducing B to A. It is length-increasing because g and f are both length-increasing. It is 1-1 also, which can be seen as follows: for any two strings x and y such that $|g(x)| = |g(y)|$, $f(g(x)) \neq f(g(y))$ follows from the nature of f. And when $|g(x)| > |g(y)|$ then $|f(g(y))| \leq p(|g(y)|) = p(r(t))$ (for some t) $< r(t+1) \leq |g(x)| \leq |f(g(x))|$.

Also note that, since g is length-squaring and f is length-nondecreasing, the resulting superprojection is at least length-squaring.

Checking P-uniformity of this step is trivial. ∎

The following corollary (the non-uniform case) is a trivial consequence of the foregoing.

Corollary 3 *For every proper complexity class $\mathcal{C}$, every set complete for $\mathcal{C}$ under NC^0 reductions is complete under one-one, length-squaring superprojections.*

Corollary 4 *For every proper complexity class $\mathcal{C}$, every set complete for $\mathcal{C}$ under NC^0 reductions is complete under reductions computable by depth two AC^0 circuits and invertible by depth three AC^0 circuits. (If the NC^0 reductions are P-uniform, then so are the AC^0 circuits.)*

Proof. First note that since a superprojection is an NC^0 reduction, it can be computed in depth two simply by expressing each output bit in DNF or CNF form.

Next note that because of the construction of Stage 2 of the proof of Theorem 2, we know that if A is complete for $\mathcal{C}$ under NC^0 reductions, then it is complete under superprojections f of the form $h \circ g$ where h is a superprojection when restricted to strings in the range of g, and strings in the range of g have the form $y10^k$. Furthermore, for each n there is an (easily-computed) m such that, for each string x of length n, if $f^{-1}(x)$ exists, then there exist y and k such that $|g(y)| = |y10^k| = m$, $f^{-1}(x) = y$, and $h(y10^k) = x$. (The point here is that m depends only on $n = |x|$.) Now to compute f^{-1} for inputs of length n, it suffices to consider the circuit computing h on inputs of length m. For inputs x of length n, $f^{-1}(x)$ exists if and only if $h^{-1}(x)$ exists and is of the form $y10^k$ for k in the correct range. If $h^{-1}(x)$ exists, then there are m bits of x that directly encode the bits of $y10^k$, since h is a superprojection.

Thus our circuit to compute $f^{-1}(x)$ first takes the string $y10^k$ that is available on the input level of the circuit (as determined by m bits of input x) and that is a candidate for $h^{-1}(x)$. Then (in depth two) it computes $h(y10^k)$, and checks that all of the bits of $h(y10^k)$ and x agree. This is an AND of several NC^0 predicates, and by expressing the NC^0 predicates in CNF and merging the two levels of AND gates we obtain a depth two circuit producing output $y10^k$ if $h^{-1}(x) = y10^k$. Since our goal is to produce output y (and also output $|y|$ in the length-encoding field) we obtain a depth three circuit by taking the OR over all possible values of $r = |y|$ of the predicate "$h^{-1}(x) = y10^k$ AND the last $m - r$ bits of $y10^k$ are in 10^*". ∎

Corollary 5 *For every proper complexity class $\mathcal{C}$, all sets complete for $\mathcal{C}$ under P-uniform NC^0 reductions are P-uniform AC^0-isomorphic. Furthermore, these isomorphisms are computable and invertible by P-uniform AC^0 circuits of depth three.*

Proof. The main result in [ABI93], showing that all sets complete under first-order projections are first-order isomorphic, carries over also into the P-uniform setting, and the same proof also works for superprojections. We refer the reader to [ABI93] for details, but we sketch some of the important steps here.

Let A and B be complete for $\mathcal{C}$ under NC^0 reductions. Thus there are superprojections f and g reducing A to B and reducing B to A, respectively. Our goal is to construct an isomorphism mapping A onto B. We first construct a depth four isomorphism between A and B, and then improve it to depth three. As in most other work constructing isomorphisms (see [BH77] for example), given an input x, we will need to compute the length of the "ancestor chain" of x, and output $f(x)$ if the length of the chain is even, and output $g^{-1}(x)$ if the length of the chain is odd.

Note that if the k^{th} ancestor exists, then (just as in the case $k = 1$ in the proof of Corollary 4) the bits of the k^{th} ancestor are available at the input level. Thus one can determine in depth three if the length of the ancestor chain is exactly r. (Namely, for all $k < r$ the appropriate inverse image of the k^{th} ancestor exists, and it does not exist for the r^{th} ancestor.) Now, the i^{th} bit of the output would be

$$((\bigvee_{k \text{ odd}} r = k) \bigwedge (i^{th} \text{ bit of } f(x))) \bigvee ((\bigvee_{k \text{ even}} r = k) \bigwedge (i^{th} \text{ bit of } g^{-1}(x))).$$

This gives a depth six circuit, however, note that the top two levels are of fan-in two, and therefore, can be "pushed down" and collapsed with the bottom two levels. This results in a depth four circuit.

To reduce the depth further, we observe that we do not need to explicitly check for the existence of the inverse at level two (as is done in the proof of Corollary 4). Instead, we distribute this work to the top two levels: Let $C_{k,m}$ be the NC^0 circuit that outputs a sequence of ones iff the k^{th} ancestor exists and has length m. Let C'_k be the depth two AC^0 circuit (with top level AND gates) that outputs a sequence containing at least one zero iff the k^{th} ancestor does *not* exist. (To see how to construct C'_k, note that if the k^{th} ancestor does exist, then there is a $k-1^{th}$ ancestor z of some length m that is completely determined by n and k, and the k^{th} ancestor is a string y where $h^{-1}(z) = y10^r$ where r cannot be too large. That is, in addition to the local consistency checks (each bit of which can computed in NC^0), the only other condition that must be checked is to say that the k^{th} ancestor does *not* exist if $h^{-1}(z) = y10^r$ ends in too many zeros. This can clearly be checked by a CNF circuit.) Let $D_{r,l,\vec{m}}$ be the circuit computing

$$(\bigwedge_{k \le r} \text{output}(C_{k,m_k}) \in 1^*) \bigwedge (l^{th} \text{ bit of output}(C'_{r+1}) = 0).$$

(Here, $\vec{m}$ is a vector of $O(\log n)$ bits encoding a sequence $m_1, \ldots, m_r$ of numbers such that $(m_i)^2 \le m_{i+1} \le n$. Since f and g are both length-squaring, the sequence of lengths occurring in the ancestor chain can be encoded as such a vector. $D_{r,l,\vec{m}}$ is still a depth two circuit since the AND gate at the top can be merged with the AND gates on top of the C_{k,m_k}s and C'_{r+1}s. The i^{th} bit of the output can now be defined as:

$$((\bigvee_{r \text{ odd}} \bigvee_{l} \bigvee_{\vec{m}} D_{r,l,\vec{m}}) \bigwedge (i^{th} \text{ bit of } f(x))) \bigvee ((\bigvee_{r \text{ even}} \bigvee_{l} \bigvee_{\vec{m}} D_{r,l,\vec{m}}) \bigwedge (i^{th} \text{ bit of } g^{-1}(x))).$$

As argued before, this is a depth three circuit. A similar circuit computes the inverse of the isomorphism. ∎

Corollary 6 *For every proper complexity class $\mathcal{C}$, all sets complete for $\mathcal{C}$ under NC^0 reductions are AC^0-isomorphic. Furthermore, these isomorphisms are computable and invertible by AC^0 circuits of depth three.*

The above corollary generalizes the result not only of [ABI93] to a larger class of complete sets, but also the results of [Ag94]. To see this, we observe that the complete sets under all the reducibilities considered in [Ag94] are also complete under NLOG-uniform projections. It was shown in [Ag94] that the complete sets under 1-L, 1-NL, and 1-omL reductions are also complete under *forgetful* 1-L, 1-NL, and 1-omL reductions respectively. A *forgetful* reducibility was defined there as one computed by a TM that, after scanning each bit of the input, ends up in a configuration that depends only on the size of the input—in particular, it is independent of the value of the bits scanned so far. It is easy to see that a forgetful TM computes a function that is a projection—the output of the TM during its scan of any input bit depends only on the bit and the length of the input; and thus every output bit depends on at most one input bit. And since all the three reducibilities 1-L, 1-NL, and 1-omL are subclasses of NLOG-reducibility, the corresponding forgetful TMs compute NLOG-uniform projections. Thus, these complete sets form a strict subclass of NC^0-complete sets as it is straightforward to construct a set that is NC^0-complete (for any proper class) but not complete under even non-uniform projections [ABI93].

