Collapsing Degrees via Strong Computation

Lane A. Hemachandra[*]
University of Rochester
Department of Computer Science
Rochester, NY 14627 USA

Albrecht Hoene[†]
Technische Universität Berlin
Fachbereich Informatik
D-1000 Berlin 10 Germany

November 20, 1990

Abstract

Though Berman and others have provided powerful techniques to collapse nondeterministic degrees at and above nondeterministic linear space, and Immerman and Szelepcsényi have provided techniques that collapse even sublinear nondeterministic space classes, it has remained an open problem whether *any* collapses could be proven for sublinear nondeterministic space degrees. This paper provides the first such collapses. For nondeterministic space classes $\mathcal{C}$ above NL, we show that all $\leq_m^{1\text{-}L}$-complete sets for $\mathcal{C}$ collapse to a single $\leq_{1l_i}^{1\text{-}L}$ degree (i.e., all $\leq_m^{1\text{-}L}$-complete sets for $\mathcal{C}$ are $\leq_{1l_i}^{1\text{-}L}$-equivalent), and that all $\leq_m^{1\text{-}NL}$-complete sets for C are NL-isomorphic (and thus P-isomorphic). Our techniques sharply improve previous results for PSPACE.

1 Introduction

Are all NP-complete sets polynomial-time isomorphic—and thus essentially the same set under different naming schemes? Berman and Hartmanis conjectured that all NP-complete sets are indeed P-isomorphic [BH77]. Though relativized [KMR89,Kur83,GJ86, HH], indirect [KLD86], and circumstantial [JY85] evidence against the Berman-Hartmanis conjecture has been gathered, it remains an open question whether all NP-complete sets are P-isomorphic.

[*]Research supported in part by the National Science Foundation under grant CCR-8996198 and a Presidential Young Investigator Award.

[†]Research supported by a Deutsche Forschungsgemeinschaft Postdoktorandenstipendium. Current address: Department of Computer Science and Engineering, FR-35, University of Washington, Seattle, WA 98195.

Indeed, the decade-long research effort devoted to the Berman-Hartmanis Conjecture has yielded scant progress on the isomorphism question; *no* nontrivial class has yet been proven to have a collapsing complete $\leq_m^p$-degree. And although researchers have obtained many relativized results on isomorphism [HS89,KMR89,Kur83,GJ86,HH], very few non-relativized results exist bearing on isomorphism. Most notably, we have:

1. **[KMR87]** There is a set A that is $\leq^L_{2\text{-}truth\text{-}table}$-complete for PSPACE such that any set that is logspace many-one equivalent to A is logspace isomorphic to A.

2. **[All88]** All $\leq_m^{1\text{-}L}$-complete set for PSPACE are P-isomorphic.

Though isomorphism results have proven surprisingly elusive, clear progress has been made on the closely related question of collapsing degrees. In working towards isomorphism, the general scheme outlined a decade ago by Berman and Hartmanis [BH77,Har78b] is to prove length-increasing (for logspace reductions, length-squaring) one-one equivalence for a given class, and then to combine this with invertibility results to achieve isomorphism; thus, collapsing degrees—in particular, achieving one-one length-increasing equivalence—is a first step towards isomorphism (see [GH89] and [KMR90] for more detailed discussions of the motivations of, and obstacles to, this program).

Though the inversion portion of the Berman-Hartmanis program has proven somewhat difficult to fulfill, many exciting equivalence results have been obtained during the last decade. In particular, all $\leq_m^p$-complete sets for deterministic exponential time collapse to a single $\leq_{1l}^p$-degree [Ber77,Wat85,GH89] (that is, they are all one-one, length-increasing, polynomial-time equivalent), all $\leq_m^p$-complete sets for nondeterministic exponential time collapse to a single $\leq_1^p$-degree [GH89], and all $\leq_m^L$-complete sets for PSPACE collapse to a single $\leq_{1l}^p$-degree (basically [Rus86]).

These results all apply to quite large complexity classes, (such as PSPACE and NEXP); however, equivalence and isomorphism results for large complexity classes cannot in general be translated downwards to smaller complexity classes.[1] In this paper, we obtain—for the first time—isomorphism and equivalence results that hold for small complexity classes. These results will be sufficiently powerful to provide new isomorphism and equivalence results for large classes, such as PSPACE.

[1] Note the exception presented as Theorem 2 of [GH89]. Also of great interest is recent work of Buhrman, Homer, and Torenvliet [BHT90]; for the quite important question of distinguishing reducibilities, they achieve results for small space classes.

The central obstacle to obtaining isomorphism results for classes of small complexity is that the crucial diagonalization technique used to obtain the previous equivalences requires that the complexity class used be able to simulated *polynomial* work on a *linear-sized (existential) guess.* In general, classes below NSPACE[n] seem unable to simulate linear-bounded existential quantification. Nonetheless, we show that small complexity classes, in the very special setting needed for the equivalence-creating diagonalization, can create the *effect* of such a simulation via the power inductive counting vests in "strong" computation. Thus, we obtain broad equivalence results for small complexity classes.

Our basic result is that, for any nice nondeterministic space class C above nondeterministic logspace, all $\leq_m^{1\text{-}L}$-complete sets for C collapse to a single $\leq_{1li}^{L}$ degree.[2]

Extending the techniques used to obtain this, we establish that all $\leq_m^{1\text{-}NL}$-complete sets for C collapse to a single $\leq_{1li}^{1\text{-}NL}$-degree. Building on this result we are able show that all $\leq_m^{1\text{-}NL}$-complete sets for C are P-isomorphic and indeed NL-isomorphic.

