Towards Uniform AC^0-Isomorphisms

Manindra Agrawal
Department of Computer Science
IIT Kanpur
Kanpur 208016, India
manindra@iitk.ac.in

Abstract

For any class $\mathcal{C}$ closed under NC^1 *reductions, it is shown that all sets complete for $\mathcal{C}$ under logspace-uniform AC^0 reductions are isomorphic under logspace-uniform AC^0-computable isomorphisms.*

1 Introduction

One of the long-standing conjecture about the structure of complete sets is the isomorphism conjecture (proposed in [9]) stating that all sets complete for NP under polynomial-time reductions are polynomial time isomorphic. Recently this conjecture was shown to be true in a restricted setting: all sets complete for NP under (non-uniform) AC^0 reductions are (non-uniform) AC^0 isomorphic [2]. As all the well-known NP-complete problems are also complete under AC^0 reductions (although there do exist NP-complete problems that are not complete under AC^0 reductions [1]), this result proves that a very natural restatement of the isomorphism conjecture is true. However, the result is not completely satisfactory in one sense: the isomorphism is computable by *non-uniform* reductions even if the original reductions are uniform. Although this drawback was removed to an extent in [1] who showed that the same result holds when we replace non-uniform AC^0 reductions with *p-uniform* AC^0 reductions, there is still a substantial gap: starting with *Dlogtime-uniform* (widely accepted as the "right" notion of uniformity for AC^0 circuits because of equivalences with first-order and alternation TM characterizations) AC^0 reductions, one gets p-uniform AC^0 isomorphisms. Ideally, one would like to have the same uniformity condition to hold for the isomorphisms as for the reductions.

In this paper, we close this gap, though not completely. We construct AC^0 isomorphisms that are *logspace-uniform* (even NC^1-uniform) instead of p-uniform. Our approach also suggests possible ways to close the gap completely.

The next section provides an outline of our proof. Section 3 contains definitions, and the subsequent sections are devoted to proving the result. We end with discussing possibilities for improvement.

2 Proof Outline

The isomorphism result of [2] is proved in following three steps:

Gap Theorem: This shows that all complete sets under AC^0 reductions (under any notion of uniformity) are also complete under *non-uniform* NC^0 reductions. This step is non-uniform.

Superprojection Construction: This proves that all complete sets under NC^0 reductions are also complete under a special kind of reductions called *superprojections*. This step is is p-uniform, i.e., if one starts with uniform NC^0 reductions then the resulting superprojections are p-uniform.

Isomorphism Construction: This proves that all complete sets under superprojections are isomorphic under AC^0 isomorphisms. This step is Dlogtime-uniform: starting with Dlogtime-uniform superprojections, one gets Dlogtime-uniform AC^0 isomorphisms.

The proof of Gap Theorem uses the Switching Lemma of [12] in the construction of NC^0 reductions and is the reason for its non-uniformity. In [1], using the method of conditional probabilities, the Switching Lemma of [12] was derandomized to make the construction p-uniform. The Superprojection Construction of [2] uses the Sunflower Lemma of [11]. Clearly, the uniformity of both these steps needs to be improved to obtain better results. It is useful to note here that the Gap Theorem *cannot* be made Dlogtime-uniform [2]. However, for our purposes it is enough if it can be made AC^0-uniform.

We first consider the Gap Theorem. The method of conditional probability used to derandomize the Switching Lemma in [1] appears inherently sequential. One possible way to improve the uniformity is to find a different way of derandomizing the lemma. There does exist a different derandomization of the lemma in the literature [10]: they obtain a *pseudo-random generator* against the Switching Lemma of [13] that stretches a seed of length $(\log n)^{O(d)}$ to n bits and "fools" the lemma for depth d circuits. However, it does not serve our purpose since derandomizing the lemma using this generator would require superpolynomial sized circuits.

We construct a new pseudo-random generator against the Swithcing Lemma of [12]. This generator stretches a seed of length $O(\log n)$ to n bits. We can thus derandomize the lemma by cycling through all the seed values. We show that the generator construction, and other related computations, can be performed by logspace TMs thus making the Gap Theorem logspace-uniform.