3.2 The Gap Theorem and The Isomorphism Theorem

3.2.1 Random Restrictions of $\mathbf{AC}^0$ reductions

An important tool will be the fact that when we randomly restrict the inputs to a circuit family computing an AC^0 function we obtain a circuit family computing an NC^0 reduction. This has been a folklore theorem since [FSS84, Aj83]. Lemma 7 below is explicitly stated in [Ar95]. The proof can be gleaned by suitable modifications of any of several alternative presentations [BS90, Fo95, B95]. The proof of a slightly stronger statement can be found in the appendix of this paper.

Definition: For a natural number a, an *a-random restriction* ρ_a on m variables is a function independently assigning to each variable a value in $\{0, 1, *\}$ as follows: set it to 0 with probability $(1 - m^{(1/a)-1})/2$, set it to 1 with probability $(1 - m^{(1/a)-1})/2$, and set it to $*$ with probability $m^{(1/a)-1}$.

Lemma 7 *For any AC^0 reduction computed by a family of circuits $\{C_m\}$, there exists an $a \in \mathbf{N}$ such that, with probability $1 - o(1)$, an a-random restriction ρ_a on m variables transforms C_m into an NC^0-circuit with $\Omega((m^{1/a}))$ input variables.*

In the appendix of this paper we prove a slightly stronger version of this, which (to the best of our knowledge) has been a folklore theorem.

Lemma 8 *Let $m = r^{2a}$, and let the m variables $x_1, \ldots, x_m$ be divided into r blocks of length r^{2a-1} (where block i consists of variables $x_{ir+1}, \ldots, x_{ir+r}$). Then with probability $1 - o(1)$, an a-random restriction assigns $*$ to at least three variables in* each *block.*

Proof. Each one of the m bits will have probability $p = m^{(1/a)-1} = r^{2a(1/a)-1} = r^{2-2a}$ of getting set to $*$. Notice that the block size is $r^{2a-1} = r/p$. The probability that any particular block of size r/p gets no more than two $*$s is given by:

$$\begin{array}{ll} & (1-p)^{r/p} + \frac{r}{p}p(1-p)^{(r/p)-1} + \binom{r/p}{2}p^2(1-p)^{r/p-2} \\ \leq & e^{-r} + re^{-r} + r^2e^{-r} \\ = & \mathrm{O}(r^2e^{-r}) \end{array}$$

Since there are r blocks the probability that one of them gets fewer than 3 $*$'s is $\mathrm{O}(r^3e^{-r}) = o(1)$. It follows that all blocks get at least 3 $*$'s with probability $1 - o(1)$. ∎

All that is needed for the results of the following section is this easy consequence of the preceding two lemmas.

Corollary 9 *For any AC^0 reduction computed by a family of circuits $\{C_m\}$, there exists an $a \in \mathbf{N}$ such that, for all large m of the form r^{2a}, there is a restriction τ_m such that τ_m transforms C_m into an NC^0 circuit, and τ_m assigns $*$ to at least three variables in* each *block of length r^{2a-1}.*

3.2.2 The Gap Theorem

Theorem 10 (Gap Theorem) *Let $\mathcal{C}$ be any proper complexity class. The sets hard for $\mathcal{C}$ under non-uniform AC^0 reductions are hard for $\mathcal{C}$ under non-uniform NC^0 reductions.*

Proof.

Let $\mathcal{C}$ be any proper complexity class, i.e., $\mathcal{C}$ is closed under Dlogtime-uniform NC^1 reductions. Let A be any set hard for $\mathcal{C}$ under AC^0-reductions. Let B be any set in $\mathcal{C}$. Clearly, B is AC^0-reducible to A. We seek to show that B is actually NC^0-reducible to A.

The proof strategy will be similar to the one followed in Stage 1 of the proof of Theorem 2. As there, we define a set $B' \in \mathcal{C}$, and use the AC^0-reduction C_n from B' to A as a starting point for a reduction from B to A. B' will have been chosen so that a suitable non-uniform restriction of C_n will give us an NC^0 reduction from B to A.

We define B' to be the set of strings accepted by the following procedure:

> On input y, let $y = 1^k0z$. Reject if k does not divide $|z|$. Otherwise, break z into blocks of k consecutive bits each. Let these be $u_1u_2u_3\cdots u_q$. For each i, $1 \leq i \leq q$, let $v_i = 0$ if the number of ones in u_i equals 0 modulo 3; $v_i = 1$ if the number of ones in u_i equals 1 modulo 3; and $v_i = \epsilon$ otherwise. Accept iff $v_1v_2\cdots v_q \in B$.

It is easy to see that B' is Dlogtime-uniform NC^1 reducible to B. Hence, by the definition of proper complexity class, $B' \in \mathcal{C}$. Since A is hard for $\mathcal{C}$ under AC^0-reductions, there must exist an AC^0 circuit family C_n computing a reduction from B' to A. Let d be a bound on the depth of the family C_n. W.l.o.g. we can assume that C_n takes n input bits and has no more than n^d output bits. (Recall that the final $O(\log n)$ bits are used to encode a number that indicates how many of the output bits to use in the reduction. We refer to these bits as the "length-encoder bits".)

Let a be the constant (depending on the depth d of C_n) from Lemma 7. Let C'_m be the family of circuits for inputs of length $m = (2q)^{2a}$, where C'_m is obtained by taking circuit C_n for $n = 1 + (2q)^{2a-1} + m$ and setting the first $1 + (2q)^{2a-1}$ bits to $1^{(2q)^{2a-1}}0$.

By Corollary 9, for all large m, there is a restriction τ_m such that τ_m transforms C'_m into an NC^0 circuit, and τ_m assigns $*$ to at least three variables in *each* block of length $(2q)^{2a-1}$. We will now show how to extend τ_m to obtain a further restriction of C'_m having only q variables, and having the length-encoder bits set to constant values. We will call this new circuit family D_q. This circuit family D_q will be our NC^0 reduction from B to A.