Our results not only apply to small complexity classes, but in fact also substantially improve previously known results about the structure of PSPACE's complete sets. In particular, we improve Allender's result that all 1-L complete sets for PSPACE are P-isomorphic; we show that all 1-NL complete sets for PSPACE are NL-isomorphic. All our $1li$ equivalence proofs in fact also yield one-one quadratically length-increasing equivalence, and thus satisfy the first requirement of the Berman-Hartmanis scheme for achieving isomorphism.

2 Notation

Our notation follows standard conventions [KMR90]. All sets are assumed to be over some fixed alphabet with at least two characters. EXP denotes $\bigcup_{k>0}$ DTIME[2^{n^k}]. NEXP denotes $\bigcup_{k>0}$ NTIME[2^{n^k}]. We use the symbol $\leq_{properties}^{class}$ to represent reductions. The subscript represents the properties of the reduction, and the superscript denotes the machine type of the reduction. Our subscripts include m representing many-one reductions, 1 representing one-one reductions, $1li$ representing one-one length-increasing reductions, and $1qli$ representing one-one quadratically length-increasing (i.e., $(\forall x)[|f(x)| \geq |x|^2]$) reductions.

[2] *It is important to note that the classes we discuss indeed do have $\leq_m^{1\text{-}L}$-complete sets (and $\leq_m^{1\text{-}NL}$-complete sets), as, if this were not the case, our results would be without meaning.* Indeed, even the canonical universal set (see, e.g., [Har78a]) is complete via one-way reductions, as reductions to the universal set simply (1) prepend a machine, (2) copy the input to the output while counting its size, and (3) print padding based on the input's size. Further discussion of the stunning relationships between one-way completeness and completeness can be found in [HIM78,HM81].

Our superscripts include p representing polynomial time, L representing logarithmic space, NL representing nondeterministic logspace, 1-L representing one-way logarithmic space, and 1-NL representing one-way nondeterministic logspace. 1-L reductions were defined and studied by Hartmanis, Immerman, and Mahaney.

Definition 2.1 [HIM78] A 1-L *reduction* is a function computed by Turing machines[3] with (1) a work tape with two-way access to $\lceil \log n \rceil$ tape cells that are laid off in advance, (2) a one-way input tape, and (3) a one-way output tape.

1-L reductions have been studied in other papers than [HIM78], notably including a paper by Hartmanis and Mahaney that defines the complexity classes 1-L and 1-NL, representing deterministic and nondeterministic one-way logspace [HM81,All88]. As a natural extension, we now define 1-NL reductions, which are simply the class of single-valued total functions computable by one-way nondeterministic logspace Turing machines.

Definition 2.2 (1-NL functions and reductions)

1. **[HM81]** A 1-NL *Turing machine* is a nondeterministic Turing machine with a one-way input tape (with end markers) and, for an input of length n, a two-way read-write work tape of length $\lceil \log n \rceil$ (with end markers).

2. (Adapting the notion of nondeterministic reduction of Book [Boo90, Section 6] to the class 1-NL) A *1-NL function* is any function computed by some 1-NL Turing machine augmented by a one-way output tape. Such a function, which in general may be partial and multivalued, maps from an input x to the set of values written on the output tape by accepting computation paths. A function is said to be a *1-NL reduction* if it is a single-valued total 1-NL function.[4]

3. A single-valued, total function h is said to be *1-NL invertible* if its inverse (which in general will be a multi-valued partial function) is a 1-NL function. Note that if h is one-one, then its inverse will be single-valued.

[3] Number of work tapes is not an issue as, for space classes, many tapes can easily be reduced to one tape, with no change in the work-tape alphabet size.

[4] Note that not every path of such a machine need accept (and thus output its value). However, every path that does accept must output the *same* value, and for every input, some path must output a value.

Definition 2.3 (NL-functions and reductions)

1. (Adapting the notion of nondeterministic reduction of Book [Boo90, Section 6] to the class NL) An *NL function* is any function computed by some NL Turing machine augmented by a one-way output tape. Such a function, which in general may be partial and multivalued, maps from an input x to the set of values written on the output tape by accepting computation paths. A function is said to be an *NL reduction* if it is a single-valued total NL function.

2. A single-valued, total function h is said to be *NL invertible* if its inverse (which in general will be a multi-valued partial function) is an NL function. Note that if h is one-one, then its inverse will be single-valued.

3. We say that sets A and B are *NL-isomorphic* if there is a one-one, onto, NL-computable, NL-invertible reduction from A to B.

Definition 2.4 [Sel78,Lon82]

1. Consider a nondeterministic Turing machine with the property that, for every input, each computation path ends in one of three distinguished states: "accept," "reject," and "no-comment" (different paths may end in different states).

   We will say that such a machine, N, is a *strong nondeterministic Turing machine* if, for each input x, it holds that either:

   (a) at least one path of $N(x)$ ends in the "accept" state and no paths of $N(x)$ end in the "reject" state, or

   (b) at least one path of $N(x)$ ends in the "reject" state and no paths of $N(x)$ end in the "accept" state.

2. A strong nondeterministic machine N is said to *accept* an input x if $N(x)$ has at least one accepting computation path; it is said to *reject* input x otherwise.

