An Isomorphism Theorem for Circuit Complexity*

Manindra Agrawal†
Institut für Informatik
Universität Ulm
Oberer Eselsberg
D 89069 Ulm
Germany
manindra@informatik.uni-ulm.de

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

Abstract

We show that all sets complete for NC^1 under AC^0 reductions are isomorphic under AC^0-computable isomorphisms.

Although our proof does not generalize directly to other complexity classes, we do show that, for all complexity classes $\mathcal{C}$ closed under NC^1-computable many-one reductions, the sets complete for $\mathcal{C}$ under NC^0 reductions are all isomorphic under AC^0-computable isomorphisms. Our result showing that the complete degree for NC^1 collapses to an isomorphism type follows from a theorem showing that in NC^1, the complete degrees for AC^0 and NC^0 reducibility coincide.

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 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. Although the isomorphism conjecture was originally stated for the NP-complete sets, subsequent work has considered whether the complete sets for other complexity classes $\mathcal{C}$ collapse to an isomorphism type.

This 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 are functions that are easy to compute but hard to invert. Beginning with [JY85] (see also [KMR89, Se92, KLD86], among others) many authors have noticed that if one-way functions exist, then the Berman-Hartmanis conjecture seems unlikely. In particular, if f is one-way, nobody has presented a general technique for constructing a p-isomorphism between SAT and f(SAT). (Rogers [Ro95] does show how to construct such isomorphisms relative to an oracle. However, the focus of our work is on non-relativized classes.)

It is important to observe that **there are one-way functions in $\mathbf{NC}^0$** if there are any one-way functions at all. Thus, although there has been an intuition that one-way functions cause the isomorphism conjecture to fail, this intuition is incorrect if the one-way functions are sufficiently easy to compute.

One can also state and investigate versions of the isomorphism conjecture for more restrictive reducibilities. In fact, there has been previous work presenting classes of reductions $\mathcal{R}$ such that the $\mathcal{R}$-complete sets in natural (unrelativized) complexity classes are all isomorphic (see e.g. [Ag94]). However, this work has all relied on the fact that the reductions in $\mathcal{R}$ can all 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

*A version of this paper will appear in the Proceedings of the Eleventh Annual IEEE Conference on Computational Complexity (formerly Structure in Complexity Theory), 1996.

†On leave from SPIC Science Foundation, Madras, India. Research partially supported by the Alexander von Humboldt Fellowship. New address after August 15, 1996: Dept. of CSE, IIT Kanpur 208016, India.

‡Supported in part by NSF grant CCR-9509603.

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, the full version of this paper will show that the sets that are complete under the reducibilities considered in [Ag94] are in fact complete under logspace-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] for the particular case of $\mathcal{C} = \mathrm{NC}^1$, except for the question of *uniformity*.

We do not know if our result holds for Dlogtime-uniform AC^0 isomorphisms (also known as first-order isomorphisms). Note that, since first-order projections are a very restricted sort of NC^0 reduction, our result showing that the sets complete under NC^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. Although we suspect that the problem of uniformity can be overcome, we believe that this will require significant additional effort.

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

Section 3 presents our main results about sets complete under NC^0 reductions.

Section 4 Presents the isomorphism theorem for NC^1, and presents some concluding remarks.

An appendix contains a proof of a technical lemma concerning constant-depth circuits, using established techniques [FSS84, Aj83, Hå87, Fo95, B95].

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 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] 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 (NC^0) is the class of functions computed in this way by circuit families of size $n^{O(1)}$ and depth $O(\log n)$ ($O(1)$), 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 Corollary 8 shows that any set hard for NC^1 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 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.)

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

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

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.*

Note that every length-increasing 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 $n < |y|$ 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 class $\mathcal{C}$ closed under Dlogtime-uniform NC^1 reductions, 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 three stages.

Stage 1: In the first stage, we show that A is also hard under NC^0 reductions that output a string whose length is at least linear in the length of the input.