Each of $O(\log n)$ length encoder bits depends only on a constant number of remaining input bits. Thus, the encoder bits depend only on $O(\log n)$ blocks. For each of these $O(\log n)$ blocks, fix *all* the remaining bits to constants so that the number of 1's in each block is 2 modulo 3 (this is always possible because we have at least 3 unset bits in each block). The length-encoder bits are fixed and we still have $2q - O(\log q)$ blocks that we have not tampered with. Pick all but the first q blocks and also fix their inputs so that the number of 1's in each of them is 2 modulo 3. For each of the remaining q blocks, set all but one bit in each block so that the total number of 1's in the block is 0 modulo 3 (again this is possible since there are at least three unset bits in each block). The result is a circuit with exactly q input bits and a fixed output size. (That is, all of the length-encoder bits have been set to constant values by setting the bits on which they depend. Let this value be r. Thus we can delete the length encoder bits and all but the first r output bits.) Call this circuit family D_q. Notice that it has size polynomial in q because q is $\Omega(n^\epsilon)$ for some $\epsilon > 0$ and C_n is of size polynomial in n. Also note that D_q is obtained from C_n by restricting attention to inputs of the form $y = 1^{(2q)^{2a-1}}0z$, where z is a string with exactly q $*$'s. For any string x of length q, denote by $y(x)$ the result of plugging the q bits of x into the q positions in z, and note that the algorithm for B' accepts $y(x)$ if and only $x \in B$ (because the algorithm for B' decodes z to obtain x).

Since C_n reduces B' to A, we see that D_q reduces B to A. This is the desired NC^0-reduction from B to A. ∎

3.2.3 The Isomorphism Theorem

The Isomorphism Theorem is an immediate consequence of the Gap Theorem and Corollary 6.

Theorem 11 (Isomorphism Theorem) *Let $\mathcal{C}$ be any proper complexity class. All sets complete for $\mathcal{C}$ under (non-uniform) AC^0 reductions are AC^0-isomorphic. Furthermore the isomorphisms are computable and invertible by depth three AC^0 circuits.*

Note that this is, in some sense, a true analog of the Berman-Hartmanis conjecture, since it presents a natural notion of computation (which then yields natural notions of reducibility and isomorphism) and it shows that in this setting the complete sets coincide with the isomorphism type of the standard complete set.

In the rest of this section, we include a few observations that we hope will shed additional light on the original Berman-Hartmanis conjecture.

Recall that our Gap Theorem says that, for sets A that are complete for NP under AC^0 reductions, all sets reducible to A under AC^0 reductions are already reducible to A under NC^0 reductions. Let us now consider another type of possible "collapse" and investigate its consequences.

For the purposes of this section, let us say that a set A is *special* if it has the property that the class of sets reducible to A via uniform AC^0-reductions is equal to the class of sets reducible to A via reductions computed by uniform threshold circuits of depth five.[2] Since AC^0 cannot compute the MAJORITY function (computable by a depth one threshold circuit), clearly no finite set is special, and it is easy to construct many other sets that are not special.

On the other hand, any set A that is complete for NP under AC^0-reductions clearly *is* special, since NP is exactly the class of sets reducible to A under either form of reducibility.

Are all sets that are complete for NP under *polynomial-time reductions* special? If so, then (as we sketch below), a non-uniform version of the Berman-Hartmanis Conjecture is true.

Let A be any set that is complete for NP under polynomial-time reductions. We observe first of all that there is a set A' that is both

1. P-isomorphic to SAT, and
2. AC^0-reducible to A.

Let f be a polynomial-time reduction from SAT to A, and let f be computed by a polynomial-time machine M. Let A' be the set $\{< x, y, C_1, \ldots, C_m >|\ C_1$ is an initial configuration of M on input x, each C_i yields C_{i+1} via one computation step of M, C_m is the final configuration of M producing output y, and $y \in A\}$. Since SAT is reducible to A' via a length-increasing and invertible reduction, it follows from [BH77] that SAT is P-isomorphic to A'.

If in addition A is special, then by Theorem 10 there is a (non-uniform) NC^0 reduction from A' to A, and by Theorem 2, there is a (non-uniform) superprojection reducing A' to A. In particular, this implies that there is a length-increasing and invertible P/poly reduction from SAT to A, and hence A is P/poly-isomorphic to SAT. (Most of the non-uniformity here can in fact be eliminated. The NC^0 reduction from A' to A can in fact be made DTIME$(n^{\log^{O(1)} n})$-uniform, by noting that the pseudorandom output produced by Nisan's generator [Ni92] must frequently produce restrictions satisfying the condition of Corollary 9; we leave the details to the reader. It follows that, if A is special, then it is quasipolynomial-time isomorphic to SAT.)

We do not view this as strong evidence in favor of the Berman-Hartmanis Conjecture, but we do feel that it casts the problem in a new light.[3]

---

[2] See the comments in Section 2, where the notion of a proper complexity class is defined, to understand this reference to threshold circuits.

[3] Indeed, the recent results of [AAIPR97] give an explicit construction of an NP-complete set that is *not* special.

3.3 Uniform Versus Non-uniform Gap Theorems

Theorem 10 cannot be made Dlogtime-uniform. That is, there exist Dlogtime-uniform AC^0-complete sets for NC^1 that are not Dlogtime-uniform NC^0-complete. It is worth mentioning at this point that it is not entirely clear what should be the "right" notion of uniformity for NC^0 circuits, since Dlogtime Turing machines can do things that cannot be done by any NC^0 circuit (and thus one might want to consider a more restrictive version of uniformity when discussing uniform NC^0, so as not to allow the uniformity machine to overwhelm the computation done by the NC^0 circuit itself). However, we show here that even under this "powerful" notion of uniformity for NC^0, Theorem 10 fails to hold.

Theorem 12 *For any class $\mathcal{C}$ closed under Dlogtime-uniform AC^0 reductions and having a set complete under AC^0 reductions, there are Dlogtime-uniform AC^0-complete sets for $\mathcal{C}$ that are not Dlogtime-uniform NC^0-complete.*

Proof. The proof uses ideas from [Ag95]. Let A be any Dlogtime-uniform AC^0-complete set for $\mathcal{C}$, and let L be any set in NTIME(n). Let χ_L^m denote the characteristic bit-vector of L for the first m strings of Σ^* (in lexicographic order). Define a set A_L as:

$$A_L = \{xz \mid z \in A \ \wedge \ |x| = |z| \ \wedge \ x = \chi_L^{|z|}\}.$$

The set A_L reduces to A via a Dlogtime-uniform AC^0 reduction: for inputs of size $2n$, the reduction circuit computes the bit-vector χ_L^n (this can be done by a Dlogtime-uniform AC^0 circuit since $L \in \mathrm{NTIME}(n)$ and the size of each of the first n strings of Σ^* is at most $\log n$), compares it with the first n bits of the input, and outputs the last n bits if all the bits match, and otherwise outputs some fixed string not in A. For inputs of odd size, the reduction circuit just outputs some fixed string not in A. It is easy to see that the entire circuit can be made Dlogtime-uniform. Therefore, $A_L \in \mathcal{C}$.

The set A reduces to A_L via a Dlogtime-uniform AC^0 reduction: for input of size n, the reduction circuit first outputs χ_L^n and then outputs the n input bits. This circuit too is a Dlogtime-uniform AC^0 circuit. Therefore, A_L is Dlogtime-uniform AC^0-complete for $\mathcal{C}$.

Now, let us assume that Dlogtime-uniform AC^0-complete sets for $\mathcal{C}$ are also Dlogtime-uniform NC^0-complete. Thus, A_L is Dlogtime-uniform NC^0-complete for $\mathcal{C}$. By Lemma 13 (below), A_L is complete under Dlogtime-uniform NC^0 reductions for which there is a constant $c \geq 1$ such that the length of the output produced on input x has length at least $|x|/2^c$. Let $\{C_n\}$ be such a reduction of the set 1^* to A_L.

We will now give a deterministic procedure that accepts L in linear time. On input x, $|x| = n$, the procedure considers the circuit $C_{2^{n+2+c}}$. This circuit, on input $1^{2^{n+2+c}}$, must output a string in A_L of size ℓ with $\ell \geq 2^{n+2}$. Since the output is in A_L, the first $\ell/2$ bits of it are $\chi_L^{\ell/2}$, and therefore, the m^{th} output bit of the circuit is $\chi_L(x)$ where m is the position of the string x in the lexicographic order. (This is because $m \leq 2^{n+1} \leq \ell/2$.)