To avoid confusion in constructions that involve both strong nondeterministic machines and standard nondeterministic machines, we will often refer to computation paths of strong machines as "strong-accepting" and "strong-rejecting," and to paths of standard machines as "standard-accepting" and "standard-rejecting."

From the result of Immerman and Szelepcsényi [Imm88,Sze88], it follows immediately that any language $L \in \text{NSPACE}[\log^2 n]$ can be accepted by a strong nondeterministic machine with the same space bound.

In analogy to the class US ([BG82]) we will consider the class US-SPACE$[\log^2 n]$: A language L is in this class if there is some $\mathcal{O}(\log^2 n)$ space nondeterministic Turing machine N such that, for all inputs x, it holds that $x \in L$ if and only if N accepts x on exactly one path.[5] Since the Immerman-Szelepcsényi result indeed collapses the entire boolean hierarchy (and more) over NSPACE classes, and since (for nice functions f) US-SPACE$[f(n)]$ is in the boolean hierarchy over NSPACE$[f(n)]$,[6] it follows immediately that US-SPACE$[f(n)]$ = NSPACE$[f(n)]$ for those functions f for which the [Imm88,Sze88] result applies. In particular, we have the following.

Proposition 2.5 US-SPACE$[\log^2 n]$ = NSPACE$[\log^2 n]$.

We say that sets A and B are P-isomorphic if there is a one-one, onto, polynomial-time invertible, polynomial-time computable reduction from A to B [BH77]. For a given reduction r, an $\leq_r$-degree is an equivalence class with respect to $\leq_r$ reductions.

3 Results

Typical of known equivalence results for complete degrees[7] is the following, whose original proof has been simplified by Watanabe [Wat85], and simplified still further by Ganesan and Homer [GH89].

Theorem 3.1 [Ber77] All $\leq^p_m$-complete sets for EXP are $\leq^p_{1l}$-equivalent.

Similar results hold for nondeterministic exponential time and PSPACE.

Theorem 3.2

[5]Note that on some inputs, the machine might well have more than one accepting path, and on those inputs it is considered to reject. This can be contrasted with the very different USPACE$[f(n)]$ classes (see [BHS90]), the analogs of Valiant's class UP [Val76], in which machines are forbidden to ever have more than one accepting path.

[6]This may be thought of as the space version of Blass and Gurevich's [BG82] inclusion US $\subseteq$ DP, though there is a slight subtlety in the proof.

[7]Excellent surveys of this work and of the general question of isomorphism have been written by Kurtz, Mahaney, and Royer [KMR90] and Young [You90]. We refer the reader to these works for general background.

1. **[GH89]** All $\leq_m^p$-complete sets for NEXP are $\leq_1^p$-equivalent via reductions that shrink their input at most logarithmically.

2. **[All88]** All $\leq_m^{1\text{-}L}$-complete sets for PSPACE are P-isomorphic.

Our goal is to prove equivalence and isomorphism results for small complexity classes. The above results are based on constructing a "magic set" that diagonalizes against reductions to itself. Unfortunately, these key diagonalizations require that the classes have the power to perform relatively powerful (i.e., linear-sized) nondeterministic guessing. For the small classes we'll discuss, such guessing is beyond the classes' apparent power. Nonetheless, our techniques will allow us to overcome this obstacle, and obtain a valid diagonalization.

Our techniques extend the diagonalization framework developed by [Har78b,Wat85, GH89]. In particular, we convert the existential quantifier of that framework to a seemingly more demanding "exists exactly one" quantifier. We then show—using the Selman-Long notion of strong computation—that the "exists exactly one" quantifier can, within the diagonalization setting, be evaluated correctly. Furthermore, after evaluating the quantifier, we show that—exactly *because* we're using an "exactly one" quantifier—the n bits of the existentially quantified string, though lost to us as we lacked the space to store them, can be regenerated.

We now prove our main results. We'll use NSPACE[$\log^2 n$] for concreteness, however, as discussed later, our results apply broadly to nondeterministic space classes above NSPACE[$\log n$], also to certain large deterministic time classes.

Theorem 3.3 All $\leq_m^{1\text{-}L}$-complete sets for NSPACE[$\log^2 n$] are equivalent under $\leq_{1l_t}^{1\text{-}L}$-reductions (and, indeed, are equivalent under $\leq_{1ql_t}^{1\text{-}L}$-reductions).

Proof of Theorem 3.3

First, we fix an enumeration $(M_i)_{i\in N}$ of 1-L transducers that compute functions $(f_i)_{i\in N}$ such that the corresponding universal function $U(i,x) = f_i(x)$ is computable in deterministic space $\mathcal{O}(\log^2 n)$.

For any pair of 1-L-complete sets A and B in NSPACE[$\log^2 n$] we define a set C in NSPACE[$\log^2 n$] that will will allow us to construct a $\leq_{1ql_t}^{1\text{-}L}$-reduction g from A to B. This set C is defined by the following algorithm:
On input $z = \langle i, x\rangle$ do:

1. if $|f_i(z)| \leq |z|^2$

2. then $z \in C \iff f_i(z) \notin B$

3. else if there is exactly one $y <_{lex} x$ such that $f_i(z) = f_i(\langle i, y \rangle)$

4. then $z \in C \iff y \notin A$

5. else $z \in C \iff x \in A$.