Next, we consider the Superprojection Construction. This uses the Sunflower Lemma which again appears inherently sequential. So we need a different construction here! We adopt the approach of Gap Theorem: define a random construction algorithm that suceeds with high probability and then derandomize it using appropriate pseudo-random generators. Following it, we obtain a new, and very simple, (randomized) construction of the superprojections, and then derandomize it. All the computations is this construction can also be performed by logspace TMs.

Combining the above constructions togather with the Isomorphism Construction, we get logspace-uniform AC^0-isomorphisms.

3 Basic Definitions and Preliminaries

We assume familiarity with the basic notions of many-one reducibility as presented, for example, in [7].

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 $1^n \mapsto C_n$ is easy to compute in some sense. In this paper, we will consider various notions of uniformity: Dlogtime-uniformity [8], AC^0-uniformity, NC^1-uniformity, logspace-uniformity, and P-uniformity [3] (in addition to non-uniform circuit families). We will follow the convention that whenever the function $1^n \mapsto C_n$ is computed by a circuit family, the circuit family is assumed to be Dlogtime-uniform. So, for example, NC^1-uniform means that the function can be computed by a Dlogtime-uniform NC^1 family of circuits.

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 $f(x)$. We will adopt the following specific convention for interpreting the output of such a circuit: 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 any other reasonable convention that allows for variable sized output, 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 $\{+, \times, \leq\}$, [15, 8] the logspace-rudimentary reductions of Jones [14, 5], logarithmic-time alternating Turing machines with $O(1)$ alternations [8] 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 [18]. The class of projections is clearly a subclass of NC^0 and has been studied by many authors; consult the references in [4].

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, $g(f(x)) = x$.

4 Logspace-uniform Gap Theorem

In this section, we construct a pseudo-random generator for the Switching Lemma. We then show how to use this generator to construct a logspace-uniform version of Gap Theorem.

4.1 Pseudo-random generator for Switching Lemma

In this section, we show that the Switching Lemma (of [12]) can be derandomized completely by constructing a pseudo-random generator for it that stretches an $O(\log n)$ bit random seed to n bits. For the construction of this generator, we need to go through the proof of the Switching Lemma as in [12] and identify the derandomization points[1]. We will follow a simplification of the original proof of [12]. This proof has been sketched at several places (see, e.g., [1]), we will sketch it once more with the required parameter values.

In the proof of the Switching Lemma, a random restriction is applied to the input bits of the circuit a number of times. Each time an unset input bit is set with certain probability. Such a random restriction can be viewed as consisting of two stages: in the first stage a subset of unset bits is identified, and in the next stage all the unset bits not in the subset are set to 0/1 randomly. With this view, we can divide the random bits needed in each random restriction into two types: (1) random bits needed to choose a subset, and (2) random bits needed to give values to bits not in the subset. We refer to these two types of random bits as *subset bits* and *value bits* respectively.

We model a *random source* as a function $S : \{0,1\}^r \mapsto \{0,1\}^n$, where any x, $|x| = r$ is a *seed* for the source and $S(x)$ is the output sequence of random bits from the source.

In this subsection, we will denote, by $\mathrm{AC}(d, w, n)$ the class of circuits with AND, OR, and NOT gates (AND and OR gates having unbounded fanin) of depth d and width w having n input bits. We now state the lemma in the form that we need:

Lemma 4.1 *There exists a constant $A_0(d,k)$ (depending on d and k only) such that for large enough n and for any circuit C in $\mathrm{AC}(d, n^k, n)$, when a sequence of random restrictions is applied to C with appropriate parameters, C reduces, with probability at least $1 - \frac{1}{n^2}$, to a depth-2 circuit depending on at most $A_0(d,k)$ of at least $n^{1/A_0(d,k)}$ unset bits.*

Proof Sketch. Let $C \in \mathrm{AC}(d, n^k, n)$ be an AC^0 circuit of depth d and width n^k on n input bits. We can assume, without loss of generality, that C is arranged into d alternating levels of AND and ORs of width n^k on $n^{\delta_0} = n$ unset bits with its leaves being depth $c_0 = 1$ decision trees. The proof proceeds in d steps. After step i, the circuit reduces to a depth $d-i$ circuit of width n^k on n^{δ_i} unset bits with leaves being depth c_i decision trees. Step i has c_{i-1} stages. We now describe a single stage of step i.