The circuit $C_{2^{n+2+c}}$ is a Dlogtime-uniform NC^0 circuit. Thus the following functions are computable in time logarithmic in input size (in other words, in time linear in n):

- $(2^{n+2+c}, g, R) \mapsto g'$, where g' is the name of the right input to gate g in $C_{2^{n+2+c}}$ if g is a gate in $C_{2^{n+2+c}}$. (The output is * if g is not a gate in $C_{2^{n+2+c}}$.)

- $(2^{n+2+c}, g, L) \mapsto g'$, where g' is the name of the left input to gate g in $C_{2^{n+2+c}}$ if g is a gate in $C_{2^{n+2+c}}$. (The output is * if g is not a gate in $C_{2^{n+2+c}}$.)
- $(2^{n+2+c}, g) \mapsto T$, where T is the type of gate that g is, i.e., $T \in$ {AND, OR, NOT, INPUT, 0, 1}. (Note that, since $C_{2^{n+2+c}}$ takes input in 1^*, all "INPUT" gates are essentially constant 1 gates.)

Our procedure to determine if x is in L first computes the number m such that m is the position of x in the lexicographic order. Then it determines g, the gate that computes the m^{th} bit of circuit $C_{2^{n+2+c}}$. (We've established that such an output gate must exist. Using a reasonable notion of uniformity, we should be able to compute the name of this output gate.) Now it computes the right and left inputs to the gate g, right and left inputs of these two gates and so on until the entire tree for the gate g has been computed. Since this tree has constant height, this computation takes only linear time (in n). Finally, the procedure computes the value output by the gate g. This can be done because it knows what sort of gates are in the tree of g, and that all of the inputs are either constants or, if they are circuit inputs, they are set to 1. The procedure accepts iff this bit is 1.

Thus, $L \in$ DTIME(n). Since L was an arbitrary language in NTIME(n), we have shown that NTIME(n) = DTIME(n), which is a contradiction, since it is known that DTIME(n) is properly contained in NTIME(n) [PPS83]. ∎

The proof is now complete, except for the following technical lemma.

Lemma 13 *For any class $\mathcal{C}$ closed under Dlogtime-uniform AC^0 reductions, all sets complete under Dlogtime-uniform NC^0 reductions are complete under Dlogtime-uniform NC^0 reductions for which there is a constant $d \geq 1$ such that the output produced on input x has length at least $|x|/d$.*

Proof. Let A be hard for $\mathcal{C}$under Dlogtime-uniform AC^0 reduction, let B be any set in $\mathcal{C}$, and let D be the set $\{0x \mid x \in B\} \cup 1^*$. D is Dlogtime-uniform AC^0-reducible to B, and thus there is a Dlogtime-uniform NC^0 reduction $\{C_n\}$ reducing D to A. We will show that the NC^0 reduction from B to A obtained by composing the obvious projection from B to D with $\{C_n\}$ satisfies the conditions of the lemma.

Consider the circuit C_n. If we set the first input bit to 1, note that each remaining input bit must influence at least one output bit, because if all bits are set to 1, C_n must produce an element of A as output, but if any bit is set to 0, then the output produced is not in A. The bound follows because there is some constant d such that each output bit depends on at most d input bits. ∎

4 Conclusions

In closing, let us summarize our results. Berman and Hartmanis conjectured in [BH77] that all sets complete for NP under poly-time many-one reductions are P-isomorphic. Following the lead of [ABI93] we have considered the analogous question, where polynomial-time reductions and isomorphisms are replaced by AC^0-computable reductions and isomorphisms. In [ABI93] it was shown that all sets complete under AC^0 *projections* are AC^0-isomorphic.

We have improved that result to show that all sets complete under NC^0 reductions are AC^0-isomorphic. To give some indication of the nature of this improvement, note that (1) projections are a very simple sort of NC^0 reduction, and (2) projections are easily invertible in AC^0, whereas NC^0 reductions are not invertible in polynomial time unless P=NP. (Invertibility is relevant here, since the likely existence of non-invertible poly-time reductions is one of the main considerations leading many researchers to conjecture that the Berman-Hartmanis conjecture is false [JY85].) We use our results about NC^0-reducibility, superprojections, and an inherent gap in the power of reductions to prove a true analog of the Berman-Hartmanis conjecture. (That is, the sets complete under AC^0 reductions are all AC^0-isomorphic.) Finally, we show that our approach cannot yield the analog of the conjecture for Dlogtime-uniform AC^0 reductions.

We especially call attention to the following problems:

1. Assuming the existence of a function that is one-way in a very strong average case sense, is it possible to construct a counter-example to the original Berman-Hartmanis conjecture?

2. Is there any natural class $\mathcal{C}$ (larger than P) such that there is a set hard for $\mathcal{C}$ under polynomial-time many-one reductions that is not hard under (non-uniform) superprojections?

3. Are there *any* natural classes $\mathcal{C}$ (larger than P) such that the classes of sets hard for $\mathcal{C}$ under (a) polynomial-time many-one reductions, and (b) uniform AC^0 reductions, differ?[4] (We have already shown that uniform NC^0 reductions do yield a strictly smaller class of complete sets.)
   Note in this regard that [Ar95] shows (a) there is a poly-time reduction f: PARITY $\leq_m^p$ Clique such that $x \in$ PARITY implies $f(x)$ has a very large clique, and $x \notin$ PARITY implies $f(x)$ has only very small cliques, and (b) no AC^0 reduction can have this property. Nonetheless, there is no version of the Clique problem (or any other NP-complete problem) that is currently known not to be complete under AC^0 many-one reductions.

4. Is there any class $\mathcal{C}$ such that Dlogtime-uniform AC^0-complete sets for $\mathcal{C}$ are all Dlogtime-uniform AC^0-isomorphic?[5]

Acknowledgments

We acknowledge helpful conversations with M. Ajtai, S. Arora, R. Boppana, O. Goldreich, M. Ogihara, D. van Melkebeek, R. Pruim, D. Sivakumar, and M. Saks. We also thank the anonymous referee for helpful suggestions.

[4]This question has been answered affirmatively by [AAIPR97], where it is shown that there is a set complete for NP under uniform NC^1 reductions that is *not* complete under AC^0 reductions.

[5]A first step in this direction is taken in [AAIPR97], where a P-uniform version of our isomorphism theorem is established.

References

[Ag94] M. Agrawal, *On the isomorphism problem for weak reducibilities*, J. Computer Sys. Sci. **53** (1996) 267–282.

[Ag95] M. Agrawal, *DSPACE*$(n)\stackrel{?}{=}$*NSPACE*(n)*: A degree theoretic characterization*, J. Computer Sys. Sci. **54** (1997) 383–392.

[Ag96] M. Agrawal, *For completeness, sublogarithmic space is no space*, Manuscript.

[AA96] M. Agrawal and E. Allender, *An isomorphism theorem for circuit complexity*, in Proc. 11th Annual IEEE Conference on Computational Complexity (1996) pp. 2–11.

[AAIPR97] M. Agrawal and E. Allender, R. Impagliazzo, T. Pitassi, and S. Rudich, *Reducing the complexity of reductions*, in Proc. 29th ACM Symposium on Theory of Computing (1997) pp. 730–738.

[Aj83] M. Ajtai, Σ_1^1 *formulae on finite structures*, Annals of Pure and Applied Logic **24** (1983) 1–48.