It follows directly that any reduction f_j from C to B must, on the set $C_j = \{z \mid (\exists x \in \Sigma^*)[z = \langle j, x \rangle]\}$, be quadratically length-increasing and one-one. Why? Suppose f_j is not quadratically length-increasing on C_j. In this case f_j can not be a reduction, because of line 2 of the algorithm. Suppose f_j is not one-one on C_j. Then there is a string x (for example, the lexicographically second smallest string that maps to some string that f_j maps more than one string to) such that there exists exactly one $y <_{lex} x$ such that $f_j(\langle j, x \rangle) = f_j(\langle j, y \rangle)$. But since in this case, by construction, $y \in A \iff \langle j, y \rangle \in C$ and $y \in A \iff \langle j, x \rangle \notin C$ hold, the function f_j cannot be a reduction from C to B. It follows by line 5 that if f_j is a reduction from C to B then $h(x) =_{def} \langle j, x \rangle$ is a reduction from A to C, and that $g(x) =_{def} f_j(h(x)) = f_j(\langle j, x \rangle)$ is the required 1-1, quadratically length-increasing, 1-L-computable reduction from A to B.

It remains to prove that the set C is in NSPACE[$\log^2 n$]. Once this is done it follows from the completeness of B that a reduction f_j from C to B exists, and thus by the above reasoning the desired reduction g exists.

To this point, the construction has generally been like the previous constructions within Hartmanis's diagonalization framework by [GH89,Har78b,Wat85], except that the "exactly one" condition on line 3 of the algorithm is different and will be crucial. In previous constructions, it has been immediate that M is in the appropriate class (usually, a very large class). In our case, we must evaluate the linear-sized existential quantifier on line 3; however, we have only the power of NSPACE[$\log^2 n$] with which to do the evaluation. Furthermore, our alteration of the standard framework's "exists" quantification to our algorithm's "exists exactly one" would seem to give us an even more complex task. We now show—using the power given to strong computation by inductive counting, and exploiting the fact that our reductions are one-way—that there indeed is a "strong" NSPACE[$\log^2 n$] machine that can evaluate the "exists exactly one" condition.

We describe a (standard) nondeterministic machine N that accepts the set C. As discussed in Section 2, we will refer to the paths of the underlying strong machines as

"strong accepting" and "strong-rejecting" paths; the paths of the standard machines will be called "standard-accepting" and "standard-rejecting" paths.

Let N_A and N_B be strong machines that accept A and B, respectively, in space $\mathcal{O}(\log^2 n)$. Fix an input $z = \langle i, x \rangle$. N starts by checking the condition $|f_i(z)| \leq |z|^2$ in line 1 of the algorithm. It is easily seen that this can be done on the available space. If $|f_i(z)| \leq |z|^2$, then N simulates N_B on the input $f_i(\langle i, x \rangle)$, which is computed bitwise as the simulation of N_B demands input, and any path of N_B that denotes strong-rejection causes N to immediately standard-accept; no-comment and strong-accept paths cause N to standard-reject on its corresponding simulation paths. Since the length of $f_i(\langle i, x \rangle)$ is bounded by $|\langle i, x \rangle|^2$ the required space is in $\mathcal{O}(\log^2 n)$.

To prove that the rest of the algorithm—namely, the case in which $|f_i(z)| > |z|^2$—can also be accomplished in NSPACE[$\log^2 n$] let:

$$C_U = \{ \langle i, x \rangle \mid \text{there is exactly one } y <_{lex} x \text{ such that } f_i(\langle i, x \rangle) = f_i(\langle i, y \rangle) \}.$$

We first show the following lemma:

Lemma 3.4 $C_U \in$ US-SPACE[$\log^2 n$].

Proof of Lemma 3.4 Informally, for an input $\langle i, x \rangle$, we guess a string $y <_{lex} x$, interleave the computations of M_i on the inputs $\langle i, x \rangle$ and $\langle i, y \rangle$, and compare the outputs bitwise. We give a formal proof below; note that the rate at which we guess y is determined by the input needs of f_i.

Without loss of generality, we globally assume that our one-way reductions never halt until they have read all their input string (up to and including the endmarker).

We describe a nondeterministic machine N' accepting C_U. N' starts by simulating M_i on input $\langle i, x \rangle$—i.e., it computes $f_i(\langle i, x \rangle)$—without holding the output on the work tape. If N' has not yet ensured that $y <_{lex} x$ then, each time a bit of x is to be read by M_i (say the jth bit), N' guesses nondeterministically whether y first differs from x on this bit. We have two cases:

Case 1 *N' guesses that y and x agree on the current bit.* In this case, if the current bit of x is the endmarker, then this particular computation path has failed to ensure that $y <_{lex} x$, so this path standard-rejects. On the other hand, if the current bit of x is a 0 or a 1, the simulation of the computation of $f_i(\langle i, x \rangle)$ is continued until the machine attempts to read the next bit, at which time N' again chooses between Case 1 and Case 2.

Case 2 *N' guesses that y and x differ, for the first time, on the current bit.* There are two subcases, one to allow N' to guess that the strings differ because y ends, and one to

allow N' to guess that y and x first differ because y has a 0 as this bit but x has a 1 as this bit (recall, we must ensure that $y <_{lex} x$.). Note that if the bit of x read is the endmarker, neither case is possible and the current path standard-rejects. If the bit of x read is a 0, only Case 2a is possible. If the bit of x read is a 1, both Cases 2a and 2b are possible, and, nondeterministically, both occur.