Take any set B in $\mathcal{C}$. Define a set D as:

$$D = \{1x \mid x \in B\} \cup \\ \{0x \mid x \text{ has an even number of ones}\}.$$

Set D is clearly in $\mathcal{C}$ (since D NC^1-reduces to B and $\mathcal{C}$ is closed under NC^1 reductions) and the set B reduces to D via the reduction $x \mapsto 1x$. Consider the NC^0 reduction of D to A, given by the sequence of circuits $\{C_n\}$ for $n > 0$ (C_n has n input bits and some number of output bits). Take the circuit C_n. Its output must depend on every input bit except possibly for the first one. (If not, then we set the first input bit to 0 and then flipping the bit on which the circuit does not depend would make no difference to the output. This is not possible as the output must be different when the parity of the number of ones in the input is different.) It follows that there must be $\Omega(n)$ output bits as each output bit can depend only on a constant number of input bits. A composition of the reductions from B to D (namely $x \mapsto 1x$) and from D to A gives us a P-uniform NC^0 reduction of B to A with the same property. (We observe that this reduction satisfies a stronger property: even if we fix a constant number of bits of the input, there are $\Omega(n)$ output bits that depend on the rest of the (unfixed) input bits. This property will be made use of in the next stage.) This completes Stage 1.

Stage 2: We now show that A is also hard under length-increasing superprojections.

Again take any set B in $\mathcal{C}$. Again we define a new set C, NC^1-reducible to B, which is accepted by the following procedure:

> On input y, let $y = 1^k 0 z$. If k does not divide $|z|$ then reject. Break z into blocks of k consecutive bits each. Let these be $u_1 u_2 u_3 \dots u_p$. We say that a u_i, $1 \le i \le p$, is *null* if the number of zeroes in it is more than $k/2$; it is *zero* if the number of ones in it are between $k/2$ and $3k/4 - 1$; otherwise it is *one*. Form a string x out of these blocks as follows. Start with $x = \epsilon$. For each i, $1 \le i \le p$, do the following: if u_i is null then do nothing; if it is zero then append a zero to x otherwise append a one to x. Now accept iff $x \in B$.

It is straightforward to verify that C is NC^1-reducible to B, and thus $C \in \mathcal{C}$ by the closure properties of $\mathcal{C}$. Hence, by the result of Stage 1 there is an NC^0 reduction of C to A, given by the family of circuits $\{D_n\}$, where the length of the output of any circuit D_n is at least linear in the length of the input. 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. 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 string $1^{4c} 0 u_1 u_2 \dots u_m$ 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 $4c + 1$ input bits to the value $1^{4c}0$. Consider the reduced circuit with $4cm$ remaining input bits. Each output bit of this circuit depends on at most 2^d input bits. Let r be the number of output bits that depend on at least one input bit, and recall that for some $\delta > 0$

$$r > \delta m. \tag{1}$$

Let O be a set that initially contains the above r output bits of the circuit.

The remaining input bits are denoted as usual with $x_1, \dots, x_{4cm}$. 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 $\le 2^d$ 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 $d' \le 2^d$. The "truth table" for this output bit has variables $v_1, \dots, v_{d'}$. 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, such that no other truth table is associated with more output bits than α is.

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,

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

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 $S_i = \{(j, x_{i,j}) : 1 \leq j \leq d'\}$. Clearly all of the sets in $\mathcal{F}$ have size at most c. 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

$$t \geq (s/(c!)^2)^{(1/c)} - 1. \qquad (3)$$

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 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 $\leq 4c^2$ 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 this paragraph 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 $(4c^3 + 1)|x|$. This gives us a bound on m: $m \leq r/\delta$ (by (1)) $\leq c \cdot s/\delta$ (by (2)) $\leq c \cdot (t+1)^c \cdot c!^2/\delta$ (by (3)), and thus if we pick t to be $(4c^3 + 1)|x|$, it follows that it is sufficient to choose m to be $c'' \cdot |x|^c$

for some constant c'' depending on c and δ.

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 3: It is clear at this point that the reduction of B to A described above is length-increasing 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 NC^1-reducible to B. The definition of E is straightforward:

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

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: $|x| < |f(x)| \leq p(|x|)$ for some polynomial $p \geq n^2$. Define a function r as follows: $r(0) = 1$, and $r(t) = p(r(t-1)) + 1$. And now define a reduction g of B to E as: $g(x) = x10^k$ where k is the smallest number such that $|x| + 1 + k = r(t)$ for some t. Function g can clearly be computed by a projection circuit, and so $f \circ g$ is an NC^0 reduction of 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 we assumed $p(n) \geq n^2$, the resulting superprojection is at least length-squaring.

Checking P-uniformity of this step is trivial. ∎

The following corollary (the nonuniform case) is a trivial consequence of the foregoing.