[Al89] E. Allender, *P-uniform circuit complexity*, J. ACM **36** (1989) 912–928.

[ABI93] E. Allender, N. Immerman, and J. Balcázar, *A first-order isomorphism theorem*, SIAM Journal on Computing **26** (1997) 557–567.

[AG91] E. Allender and V. Gore, *Rudimentary reductions revisited*, Information Processing Letters **40** (1991) 89–95.

[AS92] N. Alon and J. Spencer, *The Probabilistic Method*, John Wiley and Sons, (1992).

[Ar95] Sanjeev Arora, AC^0*-reductions cannot prove the PCP theorem*, manuscript, 1995.

[BDG88] J. Balcázar, J. Díaz, and J. Gabarró, *Structural Complexity I and II*, Springer-Verlag, 1988, 1990.

[BIS90] David Mix Barrington, Neil Immerman, Howard Straubing, *On Uniformity Within* NC^1, J. Computer Sys. Sci. **41** (1990), 274-306.

[B95] P. Beame, *A switching lemma primer*, manuscript, available from `http://www.cs.washington.edu/homes/beame/papers.html`.

[BH77] L. Berman and J. Hartmanis, *On isomorphism and density of NP and other complete sets*, SIAM J. Comput. **6** (1977) 305–322.

[BS90] Ravi Boppana and Michael Sipser, *The complexity of finite functions*, in J. van Leeuwen, ed. *Handbook of Theoretical Computer Science, Vol. A: Algorithms and Complexity*, Elsevier, 1990, pp. 757–804.

[Bo72] A. Borodin, *Computational complexity and the existence of complexity gaps*, J. ACM **19** (1972), 158–174.

[ER60] P. Erdös and R. Rado, *Intersection theorems for systems of sets*, J. London Math. Soc. **35** (1960) 85–90.

[Fo95] L. Fortnow and S. Laplante, *Circuit lower bounds à la Kolmogorov*, Information and Computation **123** (1995) 121–126.

[FSS84] Merrick Furst, James Saxe, and Michael Sipser, *Parity, Circuits, and the Polynomial-Time Hierarchy,* Math. Systems Theory **17** (1984), 13-27.

[Hå87] J. Håstad, *One-Way Permutations in* NC^0*,* Information Processing Letters **26** (1987) 153–155.

[HILL90] J. Håstad, R. Impagliazzo, L. Levin, and M. Luby, *Construction of a pseudo-random generator from any one-way function*, ICSI Technical Report, **No. 91-068** (1990).

[Jo75] Neil Jones, *Space-Bounded Reducibility among Combinatorial Problems,* J. Computer Sys. Sci. **11** (1975) 68–85.

[JY85] D. Joseph and P. Young, *Some remarks on witness functions for non-polynomial and non-complete sets in NP*, Theoretical Computer Science **39** (1985) 225–237.

[JPY94] D. Joseph, R. Pruim, and P. Young, *Collapsing degrees in subexponential time*, Proc. 9th Structure in Complexity Theory Conference (1994) pp. 367–382.

[KLD86] Ker-I Ko, Timothy J. Long, and Ding-Zhu Du, *On one-way functions and polynomial-time isomorphisms*, Theoretical Computer Science **47** (1986) 263–276.

[KMR88] S. Kurtz, S. Mahaney, and J. Royer, *Collapsing degrees*, J. Computer Sys. Sci. **37** (1988) 247–268.

[KMR90] S. Kurtz, S. Mahaney, and J. Royer, *The structure of complete degrees*, in A. Selman, editor, *Complexity Theory Retrospective*, Springer-Verlag, 1990, pp. 108–146.

[KMR95] S. Kurtz, S. Mahaney, and J. Royer, *The isomorphism conjecture fails relative to a random oracle*, J. ACM **42** (1995) 401–420.

[LV93] M. Li and P. Vitányi, *An Introduction to Kolmogorov Complexity and its Applications*, Springer–Verlag, 1993, p. 99.

[Li94] Steven Lindell, *How to define exponentiation from addition and multiplication in first-order logic on finite structures*, (manuscript). This improves an earlier characterization that appears in: Steven Lindell, *A purely logical characterization of circuit uniformity*, Proc. 7th Structure in Complexity Theory Conference (1992) pp. 185–192.

[Lu96] Michael Luby, *Pseudorandomness and Cryptographic Applications*, Princeton University Press (1996).

[Ni92] Noam Nisan, *Using Hard Problems to Create Pseudorandom Generators*, MIT Press (1992).

[PPS83] W. J. Paul, N. Pippenger, E. Szemerédi, and W. Trotter, *On determinism versus non-determinism and related problems*, Proc. Foundations of Computer Science (1983) pp. 429–438.

[Ro95] J. Rogers, *The isomorphism conjecture holds and one-way functions exist relative to an oracle*, J. Computer Sys. Sci. **54** (1997) 412–423.

[Se92] A. Selman, *A survey of one way functions in complexity theory*, Mathematical Systems Theory **25** (1992) 203–221.

[Tr64] B. A. Trakhtenbrot, *Turing computations with logarithmic delay*, Algebra i Logika **3** (1964) 33–48.

5 Appendix

In this appendix, we give a proof of Lemma 7. Although there are several people in the community who were already aware that established techniques can be used to prove this lemma, we have been unable to find a published statement of this fact, and hence we provide the proof for completeness.

In an earlier version of this work [AA96], the argument that was presented required a stronger lemma, stated in terms of "blockwise" restrictions. Since this stronger version of the lemma may be useful in some other applications, we have chosen to include the proof of this stronger lemma in the appendix. Note that we have not tried to obtain the best constants in this proof. Instead our goal was to make the proof as simple as possible.

Let $b \in \mathbf{N}$. A b-block restriction on n variables is an assignment to the variables $x_1, \ldots, x_n$ with the property that for all $j \geq 0$ either *all* of the variables $x_{jb+1}, \ldots, x_{jb+b}$ are given values, or *none* of them are. (The variables $x_{jb+1}, \ldots, x_{jb+b}$ together constitute the j-th b-block. If n is not a multiple of b, then the final block may have length less than b.)

For $a, b \in \mathbf{N}$, an a-random b-block restriction ρ_a on n variables is a b-block restriction chosen according to the following process: For each block of b variables in turn, with probability $(n/b)^{1/a-1}$ set all of the variables in that block to *, and for any $v \in \{0,1\}^b$, set the variables in the block to the vector v with probability $(1-(n/b)^{1/a-1})/2^b$. (Note that the a-random restrictions used in Section 3.2 are 1-block restrictions.)

We now restate Lemma 7 using the more general notion of block restriction.

Lemma 7 *For any AC^0 reduction computed by a family of circuits $\{C_n\}$, there exists an $a \in \mathbf{N}$ such that for any constant b, with probability $1-o(1)$, an a-random b-block restriction ρ_a on n variables transforms C_n into an NC^0-circuit with $\Omega(n^{1/a})$ b-blocks of input variables.*

Proof. Without loss of generality, the circuit family $\{C_n\}$ is

- of depth $k \geq 2$
- leveled (meaning that the circuit has n inputs $x_1, \ldots, x_n$, and n negated inputs $\neg x_1, \ldots, \neg x_n$, and that these inputs feed into AND gates (or OR gates), and that AND gates feed into OR gates, and vice-versa)

- of bottom fan-in 1 (meaning that the gates that $x_1, \ldots, x_n$, and $\neg x_1, \ldots, \neg x_n$ feed into have fan-in only 1).

The constant a is chosen to be $5(k-1)$.

To simplify notation, we will assume throughout the proof that n is a multiple of b. Let $m = n/b$.