Case 2a *N' guesses that y and x differ in that y contains no more bits.* In this case y is a strict prefix of x, and the simulations of $f_i(\langle i, x\rangle)$ and $f_i(\langle i, y\rangle)$ can be finished by comparing the output strings bitwise. If a bit occurs on which $f_i(\langle i, x\rangle)$ and $f_i(\langle i, y\rangle)$ differ N' standard-rejects the input. It standard-accepts the input if $f_i(\langle i, x\rangle)$ and $f_i(\langle i, y\rangle)$ are equal up to the last bit.

Case 2b *N' guesses that y and x differ in that the current bit of y is a 0 and the current bit of x is a 1.* The simulation of $f_i(\langle i, x\rangle)$ and $f_i(\langle i, y\rangle)$ is continued by comparing the outputs bitwise. Each time a new input bit for y is required it is checked whether the length of y exceeds the length of x. If so the simulation standard-rejects. If not it is guessed nondeterministically whether the new input bit is 0, or 1, or whether the end of y is reached. Again, N' standard-rejects the input if a bit is found on which $f_i(\langle i, x\rangle)$ and $f_i(\langle i, y\rangle)$ differ. N' standard-accepts the input if $f_i(\langle i, x\rangle)$ and $f_i(\langle i, y\rangle)$ are equal up to the last bit.

It is clear that N' has as many accepting paths as there exist strings $y <_{lex} x$ for which $f_i(\langle i, x\rangle) = f_i(\langle i, y\rangle)$ holds, and thus $C_U \in$ US-SPACE$[\log^2 n]$.

End of Proof of Lemma 3.4 ∎

By Proposition 2.5 and Lemma 3.4 we know that there exists a strong machine N_U that accepts C_U. The nondeterministic machine N, which we are constructing to accept C, now continues as follows: N simulates N_U on input $\langle i, x\rangle$. If N_U strongly-rejects $\langle i, x\rangle$, then N_A is run on input x and N standard-accepts $\langle i, x\rangle$ if and only if N_A strongly accepts x (line 5 of the algorithm). On the other hand, if N_U strongly-accepts $\langle i, x\rangle$, i.e., the condition of line 3 of the algorithm is true, then we wish N to standard-accept $\langle i, x\rangle$ if and only if N_A strongly rejects y—thus realizing line 4 of the algorithm. Note, however, that we don't *have* y directly available, as on $\log^2 n$ space there is not enough room to retain y. Nonetheless, we can regenerate y and thus compute $N_A(y)$, as described in the following.

We will simulate $N_A(y)$, providing it with inputs on demand. Note that, in the case we are in, there is *exactly* one $y <_{lex} x$ such that $f_i(\langle i, x\rangle) = f_i(\langle i, y\rangle)$. As we are simulating $N_A(y)$, suppose it tries to read a bit of its "input" y, say the jth bit of y. We immediately begin a simulation of the strong machine for C_U; that is, we simulate $N_U(\langle i, x\rangle)$. Along

each path, we note what the jth bit is of the particular y guessed on that path.[8] Now, on the unique path of $N_U(\langle i,x\rangle)$ that computes the correct y—i.e., the unique $y <_{lex} x$ such that $f_i(\langle i,x\rangle) = f_i(\langle i,y\rangle)$—we use the jth bit of y as the desired bit of y in our simulation of $N_A(y)$, and continue the simulation (repeatedly simulating $N_U(\langle i,x\rangle)$ each time we need a new bit of y).

Finally, note that the "exactly one" condition of our algorithm was used centrally. If we had tried the above regeneration scheme in a case where there were two strings, y' and y'', each $<_{lex} x$, such that $f_i(\langle i,x\rangle) = f_i(\langle i,y'\rangle) = f_i(\langle i,x\rangle) = f_i(\langle i,y''\rangle)$, then the above procedure would hopelessly jumble the bits (even to the point of giving different answers for the same bit at different times along a single path).

End of Proof of Theorem 3.3

∎

The analogous result holds for nondeterministic reductions and small space classes:

Theorem 3.5 All $\leq_m^{1\text{-}NL}$-complete sets for NSPACE[$\log^2 n$] are equivalent under $\leq_{1l}^{1\text{-}NL}$-reductions (and, indeed, under $\leq_{1ql}^{1\text{-}NL}$-reductions).

Proof

The proof builds on the same ideas as the proof of the deterministic case but requires additional consideration. Since we cannot assume that there exists an enumeration of single-valued 1-NL-transducers that has a universal function with the properties of the proof of Theorem 3.3, we have to deal with general—and thus partial and multivalued—1-NL-transducers.

Let $(N_i)_{i\in N}$ be an enumeration of 1-NL-transducers that compute functions $(f_i)_{i\in N}$ such that the corresponding universal function $U(i,x) = f_i(x)$ is computable in nondeterministic space $\mathcal{O}(\log^2 n)$.