Corollary 3 *For every class $\mathcal{C}$ closed under Dlogtime-uniform NC^1 reductions, every set complete for $\mathcal{C}$ under NC^0 reductions is complete under one-one, length-squaring superprojections.*

Corollary 4 *For every class $\mathcal{C}$ closed under Dlogtime-uniform NC^1 reductions, all sets complete for $\mathcal{C}$ under P-uniform NC^0 reductions are P-uniform AC^0-isomorphic.*

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. ∎

Corollary 5 *For every class $\mathcal{C}$ closed under Dlogtime-uniform NC^1 reductions, all sets complete for $\mathcal{C}$ under NC^0 reductions are AC^0-isomorphic.*

4 Conclusions

In a very recent paper by Arora [Ar95] there is a statement of a lemma showing that AC^0 and NC^0 reductions are quite closely related. Arora presents a number of interesting observations about the limitations of what can be proved to be hard for NP under AC^0 reductions. Although the following lemma is stated explicitly in [Ar95], it has been known since [FSS84, Aj83].

Lemma 6 *For any AC^0 reduction computed by a family of circuits $\{C_n\}$ there exists (1) a constant $\beta > 0$, (2) a partial assignment ρ assigning values in $\{0,1\}$ to $n - n^\beta$ of the n input variables of C_n, and (3) a constant c such that, for any output gate g of C_n, after the variables have been set according to ρ,* *the value of g depends on at most c of the n^β unset variables.*

The proof of this lemma follows along the lines in [BS90] (or also see [Fo95] or [B95] for alternative formulations); applying a suitably-chosen random restriction to an AC^0 circuit yields an NC^0 circuit.

We need a very slight strengthening of this lemma, that also follows easily by a slight modification of the original proof. The essential idea is to consider restrictions over a *larger* alphabet than $\{0,1\}$. For a number b, define a *partial b-block assignment* to be 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.) Then we obtain:

Lemma 7 *For any AC^0 reduction computed by a family of circuits $\{C_n\}$, and for any constant b, there exists (1) a constant $\beta > 0$, (2) a partial b-block assignment ρ assigning values in $\{0,1\}$ to $n-n^\beta$ of the n input variables of C_n, and (3) a constant c such that, for any output gate g of C_n, after the variables have been set according to ρ, the value of g depends on at most c of the n^β unset variables.*

A proof this lemma is included in the appendix.

Corollary 8 *All sets hard for NC^1 under (nonuniform) AC^0 reductions are hard under (nonuniform) NC^0 reductions.*

Proof. Let S_5 be the group of permutations on five elements, and let W_5 be the word problem on S_5 (i.e., the set of all sequences of permutations that when composed evaluate to the identity); this problem is known to be complete for NC^1 under projections [IL95]. Let A be any set hard for NC^1, and let $\{D_n\}$ be the circuit family reducing S_5 to A. We will show that there is an NC^0 reduction reducing W_5 to A.

Note that any permutation on S_5 can be encoded using 7 bits. By Lemma 7, there is a partial 21-block assignment ρ assigning values to at most $n-n^\beta$ of the n input variables of D_n, such that the output bits depend on at most a constant number of the remaining n^β unset input bits. For each of the $O(\log n)$ output bits that are used to encode the length of the output of the circuit (using the convention described in Section 2), set all of the blocks containing input bits on which they depend to zero. Note that each unset block of input variables is long enough to encode three permutations. Thus it is possible to leave the middle seven bits in each such block unset, and set remaining fields in each block either to the identity permutation or to the permutation that computes the inverse of the permutation τ that results by composing the permutations encoded in the consecutive set fields to the right or the left. The result is a new assignment ρ' with the property that (1) ρ' still has at least $1/3 \cdot n^\beta - O(\log n) \geq n^\delta$ unset variables (2) all consecutive blocks of set variables encode permutations that, when composed, yield the identity map (3) the output bits of D_n depend on at most a constant number of unset input bits (4) the $O(\log n)$ output bits that encode the length are all fixed to some value r, and thus an NC^0 circuit results by taking subcircuits for the first r output bits.

Let h be the projection that takes an input y of length n^δ and outputs the result of setting some of the n bits of output according to restriction ρ', and plugs the values of y into the remaining bits. Then y evaluates to the identity iff $h(y)$ does, and thus h reduces W_5 to itself.

Thus for any set B in NC^1, there's a projection f reducing B to W_5, and thus $x \in B$ iff $f(x) \in W_5$ iff $h(f(x)) \in W_5$ (where h is the function defined above) iff $D_n(h(f(x))) \in A$. On inputs in the range h, D_n is an NC^0 reduction, and thus the composition of these functions is an NC^0 reduction. This establishes that all sets hard for NC^1 under AC^0 reductions are hard under NC^0 reductions; isomorphism now follows by our main result. ∎

Corollary 9 *All sets complete for NC^1 under (nonuniform) AC^0 reductions are AC^0-isomorphic.*

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 polytime reductions is one of the main considerations leading many researchers to conjecture that the Berman-

Hartmanis conjecture is false [JY85].) Finally, we use our main result about NC^0-reducibility to show that a true analog of the Berman-Hartmanis conjecture does hold for NC^1. (That is, the sets complete under AC^0 reductions are all AC^0-isomorphic.)