By a simple application of the Chernoff bound (see, e.g., [AS92]), with probability $1 - o(1)$, the number of b-blocks that are set to * is between $\frac{1}{2}m^{1/a}$ and $2m^{1/a}$. Thus it is sufficient to show that, with probability $1 - o(1)$, C_n is transformed into an NC^0-circuit, *given* that the number of blocks that are set to * lies within this range.

Since we will assume that at least $m - 2m^{1/a}$ blocks are set to values in $\{0,1\}^b$, it will be convenient to consider the following process for picking the random restriction ρ_a. One-by-one, a block is picked and set to a value in $\{0,1\}^b$ (with all unset blocks being equally likely, and all values in $\{0,1\}^b$ equally likely after the block is picked). We will see that after $m - 2m^{1/a}$ blocks have been set in this manner, the circuit C_n will be transformed into an NC^0-circuit with probability $1 - o(1)$. This process will actually proceed in $k-1$ stages, where the first $k-2$ stages each decrease the depth of the circuit by one with probability $1 - o(1)$ (this is guaranteed by Claim 15), and the final stage changes the depth two circuit into an NC^0-circuit with probability $1 - o(1)$ (this is guaranteed by Claim 16). Setting any additional blocks will not alter the fact that the circuit is transformed into an NC^0-circuit. Thus, given that we are assuming that the number of blocks that are set to * lies within the acceptable range, this will prove the lemma.

More formally, let $f_k(m) = m^{(k-2)/(k-1)}$, and let $r(m,k) = \frac{m^{1/(4b(k-1))}}{b^{k-2}}$. Claim 15 shows that we can take a depth k circuit on m blocks, having bottom-fan-in $r(m,k)$, and by setting some of the blocks randomly, obtain a circuit that is equivalent to a depth $k-1$ circuit with $f_k(m)$ blocks, having bottom fan-in $r(f_k(m), k-1)$. For $i < k$, define the function $f_k^{(i)}$ as follows: $f_k^{(1)}(m) = f_k(m)$, and $f_k^{(i+1)}(m) = f_k(f_{k-1}^{(i)}(m))$. Note that $f_k^{(i)}(m) = m^{(k-(i+1))/(k-1)}$. Also note that $r(f_k^{(i)}(m), k-i) = b^i r(m,k)$.

Applying Claim 15 $k-2$ times, we obtain a circuit with depth 2, on $f_k^{(k-2)}(m) = m^{1/(k-1)}$ blocks, having bottom fan-in $r(f_k^{(k-2)}(m), 2) = b^{k-2} r(m,k) = m^{1/(4b(k-1))}$. Now we apply Claim 16 and obtain a circuit on $m^{1/(4(k-1))} = \omega(n^{1/a})$ blocks with bottom fan-in $O(1)$. ∎

Before we can present the proofs of Claims 15 and 16, we will need some additional notation.

Let $R_{n,b}^{\ell}$ denote the set of all b-block restrictions to the variables $x_1, \ldots, x_n$ that have exactly ℓ blocks of b variables unset. It will usually be the case that n and b are understood from context, and thus we will usually delete the subscripts.

Note that

$$|R^{\ell}| = \binom{m}{\ell} 2^{n - \ell b}.$$

Thus there is a constant c_1 such that, for any $\rho \in R_{n,b}^{\ell}$,

$$K(\rho | n, \ell, b) \leq c_1 + n - \ell b + \log \binom{m}{\ell}$$

(Here, we are assuming some fixed system for encoding restrictions; the particulars are irrelevant. We are assuming that the reader is familiar with Kolmogorov complexity. For details, see [LV93].) Note also that, for $s > 0$,

$$|R^\ell|/|R^{\ell-s}| = 2^{-sb} \cdot \frac{\binom{m}{\ell}}{\binom{m}{\ell-s}} \geq \left(\frac{m-\ell}{2^b \ell}\right)^s$$

In our applications, s will be much larger than b, and n/b will be much larger than ℓ, and thus $R^{\ell-s}$ is much smaller than R^ℓ. In particular, although at first it may seem counterintuitive, a Kolmogorov-random element $\rho \in R^\ell$ has *greater* Kolmogorov complexity than any element ρ' of $R^{\ell-s}$ (even if ρ' is the result of assigning values to some of the unset variables of ρ). This simple fact is a key observation underlying the proof of the switching lemma.

In the following, we will not make any meaningful distinction between a circuit C_n and the bit string that describes an encoding of C_n. Thus given any circuit C_n of size t, any gate in the circuit can be described using an additional $\log t$ bits, and thus a pair (C_n, i) is a description of the function computed by gate i of C_n. If gate i is on level two of C_n (and thus by our assumptions it is an OR of ANDs, where the ANDs are connected to input literals), then the pair (C_n, i) immediately gives a DNF formula $F_{n,i}$ where this formula lists the AND gates feeding into gate i (listing them in the order in which they appear in C_n) where each AND gate is presented by the literals feeding into that gate (where the ordering on the variables imposes an order on the literals).

Given any DNF formula F and restriction ρ, $F|_\rho$ is the formula that results by (1) deleting from F all terms (i.e., conjunction of literals) that are made false by ρ, and (2) replacing each remaining term C by the term $C|_\rho$ obtained by deleting from C all the literals that are satisfied by ρ. It should be emphasized that this is a syntactic operation, and that $F|_\rho$ is a syntactic object. A formula with an empty term denotes the constant function 1; a formula with no terms denotes the constant function 0.

Given any DNF formula F, we follow [B95] and define the *canonical decision tree for F* (denoted $T(F)$) as follows.

1. If F has no terms, then $T(F)$ is a single leaf labelled 0.

2. If the *first* term in F is empty, then $T(F)$ is a single leaf labelled 1.

3. If the first term C_1 of F is not empty, then let F' be the rest of F (i.e., $F = C_1 \vee F'$). The decision tree $T(F)$ begins by querying *all* of the variables that appear in any b-block containing a variable in C_1. (That is, if C_1 has r variables, the tree $T(F)$ begins with a complete binary tree on the $r'b \leq rb$ variables in the r' blocks that contain variables appearing in C_1.) Each leaf v_σ of this complete binary tree is reached by some path labelled by a restriction σ recording an assignment to the variables in C_1. Each such node v_σ is the root of a subtree in $T(F)$ given by $T(F|_\sigma)$. For any σ that satisfies C_1, $T(F|_\sigma)$ is a single node labelled 1. For all other σ, $T(F|_\sigma) = T(F'|_\sigma)$.

For a restriction π, let $\mathrm{Dom}(\pi)$ denote $\pi^{-1}(\{0,1\})$, i.e., the variables set by π. For $S \subseteq \{x_1, \ldots, x_n\}$, let $\pi|_S$ denote the restriction

$$\pi|_S(x_i) = \begin{cases} *, & x_i \notin S; \\ \pi(x_i) & x_i \in S \end{cases}$$

For a Boolean expression F, we will sometimes write $\pi(F)$ for $F|_\pi$.

Following [B95], we will show that for any Kolmogorov-random ρ and for any DNF formula F, the height of $T(F|_\rho)$ (denoted $|T(F|_\rho)|$) is small.

It is easy to observe that if f is a Boolean function having a decision tree with height bounded by s, then it has a DNF formula with term size bounded by s. (The disjuncts consist of the conjunctions, over all paths of the tree ending in 1, of the literals queried along those paths.) Similarly, such an f has a CNF formula with term size bounded by s (since $\neg f$ clearly has a small decision tree, and hence a DNF formula with small term size). Note that saying that $|T(F)|$ is small is a much stronger statement than merely saying that the function computed by F has a decision tree with small height; this is because $T(F)$ may well be a very inefficient decision tree.