[1] It is interesting to note that the stronger Switching Lemma of [13] does not admit such a construction.

In the first stage of step i, the bottom layer of the circuit consists of ANDs or ORs of depth c_{i-1} decision trees. Assume it is ANDs of decision trees (proof for ORs is identical). Each subcircuit can be expressed as AND of size c_{i-1} ORs. Denote these by Q_1, Q_2, ..., Q_{n^k} (there will be at most n^k such subcircuits since the width is n^k). For each Q_j, define set Maxset(Q_j) to be the lex-first maximal set of clauses in Q_j that are variable disjoint. If these are more than $f \log n$ (the value of f will be indicated later) then redefine Q_j to be the lex-first $f \log n$ of these clauses. So each Q_j contain at most $c_{i-1} f \log n$ variables.

Now use subset bits to choose a random $n^{\delta_i/2}$ sized subset of unset bits and then use value bits to set the remaining unset bits. A simple calculation (based on Chernoff bounds on tail distribution) shows that the probability that a Q_j has more than c' unset bits is less than $(\frac{e \cdot c_{i-1} \cdot f \cdot \log n}{c' \cdot n^{\delta_i/2}})^{c'}$. Choosing c' appropriately, we can make this probability less than $\frac{1}{n^{k+3}}$ for large enough n assuming that f does not depend on n (which will turn out to be true later). Summing over all Q_js, the probability that any Maxset(Q_j) has more than c' unset bits is less than $\frac{1}{n^3}$ for large enough n.

Consider now those Q_js for which $|\text{Maxset}(Q_j)| = f \log n$. By the above calculation, most of the restrictions will have at most c' unset bits in it. We consider such restrictions only. Drop the (at most) c' ORs that have an unset bit from the set. As the set input variables take random values, the probability that a particular OR in the set will have the value 1 is at most $1 - \frac{1}{2^{c_{i-1}}}$. And since the ORs in the set are disjoint, the probability that *all* of them will have value 1 is at most $(1 - \frac{1}{2^{c_{i-1}}})^{f \log n - c'}$. Choosing the value of f appropriately and independent of n, we can make this less than $\frac{1}{n^{k+3}}$ for large enough n. Summing over all Q_js, the probability that some Q_j with $|\text{Maxset}(Q_j)| = f \log n$ survives the restriction is less than $\frac{1}{n^3}$ for large enough n.

We can now replace every surviving Q_j with a decision tree of depth c' whose leaves are ANDs of ORs of size at most $c_{i-1} - 1$. This finishes stage 1 of step i. Repeating this c_{i-1} times will reduce the circuit to depth $d - i$ circuit of the kind mentioned above with suitable values of c_i and δ_i. Further, this will happen with probability at least $1 - O(\frac{1}{n^3})$ for large enough n. After d steps, the circuit will be simply a depth c_d decision tree thus depending on at most 2^{c_d} unset bits out of at least n^{δ_d} for large enough n. Moreover, this event will occur with probability at least $1 - O(\frac{1}{n^3}) \geq 1 - \frac{1}{n^2}$ for large enough n. Choosing $A_0(d,k) = \max\{2^{c_d}, \frac{1}{\delta_d}\}$ completes the proof. ∎

we now proceed with the derandomization. Notice the following three crucial points about any particular stage of the above proof:

1. In any stage, we have argued about properties of sets of input bits of size at most $c_d f \log n$.

2. The property of subset bits used is that given any subset of size at most $c_d f \log n$ of a set of $m \geq n^{1/A_0(d,k)}$ elements, the probability that a random subset of size $m^{1/2}$ chosen from the subset space intersects the given subset with cardinality more than c' is at most $\frac{1}{n^{k+3}}$.

3. The property of value bits used is that given any AND of disjoint ORs, with AND of fanin at least $f \log n - c_d$ and ORs of fanin at most c_d, the probability that a random assignment to the inputs chosen from the value space makes the AND output a 1 is at most $\frac{1}{n^{k+3}}$.

Therefore, if we take random restrictions from any subset and value spaces satisfying the above conditions, the proof will still go through. Of course, for each stage we need to generate a fresh set of random bits independent from the previous stages.