Again, for any pair of 1-NL-complete sets A and B in NSPACE[$\log^2 n$], we define a set C in NSPACE[$\log^2 n$] that is $\leq_{1ql}^{1\text{-}NL}$-reducible to B and yields a $\leq_{1ql}^{1\text{-}NL}$-reduction g from A to B. C is defined by the following algorithm:

On input $z = \langle i,x\rangle$ do:

[8] Though we did not discuss earlier exactly how the conversion from US-SPACE machines to NSPACE machines to strong NSPACE machines worked, it is in fact true that we may ensure that our strong simulation indeed retains the ability to note a particular bit of the guessed y; one could ensure this formally by also considering a language of the form $\{\langle i,x,j,b\rangle \mid$(1) there is exactly one $y <_{lex} x$ such that $f_i(\langle i,x\rangle) = f_i(\langle i,y\rangle)$, and (2) the jth bit of this unique y is $b\}$, which can be seen to be in NSPACE[$\log^2 n$].

1. if either (1) there is no accepting computation path of $N_i(z)$ (i.e., N_i on input z does not output a string) or (2) there exist two accepting computation paths y_1 and y_2 such that N_i on input z outputs w_1 on path y_1 and w_2 on path y_2 and $w_1 \neq w_2$, then $z \notin C$

2. else if $|f_i(z)| \leq |z|^2$

   (*where $f_i(z)$ denotes the uniquely determined value that occurs on one or more accepting computation paths of N_i on input z*)

3. then $z \in C \iff f_i(z) \notin B$

4. else if there is exactly one $y <_{lex} x$ such that $f_i(z) = f_i(\langle i, y \rangle)$

5. then $z \in C \iff y \notin A$

6. else $z \in C \iff x \in A$.

Line 1 in the algorithm checks whether the machine N_i on input z is defined and single-valued. We have to show that this condition can be checked on nondeterministic space $\mathcal{O}(\log^2 n)$. Note that the set:

C_{S_1} = $\{\langle i, x \rangle |$ N_i on input $\langle i, x \rangle$ does not output a string on any accepting computation path$\}$

is in coNSPACE[$\log^2 n$] and thus in NSPACE[$\log^2 n$], and that the set:

C_{S_2} = $\{\langle i, x \rangle |$ there exist two accepting computation paths y_1 and y_2 such that N_i on input z outputs w_1 on path y_1 and w_2 on path y_2 and $w_1 \neq w_2\}$

is in NSPACE[$\log^2 n$]. Thus there exists a strong machine N_0 that strongly accepts or strongly rejects $C_{S_1} \bigcup C_{S_2}$, and accordingly computes whether the condition in line 1 is true or false. The rest of the proof follows from arguments generally analogous to those of the deterministic case. ∎

Theorems 3.3 and 3.5 provide quadratic length-increasing equivalence. Combined with inversion results, this would yield isomorphism, following the program outlined by Berman and Hartmanis over a decade ago [BH77,Har78b], and embodied in Lemma 3.6.

Lemma 3.6 [Har78b] Let p and q be, respectively, L-reductions of A to B and B to A, $A \subseteq \Sigma^*$, $B \subseteq \Gamma^*$. A and B are L-isomorphic if the following conditions hold:

(a) p and q are one-one,

(b) p^{-1} and q^{-1} are L-computable, and

(c) $|p(z)| > |z|^2$ and $|q(z)| > |z|^2$.

We generalize Hartmanis's result to the case of NL-reductions in the following lemma, which will later be used to establish the NL-isomorphism of all $\leq_m^{1\text{-}NL}$-complete sets for NSPACE[$\log^2 n$].

Lemma 3.7 Let p and q be, respectively, 1-NL-reductions of A to B and B to A, $A \subseteq \Sigma^*$, $B \subseteq \Gamma^*$. A and B are NL-isomorphic if the following conditions hold:

(a) p and q are one-one, and

(b) $|p(z)| > |z|^2$ and $|q(z)| > |z|^2$.

Proof: The proof follows the technique of [Har78b], crucially also relying on the power of strong computation and inductive counting to allow failsafe inversion of non-onto functions.

Let h be any honest, one-one, 1-NL reduction. The set $\widehat{L} = \{y \in \Sigma^* |\ (\forall x)[h(x) \neq y]\}$ is in coNL, and thus is in NL, and thus is accepted by a strong NL machine. So we may construct a nondeterministic logspace machine $\widehat{N}_h$ that behaves on input y as follows:

(1) If $h^{-1}(y)$ is defined $\widehat{N}_h(y)$ outputs the unique string x for which $h(x) = y$ holds.

(2) If $h^{-1}(y)$ is not defined $\widehat{N}_h(y)$ outputs "$\star$."

$\widehat{N}_h$ works as follows. First, it uses a strong machine for $\widehat{L}$ to determine whether to output "$\star$." If "$\star$" is not the correct output, $\widehat{N}_h$ guesses (bit by bit) all strings x of appropriate (with respect to the honesty of the reduction) lengths and runs $h(x)$ as it guesses x, writing the guessed bits to its output tape and comparing, bitwise, $h(x)$ with y. We accept on the path that guesses x such that $h(x) = y$; thus, this path outputs x. All other paths reject. We'll in particular be concerned with the machines $\widehat{N}_p$ and $\widehat{N}_q$.