Lemma 14 *(Håstad Switching lemma) There is a constant c_4 such that, for any DNF formula F in n variables with terms of length at most r, and for any b and for any $0 < s < \ell \leq n/b$, and for any $\rho \in R^\ell_{n,b}$, if*

$$K(\rho|F, n, \ell, s, b) > c_4 + n - \ell b + sb\log(16r) + \log\binom{n/b}{\ell - s}$$

then $|T(F|_\rho)| < sb$.

At this point, we can present the proofs of the Claims 15 and 16.

Claim 15 *Let C be a leveled circuit on m blocks with depth $k \geq 3$ and with bottom fan-in $\leq r = r(m,k)$, and having no more than 2^r gates at levels 2 and above. Then for all large n, with probability $1 - o(1)$, a random b-block assignment ρ_a leaving $f_k(m)$ blocks unset has the property that $C|_\rho$ is computed by a depth $k-1$ circuit with bottom fan-in $r(f_k(m), k-1)$ and with no more than $2^{r(f_k(m),k-1)}$ gates at level 2 and above.*

Proof. (of Claim 15) Assume without loss of generality that the bottom level gates are AND gates. The other case follows by DeMorgan's laws.

Let $s = r = r(m,k)$ and let $\ell = f_k(m)$. Note that for a randomly chosen ρ in $\rho \in R^\ell_{n,b}$, $K(\rho|C, n, \ell, s, b)$ is high. In particular, with probability $(1 - 1/\log n)$, we have that $K(\rho|C, n, \ell, s, b) \geq n - \ell b + \log\binom{m}{\ell} - \log\log n$.

Consider any of the gates on level two of C, and consider the DNF formula F represented by this gate. Before we go further, we must argue that

$$K(\rho|F, n, \ell, s, b) > c_4 + n - \ell b + sb\log(16s) + \log\binom{n/b}{\ell - s}$$

and that we can therefore appeal to the Switching Lemma.

Since there are most 2^s gates at level two of C, it is easy to see that $K(\rho|C, n, \ell, s, b) \leq K(\rho|F, n, \ell, s, b) + c_5 + s$. (The constant c_5 is enough bits to say "count the number p of gates in C at level 2 and above, and use the next $\lceil \log p \rceil \leq s$ bits to identify one of the OR gates on level two of C. Construct the DNF formula F computed by this gate, and then use the remaining bits of information to construct ρ from F.")

Thus $K(\rho|F,n,\ell,s,b) \geq K(\rho|C,n,\ell,s,b) - c_5 - s \geq n - \ell b + \log\binom{m}{\ell} - c_5 - s - \log\log n$. This value is bounded below by $c_4 + n - \ell b + sb\log(16s) + \log\binom{m}{\ell-s}$ if and only if

$$\log\frac{\binom{m}{\ell}}{\binom{m}{\ell-s}} \geq (c_4+c_5) + s + sb(\log 16s) + \log\log n.$$

Thus it is sufficient to show that

$$\log\left(\frac{m-\ell}{\ell}\right)^s \geq (c_4+c_5) + s + sb(\log 16s) + \log\log n.$$

This is equivalent to showing that

$$m - \ell \geq 2^d \ell (16s)^b (\log n)^{1/s}$$

where $d = 1 + (c_4+c_5)/s$. Note that $d < 2$ for all large n, since $s = r(n,k)$. Since there is some $\gamma > 0$ such that $\ell s^b \leq n^{1-\gamma}$, it is thus sufficient to show that $n/b - \ell \geq 2^{4b+2} n^{1-\gamma}\log n$, which is clearly true for all large n. Thus we have established that, for the chosen values of ℓ, s, and ρ, the hypothesis of the Switching Lemma is satisfied.

Thus by the Switching Lemma, each of the OR gates at level two of $C|_\rho$ has a decision tree of height $\leq sb$, and thus each of those OR gates can be computed by CNF circuits with bottom fan-in sb. The circuit that results by substituting these CNF circuits into $C|_\rho$ has AND gates on levels two and three. If we merge these two levels, we obtain a circuit having depth $k-1$, having ℓ blocks of b variables, and with bottom fan-in bounded by $sb = r(\ell, k-1)$ and with no more than $2^{r(\ell,k-1)}$ gates at level two and above. ∎ (of Claim 15)

Claim 16 *Let n be sufficiently large. Let f be a function on n inputs such that $|f(x)| \leq n^B$ for all x of length n, where f is computed by a depth-two circuit C having bottom fan-in $\leq r = r(m) = m^{1/4b}$. Then with probability $1 - o(1)$, a randomly-chosen ρ leaving $m^{1/4}$ b-blocks unset has the property that each output bit of $C|_\rho$ has a decision tree of height at most sb, where $s = 4B$.*

Proof. Assume without loss of generality that the output bits of C are ORs. The other case follows easily.

Pick $\ell = m^{1/4}$, and let ρ be randomly chosen element of $R^{\ell}_{n,b}$. Thus, with probability at least $1 - 1/\log n$, $K(\rho|C,n,\ell,s,b)$ is greater than $n - \ell b + \log\binom{m}{\ell} - \log\log n$.

Consider any of the output gates of C, and consider the DNF formula F represented by this gate. Before we go further, we must argue that

$$K(\rho|F,n,\ell,s,b) > c_4 + n - \ell b + sb\log(16r) + \log\binom{n/b}{\ell - s}$$

and that we can therefore appeal to the Switching Lemma.

Since there are most n^B output gates in C, it is easy to see that $K(\rho|C,n,\ell,s,b) \leq K(\rho|F,n,\ell,s,b) + c_5 + B\log n$. Thus $K(\rho|F,n,\ell,s,b) \geq K(\rho|C,n,\ell,s,b) - c_5 - B\log n \geq$

$n - \ell b + \log\binom{m}{\ell} - c_5 - B\log n - \log\log n$. This value is bounded below by $c_4 + n - \ell b + sb\log(16r) + \log\binom{m}{\ell-s}$ if and only if

$$\log\frac{\binom{m}{\ell}}{\binom{m}{\ell-s}} \geq (c_4+c_5) + B\log n + sb(\log 16r) + \log\log n.$$

Thus it is sufficient to show that

$$\log\left(\frac{m-\ell}{\ell}\right)^s \geq (c_4+c_5) + B\log n + sb(\log 16r) + \log\log n.$$

By our choice of s this is equivalent to showing that

$$\frac{n}{b} - \ell \geq 2^{(c_4+c_5)/s} n^{1/4}\ell(16r)^b(\log n)^{1/s}.$$

Since $n^{1/4}r^b\ell = \Theta(n^{3/4})$, the hypothesis of the Switching Lemma is satisfied for all large n.

Now the claim follows immediately from the Switching Lemma. ■ (of Claim 16)

Thus it will suffice to give a proof of the Switching Lemma.

Proof. (of the Switching Lemma)

Let F, ℓ, s, n, t, ρ be as in the hypothesis of the lemma. We will prove the contrapositive. That is, we'll show that if $T(F|_\rho)$ contains a path π of length at least sb, then $K(\rho|F, n, \ell, s, b) \leq c_4 + n - \ell b + sb\log(16r) + \log\binom{n/b}{\ell-s}$.

The strategy is to construct $\rho' \in R^{\ell-s}$ extending ρ (i.e., fixing more variables), such that $K(\rho|\rho', F, \ell, s, b)$ is not much bigger than $K(\rho'|F, n, \ell, s, b)$. As we observed above, $K(\rho'|F, n, \ell, s, b)$ is small, because $R^{\ell-s}$ is small.