We can easily derandomize the construction of such spaces, in fact, they already exist in the literature. We now describe these derandomizations.

4.1.1 Derandomizing value bits

This is straightforward: we need to use any $c_d f \log n$-wise independent source. However, such sources have seed size of at least $(\log n)^2$. So, instead, we use a $c_d f \log n$-wise independent $\frac{1}{n^{k+3}}$-biased source [16]. Efficient constructions of such sources are known [16, 6]. We describe one of these (given in [6]).

Let F be the field of 2^m elements. The seeds for the source are all pairs (x, r), $x, r \in F$. Given (x, r), the i^{th} bit of the source is defined as $x^i \cdot r$ where '$\cdot$' is the inner product operation. The source provides up to $2^{3m/4}$ bits and, if t bits are taken from it, they are $\frac{m}{4}$-wise independent with a bias of at most $\frac{t}{2^{3m/4}}$. We refer to this source as $\mathcal{V}(m)$.

4.1.2 Derandomizing subset bits

Here we use *designs* defined in [17]:

Definition 4.2 A (k, ℓ, d)-design is a seqeunce of sets S_1, ..., S_m with $|S_i| = \ell$ and $S_i \subseteq \{1, 2, \ldots, d\}$, such that for every $i \neq j$: $|S_i \cap S_j| \leq k$.

The construction of designs in [17] shows the following:

Lemma 4.3 *[17] For every $c \geq 0$, there exists a $(c, n^{1/2}, n)$-design with at least $n^{c/2}$ sets. Further, any element of any set of this design can be computed in time polynomial in $\log n$ and c.*

Proof Sketch. Let $m = \frac{\log n}{2}$. Let $\bar{a} = (a_0, \ldots, a_c)$ with $a_i \in F_{2^m}$, the field of 2^m elements. For polynomial $P_{\bar{a}}(x) = \sum_{i=0,c} a_i \cdot x^i$ let

$$S_{\bar{a}} = \{(x, P_{\bar{a}}(x)) \mid x \in F_{2^m}\}.$$

It is straightforward to see that two such sets have intersection of size at most c and there are $n^{c/2}$ such sets. Computing an element of any set of this sequence is can clearly be done in time polynomial in $\log n$ and c. ■

The seeds for our source will consist of all choices of $\bar{a}$ for an appropriately chosen value of c. This source will provide $n^{c/2}$ subsets of size $n^{1/2}$ that are precisely the set sequence of the $(c, n^{1/2}, n)$-design above. Call this source $\mathcal{S}(c, n)$.

The following lemma shows that this source is sufficient to derandomize the subset bits.

Lemma 4.4 *Let X be any subset of $\{1, 2, \ldots, n\}$, and $|X| = t \leq c + 1$. Then exactly $n^{(c+1-t)/2}$ sets from the source $\mathcal{S}(c, n)$ contain X.*

Proof. Set X gives rise to a system of t linearly independent equations with $c+1$ unknowns over the field F_{2^m}. This has exactly $n^{(c+1-t)/2}$ solutions. ■

Corollary 4.5 *Any set Y of k bits intersects with at most 2^k sets from $\mathcal{S}(c, n)$ at more than c bits.*

Proof. For any subset of Y of more then c bits, at most one set from the source contains it. Therefore, there are less than 2^k sets of this kind. ■

Corollary 4.6 *In the analyzed stage of the proof of 4.1, choosing subsets from the source $\mathcal{S}(c', n^{\delta_i})$ for appropriately chosen c' suffices.*

4.1.3 The pseudo-random generator

It is now clear how to derandomize the Switching Lemma: the proof of the lemma has a constant number of stages, and each stage uses a random restriction on n^δ unset input bits to leave $n^{\delta/2}$ bits unset. For this stage, we use subset bits from a $\mathcal{S}(c', n^\delta)$ source (for appropriately chosen c') and use value bits from the source $\mathcal{V}(c_d f \log n)$. It is useful to observe here that the size of the seed of the source $\mathcal{S}(c', n^\delta)$ keeps increasing across the stages: from one stage to next, the number of unset bits gets square rooted, but the number of bits in the set Maxset(Q_j) increases (as it depends on the constant c_i) and so c' has to be a much larger constant than the one in previous stage.