Our isomorphism is defined by:

$$\phi(z) \ = \ \text{if } z \in R_1 \text{ then } p(z) \text{ else } q^{-1}(z),$$

where $R_1 = \{(q \circ p)^k(x) |\ k \geq 0 \text{ and } x \notin q(\Gamma^*)\}$. Note that:

$$\phi^{-1}(z) \ = \ \text{if } z \in S_2 \text{ then } p^{-1}(z) \text{ else } q(z),$$

where $S_2 = \{p \circ (q \circ p)^k(x) | \ k \geq 0 \text{ and } x \notin q(\Gamma^*)\}$. Also, we define $R_2 = \{q \circ (p \circ q)^k(x) | \ k \geq 0 \text{ and } x \notin p(\Sigma^*)\}$. For the reasons given in [Har78b, Theorem 2.1], it follows that ϕ is an isomorphism between A and B. The crucial point is to show that ϕ is computable in nondeterministic logspace. It is sufficient to show that there is a strong nondeterministic machine that decides the condition $z \in R_1$ ($z \in S_2$). Once this is done we know that the isomorphism is NL-computable (using $\widehat{N}_q$ and p) and the claim follows.

Let $\widehat{N}_q$ be as described earlier—a total nondeterministic logspace machine that computes the inverse of q; i.e., on input x the output is $q^{-1}(x)$ if it is defined and "$\star$" otherwise. Similarly, let $\widehat{N}_p$ be a total nondeterministic logspace machine that computes the inverse of p. To determine whether $x \in R_1$ or $x \in R_2$ holds we have to find the minimal $k \geq 0$ such that:

$$(q^{-1} \circ p^{-1})^k \circ q^{-1}(x) = \star \text{ or}$$
$$p^{-1} \circ (q^{-1} \circ p^{-1})^k \circ q^{-1}(x) = \star$$

holds, and which of the above holds for that k. As in [Har78b], we compute (in order) $q^{-1}(x)$, $p^{-1}(q^{-1}(x))$, $q^{-1}(p^{-1}(q^{-1}(x)))$,... by simulating $\widehat{N}_q$ and $\widehat{N}_p$. Since the outputs might be too large to be held on the work tape they will be recomputed whenever they are needed as an input. Since the simulated machines are nondeterministic we have to deal with two difficulties. First, the simulated path of the nondeterministic computation might not end in an accepting final state; in this case, this path of the simulating machine halts and outputs nothing. Second, though $\widehat{N}_q$ and $\widehat{N}_p$ compute single-valued, total functions, on an input x the computation along some path might have written symbols on the output tape though it does not end in an accepting final state. Thus the output on such a path does not necessarily correspond to the correct value of the computation (e.g., $q^{-1}(x)$ or "$\star$" in the case of $\widehat{N}_q$), and we have to make sure that these incorrect outputs are ignored.

We first compute $q^{-1}(x)$ and test whether it is "$\star$." If it is not, our next task it to test whether $p^{-1} \circ q^{-1}(x) = \star$. Let us describe how this is done. We start by simulating $\widehat{N}_p$. When it asks for its first bit of input, we store the configuration of $\widehat{N}_p$ and begin a simulation of $\widehat{N}_q(x)$, noting the first bit. As soon as $\widehat{N}_q$ reaches an accepting final state (along the path being nondeterministically followed), the simulation of $\widehat{N}_p$ continues, using as the desired input bit the bit that was the first bit computed along that path. When $\widehat{N}_p$ tries to read future bits of its input, $q^{-1}(x)$, we compute them in the same fashion. Eventually, we determine (strongly) whether $p^{-1} \circ q^{-1}(x) = \star$ or not. If not, we move on to computing whether $q^{-1} \circ p^{-1} \circ q^{-1}(x) = \star$, and so on, by dynamically maintaining as stack of

configurations of these machines. At such stages, we repeat the above procedure, with the appropriate number of levels of machines with inputs generated by the output of a machine with inputs generated by... and so on. This scheme is essentially that of Hartmanis; the crucial distinction is that, when the path we are on sees the bit of input that it is looking for, it cannot use it immediately; rather, it must drive the simulation through to completion in order to ensure that the bit output is on an accepting computation path. As in [Har78b, Proof of Theorem 2.1] this simulation can be carried out in $\mathcal{O}(\log n)$ space; in particular, since the reductions are quadratically length-increasing, the space used to maintain the stack is less than $\log n + \log \sqrt{n} + \log \sqrt[4]{n} + \cdots$, and thus is $\mathcal{O}(\log n)$. ∎

Thus, we have:

Theorem 3.8 All $\leq_m^{1\text{-}NL}$-complete sets for NSPACE[$\log^2 n$] are NL-isomorphic (and thus P-isomorphic).

Proof of Theorem 3.8 It follows from Theorem 3.5 that all sets that are $\leq_m^{1-NL}$-complete for NSPACE[$\log^2 n$] are equivalent under 1-NL reductions that are one-one, and quadratically length-increasing. Thus they are NL-isomorphic by Lemma 3.7. ∎

The theorems of this paper, though stated for NSPACE[$\log^2 n$], apply far more generally. Informally put, they apply to any standard nice (space-constructible non-decreasing space-bounds, closed under quadratic stretching) nondeterministic space class above nondeterministic logspace. In particular, the theorems apply to NPSPACE. Additionally, they apply to many deterministic space classes at or above $\mathrm{E} = \bigcup_{k>0} \mathrm{DSPACE}[2^{cn}]$; for example, Theorem 3.3 applies to EXP, and if one replaces *1qli* with *1li*, then it applies even to E. (Indeed, since the one-way nature of our reductions is used only (1) for inversion, and (2) to handle the requirements of sublinear space, for large classes our *1li* results hold for $\leq_m^L$ as well as $\leq_m^{1\text{-}L}$ (see Corollary 3.9, part 1).)