An obvious open question is whether similar collapses hold for other complexity classes, such as NP. It would also be interesting to see if the constructions presented here can be made uniform. We especially call attention to the following problems:

1. Are there other classes $\mathcal{C}$ such that the classes of sets hard for $\mathcal{C}$ under AC^0 and NC^0 reductions coincide?
2. 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 NC^0 reductions, differ?

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 (or NC^0) many-one reductions.

Acknowledgments

We acknowledge helpful conversations with M. Ajtai, S. Arora, R. Boppana, and M. Ogihara.

References

[Ag94] M. Agrawal, *On the isomorphism problem for weak reducibilities*, in Proc. 9th Structure in Complexity Theory Conference (1994) pp. 338–355.

[Ag95] M. Agrawal, *$DSPACE(n) \overset{?}{=} NSPACE(n)$: A degree theoretic characterization*, in Proc. 10th Structure in Complexity Theory Conference (1995) pp. 315–323.

[Aj83] M. Ajtai, *Σ_1^1 formulae on finite structures*, Annals of Pure and Applied Logic 24, 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*, to appear in SIAM Journal on Computing. A preliminary version appeared in Proc. 10th Symposium on Theoretical Aspects of Computer Science, 1993, Lecture Notes in Computer Science 665, pp. 163–174.

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

[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.

[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 a la Kolmogorov*, manuscript.

[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.

[IL95] Neil Immerman and Susan Landau, *The Complexity of Iterated Multiplication,* Information and Computation **116** (1995), 103-116.

[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 nonpolynomial and noncomplete sets in NP*, Theoretical Computer Science **39** (1985) 225–237.

[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.

[KMR89] S. Kurtz, S. mahaney, and J. Royer, *The isomorphism conjecture fails relative to a random oracle*, Proc. 21st ACM Symposium on Theory of Computing, 1989, pp. 157–166.

[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.

[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.

[Ro95] J. Rogers, *The isomorphism conjecture holds and one-way functions exist relative to an oracle*, in Proc. 10th Structure in Complexity Theory Conference (1995) pp. 90–101.

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

5 Appendix

In this appendix, we give a proof of Lemma 7. 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.

Recall the statement of the lemma.

Lemma 7 *For any AC^0 reduction computed by a family of circuits $\{C_n\}$, and for any constant b, there exists (1) a constant $\beta > 0$, (2) a partial b-block assignment ρ assigning values in $\{0,1\}$ to $n - n^\beta$ of the n input variables of C_n, and (3) a constant c such that, for any output gate g of C_n, after the variables have been set according to ρ, the value of g depends on at most c of the n^β unset variables.*

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

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

In the settings where we make use of Lemma 7, n will always be a multiple of b. To simplify notation, we will assume throughout the proof that n is a multiple of b. Let $m = n/b$.

Let $R^\ell_{n,b}$ denote the set of all partial b-block assignments 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. We will frequently use the word "restrictions" as a synonym for "partial b-block assignments" to refer to the elements of R^ℓ.

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

Thus there is a constant c_1 such that, for any $\rho \in R^\ell_{n,b}$,
$$K(\rho | n, \ell, b) \le c_1 + n - \ell b + \log \binom{m}{\ell}$$

(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 $s > 0$,
$$|R^\ell|/|R^{\ell-s}| = 2^{-sb} \cdot \frac{\binom{m}{\ell}}{\binom{m}{\ell-s}} \ge \left(\frac{m-\ell}{2^b \ell}\right)^s$$

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 the unique 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 10 *(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 $|T(F|_\rho)| < sb$.

First, let us see why the Switching Lemma gives a proof of Lemma 7. This is accomplished by the following two claims.

Claim 11 *Let C be a leveled circuit on n inputs with depth $k \geq 3$ and with bottom fan-in $\leq r = r(n,k) = \frac{n^{1/4(k-1)b}}{b^{k-2}}$ and having no more than 2^r gates at levels 2 and above. Let $b \geq 2$. Then for all large n, there is a partial b-block assignment ρ leaving $n^{1/(k-1)}$ blocks unset, such that there is a depth-two circuit (either with each output being an OR of ANDs or with each output being an AND of ORs) with bottom fan-in $\leq n^{1/4(k-1)b}$ computing $C|_\rho$.*

Proof. (of Claim 11) We prove the claim by induction on k. We first establish a few facts, and then proceed with the basis and inductive steps.