So, given any circuit from AC(d, n^k, n), a pseudo-random generator against the Switching Lemma for these circuits is obtained by a hybrid source $\mathcal{H}_0$ that uses $A(d, k)$ pairs of sources—one for each stage—with i^{th} pair being $(\mathcal{V}(A(d, k) \cdot \log n), \mathcal{S}(B(d, i), n^{1/2^{i-1}}))$ for an appropriately chosen sequence of constants $B(d, 1)$, ...,

$B(d, A(d,k))$. For the sake of simplicity, we will henceforth refer to the pair used for i^{th} stage as $(\mathcal{V}_i, \mathcal{S}_i)$.

Given a seed $((v_1, s_1), \ldots, (v_{A(d,k)}, s_{A(d,k)}))$ of the hybrid source $\mathcal{H}_0$, the j^{th} bit of the source is calculated as follows:

> Test if the subset s_1 contains number $j = (j', p)$. If not, then output the j^{th} bit of the value specified by v_1. Otherwise, repeat the algorithm for s_2 with $j = j'$ and so on. If none of the subsets set the j^{th} bit then it remains unset.

By the arguments above, the derandomization of the Switching Lemma follows:

Lemma 4.7 *There exists a constant $A(d,k) \geq 2$ (depending on d and k only) such that for large enough n, and for any circuit C in $\mathrm{AC}(d, n^k, n)$, when the input to C is set using the restriction output by the hybrid source $\mathcal{H}_0$, C reduces, with probability at least $1 - \frac{1}{n^2}$, to a depth-2 circuit depending on at most $A(d,k)$ of $n^{1/2^{A(d,k)}}$ unset bits.*

An interesting feature of using this source for obtaining random restrictions is that all the restrictions set all except one bit in every successive block of $n^{1-\frac{1}{2A(d,k)}}$ bits—this follows since sources $\mathcal{S}(B(d,i), n^{1/2^{i-1}})$ leave exactly one unset bit in successive blocks of $n^{1/2^i}$ (so far unset) bits. This feature will be very useful in our uniform construction.

4.2 Making Gap Theorem uniform

Theorem 4.8 *Let $\mathcal{C}$ be any class closed under uniform-*NC^1 *reductions. The sets hard for $\mathcal{C}$ under logspace-uniform* AC^0 *reductions are also hard under logspace-uniform* NC^0 *reductions.*

Proof Sketch.

We first outline the proof of the Gap Theorem in [2] and then mention the changes to be made. Given a set A that is hard for the class $\mathcal{C}$ under AC^0 reductions and an arbitrary set B in $\mathcal{C}$, first an intermediate set B' is constructed that is a simple encoding of B in the following way: corresponding to every $x \in B$, B' contains several strings; string x can be obtained by splitting any such string into equal sized blocks, computing number of ones modulo 3 for each block, ignoring a block if this value is 2, and concatenating the (one bit) values obtained for remaining blocks. The set B' reduces to B via a Dlogtime-uniform NC^1 reduction, and so belongs to $\mathcal{C}$.

Now fix an AC^0 reduction of B' to A given by circuit family $\{C_n\}$. According to the Switching Lemma, when a random reduction on circuit C_n is applied, it reduces to an NC^0 circuit with high probability. Fix a restriction that achieves this, as well as leaves at least three bits unset in each block of input (input is an instance of B' and it can be shown that a random restriction will leave at least three unset bits in each block with high probability provided the size of block is large enough). Call such a restriction a good restriction. Set all the input bits that feed into that part of the circuit that determines the last $O(\log n)$ bits of the output specifying the output length. Since the circuit has reduced to an NC^0 circuit, there would be only $O(\log n)$ such bits. Set these bits and if needed, $O(\log n)$ additional bits, to ensure that all the blocks in which these bits occur have number of ones equal to two modulo 3.

Now define a reduction of B to B' as follows: map i^{th} bit of input string x to a bit position corresponding to the first unset bit in the i^{th} block that contains unset bits setting the remaining unset bits of the block such that the number of ones in the block modulo 3 equals the value of i^{th} bit of x. The remaining output bits of the reduction are set according to the settings given above. A composition of this reduction with the reduction from B' to A yields an NC^0 reduction from B to A.