Let $\rho \in R^\ell$ and let π be any path in $T(F|_\rho)$ having length at least sb. Note that we may view π as a restriction (namely the restriction that gives to the $\geq sb$ variables queried along π the value determined along path π, and leaving the other $\leq n - sb$ variables unset).

We now define sequences of restrictions $\sigma_1, \sigma_2, \ldots$ and $\pi_1, \pi_2, \ldots$ (where each π_i in turn is decomposed into $\pi_i = \pi'_i\pi''_i$, where $\mathrm{Dom}(\pi'_i) = \mathrm{Dom}(\sigma_i)$). Our goal is to define $\rho' = \rho\sigma_1\pi''_1\sigma_2\pi''_2\ldots\sigma_k\pi''_k$ in such a way that ρ is easy to retrieve from ρ'.

Let C_{i_1} be the first term in F that is not set to 0 by ρ. (Such a term must exist, since otherwise the height of $T(F|_\rho)$ would be zero; by the same observation, $C_{i_1}|_\rho$ is not empty.) Thus $C_{i_1}|_\rho$ is the first term of $F|_\rho$. Let S_1 be the set of variables in $C_{i_1}|_\rho$, and let σ_1 be the unique restriction of the variables in S_1 that satisfies $C_{i_1}|_\rho$. (Thus, $S_1 = \mathrm{Dom}(\sigma_1)$.) Let P be the set of variables that are in b-blocks containing an element of S_1, and let $P_1 = P \setminus S_1$. Note that, by the definition of the canonical decision tree, $P \subseteq \mathrm{Dom}(\pi)$, since the tree queries all variables in each block touched by a term. Let $\pi'_1 = \pi|_{S_1}$ and $\pi''_1 = \pi|_{P_1}$, and let $\pi_1 = \pi|_P$, so that $\pi_1 = \pi'_1\pi''_1$. Note that π_1 is a prefix of path π.

If π_1 is in fact *all* of π, then the first part of the construction is over. Otherwise, by the definition of the canonical decision tree, it must be the case that there must be some C_{i_2} that is the first term in F that is not set to zero by $\rho\pi_1$, and $C_{i_2}|_{\rho\pi_1}$ is the first term of $F|_{\rho\pi_1}$, and it is this term that is explored next in the decision tree along path π. As above, let σ_2 be the unique assignment to the variables S_2 in $C_{i_2}|_{\rho\pi_1}$ that satisfies this term, let P_2 be the other variables in the blocks touched by S_2 and let $\pi'_2 = \pi|_{S_2}$ and $\pi''_2 = \pi|_{P_2}$.

Continue in this way until the entire path π is processed, maintaining the property that clause C_{i_h} is the first term of F not falsified by $\rho\pi_1 \dots \pi_{h-1}$, and $\rho\pi_1 \dots \pi_{h-1}\sigma_h$ satisfies C_{i_h}. It is important to observe also that each set S_h is nonempty.

Note that, since the length of π is at least sb, it must touch at least s b-blocks. A slight problem is caused by the fact that π may touch *more* than s blocks. Since we want ρ' to be in $R^{\ell-s}$, we simply consider the first stage k such that $\pi_1 \dots \pi_{k-1}\sigma_k$ touches at least s b-blocks, and redefine S_k to be the initial sequence of variables in clause C_{i_k} up through the s^{th} block touched, define P_k to be the other variables in those blocks, and redefine σ_k to be the setting to those variables that does not falsify the clause, $\pi'_k = \pi|_{S_k}$, and $\pi''_k = \pi|_{P_k}$. (Note that $k \le s$.) By setting ρ' equal to $\rho\sigma_1\pi''_1\sigma_2\pi''_2 \dots \sigma_k\pi''_k$, we have defined the desired element of $R^{\ell-s}_{n,b}$.

We now want to show that ρ is easy to recover from ρ'. To do this, we define a sequence $\beta_1, \dots, \beta_k$, where each β_h describes how σ_h and π'_h differ. Each β_h is a string of length r (recall that r is the bottom fan-in of the circuit C) over the alphabet $\{0, 1, *\}$, defined as follows. The j^{th} bit of β_h is $*$ if either (1) clause C_{i_h} of the original formula F has fewer than j variables, or (2) the j^{th} variable in clause C_{i_h} is not in S_h. The j^{th} bit of β_h is 0 if the j^{th} variable in C_{i_h} is in S_h and σ_h and π'_h agree on this variable. The j^{th} bit of β_h is 1 if the j^{th} variable in C_{i_h} is in S_h and σ_h and π'_h disagree on this variable. Note that each β_h contains at least one symbol that is not a $*$. The total number of non-* symbols is at least s and no more than sb (since the σ_h's together touch exactly s blocks). Let $\beta = \beta_1 \dots \beta_k$; thus β is a string of length $kr \le sr$. We will pad β with *'s at the end to obtain a string β' of length exactly sr.

Observe that, for some constant c_2, $K(\beta'|s, b, r) \le c_2 + sb(3 + \log r)$. (This is because β' is of the form $*^{j_1-1}b_1 *^{j_2-1} b_2 \dots *^{j_v-1} b_v ** \dots *$ where each $b_h \in \{0, 1\}$, and $v \le bs$ and each $j_h \le 2r$ (since no β_h is all *'s. Note that many of the numbers j_h may be equal to zero. By making $j_{v+1} = \dots j_{bs} = 0$ and making $b_v = \dots = b_{bs}$ we can encode β' using exactly bs such numbers. Thus we can encode β' as a sequence of exactly bs numbers j_h of exactly $\lceil \log r \rceil + 1$ bits, and a bit string of exactly bs bits encoding the b_h's.)

Now we claim that, for some constant c_4,

$$K(\rho|F, n, \ell, s, b) \le c_4 + n - \ell b + sb\log(16r) + \log \binom{n/b}{\ell - s}.$$

To see this, note that ρ' can be described with $c_1 + n - \ell b + sb + \log \binom{n/b}{\ell-s}$ bits. Since r can be obtained from F, β' can be described with $c_2 + sb(3 + \log r)$ bits. The bound on the Kolmogorov complexity of ρ now follows from an additional c_3 bits of information, encoding the following instructions:

> Find the first clause C_{i_1} in F that is not made false by ρ'. (Note that $\rho\sigma_1$ makes C_{i_1} true, and the further assignments made by ρ' cannot change this.)
>
> Let β_1 be the first r bits of β'. Use β_1 to find the elements of S_1. (If the j^{th} bit of β_1 is not *, then the j^{th} variable in C_{i_1} is in S_1.)
>
> Use C_{i_1} and S_1 to obtain σ_1. Use σ_1 and β_1 to obtain π'_1. P_1 consists of all of the other variables in blocks touched by S_1; π''_1 is $\rho'|_{P_1}$; $\pi_1 = \pi'_1\pi''_1$.
>
> Let C_{i_2} be the first clause of F that is not made false by $\rho\pi_1\sigma_2\pi''_2 \dots \sigma_k\pi''_k$. (Note that this can be obtained from ρ' (without knowing what ρ is) by changing

the setting of the variables in S_1 and P_1 to π_1. As above, compute σ_2 and π_2, and then use $\rho\pi_1\pi_2\sigma_3\pi_3'' \ldots \sigma_k\pi_k''$ to find C_{i_3}, and continue in this way, to obtain $\pi_1, \ldots, \pi_k$.

Obtain ρ by defining $\rho(x_i) = *$ if $x_i \in \mathrm{Dom}(\pi_1 \ldots \pi_k)$, and $\rho(x_i) = \rho'(x_i)$ otherwise.

∎