Let $s = r$ and let $\ell = n^{(k-2)/(k-1)}$. Pick $\rho \in R^\ell_{n,b}$ such that $K(\rho|C, n, \ell, s, b)$ is maximized. (Thus $K(\rho|C, n, \ell, s, b) \geq n - \ell b + \log \binom{n/b}{\ell}$.)

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

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

Thus it is sufficient to show that

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

This is equivalent to showing that

$$\frac{n}{b}-\ell \geq 2^d\ell(16s)^b$$

where $d = 1+(c_4+c_5)/s$. For all large n, since $s = r(n,k)$, note that $d < 2$. Since there is some $\gamma > 0$ such that $\ell s^b \leq n^{1-\gamma}$, it is thus sufficient to show that $n/b - \ell \geq 2^{3b+2}n^{1-\gamma}$, 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 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.

When $k = 3$, this establishes the basis.

For the inductive step, note that we can apply DeMorgan's laws to the circuit $C|_\rho$ and obtain a circuit D computing $\neg C|_\rho$ that satisfies the inductive hypothesis. Thus there is a ρ' leaving $\ell^{1/(k-2)} = n^{1/(k-1)}$ b-blocks unset, such that $D|_{\rho'}$ is computed by a depth two circuit with bottom fan-in $\leq \ell^{1/4(k-2)b} = n^{1/4(k-1)b}$. Letting ρ'' be the negation of ρ', it is now clear that $C|_{\rho\rho''}$ is the restriction of C computed by a depth two circuit as required by the claim. ∎ (of Claim 11)

Claim 12 *Let n be sufficiently large. Let f be a function on n inputs such that $|f(x)| \leq n^a$ for all x of length n, where f is computed by a depth-two circuit C having bottom fan-in $\leq r = r(n) = n^{1/4b}$. Then there is a restriction ρ leaving $n^{1/4}$ b-blocks unset, such that each output bit of $C|_\rho$ has a decision tree of height at most sb, where $s = 4a$.*

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

Pick $\ell = n^{1/4}$, and let ρ be an element of $\rho \in R^\ell_{n,b}$ such that $K(\rho|C,n,\ell,s,b)$ is maximized.

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^a output gates in C, it is easy to see that $K(\rho|C,n,\ell,s,b) \leq K(\rho|F,n,\ell,s,b) + c_5 + a\log n$. Thus $K(\rho|F,n,\ell,s,b) \geq K(\rho|C,n,\ell,s,b) - c_5 - a\log n \geq n - \ell b + \log\binom{n/b}{\ell} - c_5 - a\log n$. This value is bounded below by $c_4 + n - \ell b + sb\log(16r) + \log\binom{n/b}{\ell-s}$ if and only if

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

Thus it is sufficient to show that

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

By our choice of s this is equivalent to showing that

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

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

Now the claim follows immediately from the Switching Lemma. ∎ (of Claim 12)

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 $C_{1_1}|_\rho$ 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 $\rho\pi_1 \ldots \pi_{h-1}$, and $\rho\pi_1 \ldots \pi_{h-1}\sigma_h$ 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 $\pi_1 \ldots \pi_{k-1}\sigma_k$ 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 $k \leq s$.) By setting ρ' equal to $\rho\sigma_1\pi''_1\sigma_2\pi''_2 \ldots \sigma_k\pi''_k$, 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 $\beta_1, \ldots, \beta_k$, where each β_h describes how σ_h and π'_h differ. Each β_h is a string of length r 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 $\beta = \beta_1 \ldots \beta_k$; thus β is a string of length $kr \leq sr$. 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, $K(\beta'|s, b, r) \leq c_2 + sb(3 + \log r)$. (This is because β' is of the form $*^{j_1 - 1}b_1 *^{j_2-1} b_2 \ldots *^{j_v - 1} b_v ** \ldots *$ where each $b_h \in \{0, 1\}$, and $v \leq bs$ and each $j_h \leq 2r$ (since no β_h is all *'s. Note that many of the numbers j_h may be equal to zero. By making $j_{v+1} = \ldots j_{bs} = 0$ and making $b_v = \ldots = b_{bs}$ 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,

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

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.)

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 $\rho\pi_1\sigma_2\pi''_2\ldots\sigma_k\pi''_k$. (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.

∎