In this proof, the uniformity of the final reduction is determined by the effort needed to identify a good restriction. Other steps can be computed by Dlogtime-uniform NC^1 circuits. We use the derandomized Switching Lemma to identify a good restriction using a logspace TM. In fact, it would be easier to explain the construction in terms of circuits, and we show that a Dlogtime-uniform NC^1 circuit can compute the restriction. This circuit has an AC^0 circuit (computing a function) as input and needs to (1) find out a "good" seed for the hybrid source constructed in the previous subsection, and (2) compute the good restriction using this seed. The seed is good if the restriction generated by it succeeds in all the stages as mentioned in the proof of 4.1 as well as leaves at least three bits unset in every block of $n^{1-\epsilon}$ bits for some ϵ. The later property holds for *all* the seeds as noted at the end of previous subsection. So we only need to find a seed that succeeds at all the stages. As different parts of the seed are used in different stages, it is enough if an NC^1 circuit can verify if a given value of a particular segment of the seed is good for a particular stage and then output the (reduced) circuit after the application of the generated restriction. Since there are constant number of stages and only polynomially many possible values of the seed, the overall circuit would still be an NC^1 circuit.

Let us now concentrate on stage 1 of step i of the proof of 4.1. The bottom layer of the circuit given is an AND of fanin c_{i-1} ORs. A good restriction will allow to tranform this layer to constant depth decision trees whose leaves are ANDs of fanin $c_{i-1} - 1$ ORs. One can test if a given restriction is good by handling each AND gate seperately: guess the constant number of unset bits that should go in the decision tree, go through all possible settings of these bits and see if these settings combined with the bits set by the re-

striction either set the AND gate output to 0 or reduce fanin of each feeding OR by at least one. If yes, then we get a transformation of this AND gate for the next stage as well. The Lemma 4.7 guarantees that there will always exist such a good restriction. This verification (and the transformation of the circuit for the next stage) can be easily done by even a Dlogtime-uniform AC^0 circuit. The place where we need an NC^1 circuit is in computing the restriction from the seed segment. While the subset bits can be computed by a Dlogtime-uniform AC^0 circuit (these need constant number of multiplications in a field of size poly(n)), the value bits require computation of x^i in a field of size poly(n) for i up to n. This requires a Dlogtime-uniform NC^1 circuit (although it may be possible to improve on this—see the concluding section). ■

5 Logspace-uniform Superprojection Construction

We start with the definition of a superprojection [2].

Definition 5.1 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.

In [2], for any given NC^0 circuit on n inputs, a restriction was constructed that left at least n^ϵ bits unset (for some $\epsilon > 0$) and the resulting circuit became a superprojection. This construction is, however, p-uniform as it uses the Sunflower Lemma of [11]. In this section, we give a different construction for the required restriction. The construction is a randomized one that can be derandomized easily to yield better uniformity conditions.

Let C be an NC^0 circuit with n input bits, m output bits, and of size n^k. C can be easily transformed (by a Dlogtime-uniform AC^0 circuit) into a circuit such that:

- each output bit of the circuit is an OR of ANDs of input bits or their negations,
- the fanin of every gate of bounded by some constant c, and
- for every output bit and for every input bit that feeds into the circuit computing the output bit, there is some setting of remaining input bits such that the output bit value depends on the input bit value.

Now define a random restriction ρ of C as follows: for each input bit, leave it unset with probability $\frac{1}{2}$, set it to one or zero with probability $\frac{1}{4}$ each. Consider an input bit of C. Say that it is *good* under the restriction ρ if it remains unset and at least one of the output bits of C now depends only on this bit. An unset input bit will be good if for some output bit whose circuit takes it as input, all the other feeding input bits are set to values such that the output bit depends only on this bit. By the above transformation, we have ensured that an input bit that influences at least one output bit will be good with probability at least $\frac{1}{2} \cdot \frac{1}{4}^{c-1} \geq \frac{1}{4^c}$.

Since we can ensure (see later) that a large fraction of input bits of circuit C influence some output bit, we have that the expected number of good bits in a random restriction are large. Non-uniformly fixing a restriction that has at least these many good bits and also fixing all unset input bits that are not good to arbitrary values, we get a restriction such that the resulting circuit becomes a superprojection. This will allow us to get a non-uniform version of the Superprojection Construction. It can be made uniform provided we can derandomize this construction. And derandomizing this is easy! The only place we have used independence of randomly assigned values is in deriving a lower bound on the probability that a given input bit is good. The argument there actually needs only c-wise independence. So if we use restrictions whose values are c-wise independent, we can still find a restriction that given the Superprojection Construction. And to get c-wise independent restriction values, it is enough to sample from a $2c$-wise independent distribution. Such distributions with polynomial sized sample spaces are well known (see for example []). We can now prove the logspace-uniform version of the Superprojection Construction:

Theorem 5.2 *For every class $\mathcal{C}$ closed under Dlogtime-uniform NC^1 reductions, every set hard for $\mathcal{C}$ under logspace-uniform NC^0 reductions is hard under logspace-uniform one-one, length-increasing superprojections.*

Proof. Again we start by providing a brief sketch of the construction as given in [2] and then mention the modifications. Let A be hard for $\mathcal{C}$ under NC^0 reductions. Take any set B in $\mathcal{C}$ and define a new set C accepted by the following procedure (the definition of this set is slightly different from the one used in [2]):

> 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 2 modulo 3; as *zero* if the number of ones in it is 0 modulo 3; 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 $x \in B$.

This definition is almost same as the one of set B' in the previous proof. The difference being in the behaviour when

a block is all ones or all zeroes. We need this to ensure that at least one bit in each block of input bits influences some output bit.

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 the construction of [2], Sunflower Lemma [11] is used to identify n^ϵ (for some $\epsilon > 0$) good input bits in D_n and then a projection reduction is constructed of B to C that maps input bits to the good bits ensuring that the instance of C thus obtained encodes the input instance of B.

Instead of Sunflower Lemma, we now use the derandomized construction described above. Notice that we need to ensure that the restriction that we construct should not set bits in any block to all zeroes or all ones. We ensure this by making the block size $\Theta(\log n)$ bits long and choosing restrictions from a $O(\log n)$-wise independent, $\frac{1}{n^3}$-biased source instead of a $2c$-wise independent one. This ensures that when a restriction is chosen from this source, then with high probability no block gets all zeroes or ones. It also ensures that every block will have at least three unset bits with high probability. To identify a restriction that satisfies these conditions and contains a large number of good bits, we only need a Dlogtime-uniform NC^1 circuit, and once the restriction is identified, a projection reduction can be constructed that maps bits of instance of B to good bits. While constructing the projection, we need to ensure that (1) none of the blocks get all zeroes or all ones, (2) when an input bit is mapped to a good bit in a block, all the other bits in the block are set such that the number of ones equals zero modulo 3, and (3) all the input bits can be mapped in this way to good bits.

(3) is ensured automatically since there are $\Omega(n)$ good bits and therefore $\Omega(\frac{n}{\log n})$ blocks containing good bits. (1) is ensured by the choice of restriction, and (2) is ensured by setting the remaining unset bits (there will be at least two) so that the number of ones is zero modulo 3 (as in the previous proof). It is easy to see that all this can be done by a Dlogtime-uniform NC^1 circuit. ■

6 Dlogtime-uniform Isomorphisms?

The ultimate goal of this direction of work is to obtain a completely uniform version of the AC^0 Isomorphism Theorem. This now looks possible. We have developed techniques to bring down the uniformity condition of [1] from polynomial-time to Dlogtime-uniform NC^1. If we notice carefully, the steps in our construction that require uniformity weaker than Dlogtime-uniform AC^0 are:

1. To compute a bit from the hybrid source defined in section 4.1.3, we need to compute x^i for $i \leq n$ in a field of size poly(n). The current best way of uniformly computing this is via a NC^1 circuit. However, it may be possible to compute it by a Dlogtime-uniform AC^0 circuit. Another way of getting around this problem is to use a different ϵ-biased $O(\log n)$-independent source. For example, if we use a different source from [6] that is based on distribution of quadratic residues in a finite field, its bits *can* be computed by a Dlogtime-uniform AC^0 circuit.

2. To ensure that the number of ones in a block of $n^{1-\epsilon}$ bits equals zero modulo 3 in the proof of Gap Theorem of section 4.2. Not only does this step require an NC^1 circuit, it *cannot* be done by even a non-uniform AC^0 circuit in general. It seems essential to use blocks of size $n^{1-\epsilon}$ since the Switching Lemma only leaves n^ϵ bits unset. One way to get around this obstacle is to use a derandomized version of Håstad's Switching Lemma [13]: this lemma leaves $n/(\log n)^{O(1)}$ bits unset, and so the block size need be only $(\log n)^{O(1)}$. Addition of these many bits can be easily done by Dlogtime-uniform AC^0 circuits.

3. To ensure that there are a large number of good bits in the chosen restriction in the proof of Theorem 5.2, one needs to count the number of good bits under the restriction. This requires at least a TC^0 circuit. Similarly, during the construction of the projection, the i^{th} input bit needs to be mapped to the i^{th} good bit and to identify the i^{th} good bit we need to count the number of good bits to the 'left' of any given bit, again requiring a TC^0 circuit. It seems that we need to have a different construction in order to improve the uniformity condition here. However, the underlying philosophy should be the same (get a randomized construction and then derandomize) as this seems best suited for obtaining high degree of uniformity.

References

[1] M. Agrawal, E. Allender, R. Impagliazzio, T. Pitassi, and S. Rudich. Reducing the complexity of reductions. In *Proceedings of Annual ACM Symposium on the Theory of Computing*, pages 730–738, 1997.

[2] M. Agrawal, E. Allender, and S. Rudich. Reductions in circuit complexity: An isomorphism theorem and a gap theorem. *J. Comput. Sys. Sci.*, 57:127–143, 1998.

[3] E. Allender. P-uniform circuit complexity. *J. ACM*, 36:912–928, 1989.

[4] E. Allender, J. Balcázar, and N. Immerman. A first-order isomorphism theorem. In *Proceedings of the Symposium on Theoretical Aspects of Computer Science*, 1993.

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

[6] N. Alon, O. Goldreich, J. Hastad, and R. Peralta. Simple constructions of almost k-wise independent random variables. In *Proceedings of Annual IEEE Symposium on Foundations of Computer Science*, pages 544–553, 1990.

[7] J. Balcázar, J. Díaz, and J. Gabarró. *Structural Complexity I*. EATCS Monographs on Theoretical Computer Science. Springer-Verlag, 1988.

[8] D. M. Barrington, N. Immerman, and H. Straubing. On uniformity within NC^1. *J. Comput. Sys. Sci.*, 41:274–306, 1990.

[9] L. Berman and J. Hartmanis. On isomorphism and density of NP and other complete sets. *SIAM Journal on Computing*, 1:305–322, 1977.

[10] J. Cai, D. Sivakumar, and M. Strauss. Constant depth circuits and the Lutz Hypothesis. In *Proceedings of Annual IEEE Symposium on Foundations of Computer Science*, pages 595–604, 1997.

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

[12] M. Furst, J. Saxe, and M. Sipser. Parity, circuits, and the polynomial hierarchy. *Mathematical Systems Theory*, 17:13–27, 1984.

[13] J. Hastad. *Computational limitations on small depth circuits*. PhD thesis, Massachusetts Institute of Technology, 1986.

[14] N. Jones. Space-bounded reducibility among combinatorial problems. *J. Comput. Sys. Sci.*, 11:68–85, 1975.

[15] S. Lindell. A purely logical characterization of circuit complexity. In *Proceedings of the Structure in Complexity Theory Conference*, pages 185–192, 1992.

[16] J. Naor and M. Naor. Small-bias probability spaces: Efficient constructions and applications. In *Proceedings of Annual ACM Symposium on the Theory of Computing*, pages 213–223, 1990.

[17] N. Nisan and A. Wigderson. Hardness vs. randomness. *J. Comput. Sys. Sci.*, 49(2):149–167, 1994.

[18] J. H. stad. One-way permutations in NC^0. *Information Processing Letters*, 26:153–155, 1987.