Since NPSPACE = PSPACE [Sav70], let us compare the implications of the preceding theorems with Allender's result. Allender [All88] proved that all 1-L complete sets for PSPACE are P-isomorphic. We have succeeded in weakening the strength of the isomorphism needed from P-isomorphism to NL-isomorphism, while simultaneously broadening the class of isomorphic sets from all 1-L complete sets to all 1-NL complete sets. Also, maintaining 1-L reductions, even logspace reductions suffice to achieve *1li* equivalence. The corollary below makes these claims explicit. Though the following new results are about the large class PSPACE, it should be emphasized that the results of this paper—unlike any previous work on collapsing degrees—apply even to *sublinear* nondeterministic space

classes.

Corollary 3.9

1. All $\leq_m^{1\text{-}L}$-complete ($\leq_m^L$-complete) sets for PSPACE are $\leq_{1l}^{1\text{-}L}$-equivalent ($\leq_m^L$-equivalent).

2. All $\leq_m^{1\text{-}NL}$-complete set for PSPACE are NL-isomorphic (and thus P-isomorphic).

4 Conclusions

By exploiting the power vested in strong computation by inductive counting, this paper has provided the first known collapses of sublinear space degrees. Moreover, applied to larger space classes such as PSPACE, this paper's techniques sharply strengthen previous results.

Acknowledgments

We are grateful to Stuart Kurtz for providing an early draft of [KMR90], to Steven Homer for providing a copy of [BHT90], to William Gasarch for proofreading an earlier draft of this paper, and to Eric Allender, Gerhard Buntrock, and Uwe Schöning for helpful comments.

References

[All88] E. Allender. Isomorphisms and 1-L reductions. *Journal of Computer and System Sciences*, 36(6):336–350, 1988.

[Ber77] L. Berman. *Polynomial Reducibilities and Complete Sets.* PhD thesis, Cornell University, Ithaca, NY, 1977.

[BG82] A. Blass and Y. Gurevich. On the unique satisfiability problem. *Information and Control*, 55:80–88, 1982.

[BH77] L. Berman and J. Hartmanis. On isomorphisms and density of NP and other complete sets. *SIAM Journal on Computing*, 6(2):305–322, 1977.

[BHS90] G. Buntrock, L. Hemachandra, and D. Siefkes. Using inductive counting to simulate nondeterministic computation. In *Proceedings of the 15th Symposium on Mathematical Foundations of Computer Science*, pages 187–194. Springer-Verlag *Lecture Notes in Computer Science #452*, August 1990.

[BHT90] H. Buhrman, S. Homer, and L. Torenvliet. On complete sets for nondeterministic classes. Technical Report 90-013, Boston University Department of Computer Science, Boston, MA, 1990.

[Boo90] R. Book. On separating complexity classes. In *Proceedings of the 5th Structure in Complexity Theory Conference*, pages 299–304. IEEE Computer Society Press, July 1990.

[GH89] K. Ganesan and S. Homer. Complete problems and strong polynomial reducibilities. In *Proceedings of the 6th Annual Symposium on Theoretical Aspects of Computer Science*, pages 240–250. Springer-Verlag *Lecture Notes in Computer Science #349*, 1989.

[GJ86] J. Goldsmith and D. Joseph. Three results on the polynomial isomorphism of complete sets. In *Proceedings of the 27th IEEE Symposium on Foundations of Computer Science*, pages 390–397, 1986.

[Har78a] J. Hartmanis. *Feasible Computations and Provable Complexity Properties.* CBMS-NSF Regional Conference Series in Applied Mathematics #30. SIAM, 1978.

[Har78b] J. Hartmanis. On log-tape isomorphisms of complete sets. *Theoretical Computer Science*, 7(3):273–286, 1978.

[HH] J. Hartmanis and L. Hemachandra. One-way functions and the non-isomorphism of NP-complete sets. *Theoretical Computer Science.* To appear.

[HIM78] J. Hartmanis, N. Immerman, and S. Mahaney. One-way log-tape reductions. In *Proceedings of the 19th IEEE Symposium on Foundations of Computer Science*, pages 65–71, 1978.

[HM81] J. Hartmanis and S. Mahaney. Languages simultaneously complete for one-way and two-way log-tape automata. *SIAM Journal on Computing*, 10(2):383–390, 1981.

[HS89] S. Homer and A. Selman. Oracles for structural properties: the isomorphism problem and public-key cryptography. In *Proceedings of the 4th Structure in Complexity Theory Conference*, pages 3–14. IEEE Computer Society Press, June 1989.

[Imm88] N. Immerman. Nondeterministic space is closed under complementation. *SIAM Journal on Computing*, 17:935–938, 1988.

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

[KLD86] K. Ko, T. Long, and D. Du. On one-way functions and polynomial-time isomorphisms. *Theoretical Computer Science*, 47:263–276, 1986.

[KMR87] S. Kurtz, S. Mahaney, and J. Royer. Progress on collapsing degrees. In *Proceedings of the 2nd Structure in Complexity Theory Conference*, pages 126–131. IEEE Computer Society Press, June 1987.

[KMR89] S. Kurtz, S. Mahaney, and J. Royer. The isomorphism conjecture fails relative to a random oracle. In *Proceedings of the 21st ACM Symposium on Theory of Computing*, pages 157–166. ACM Press, May 1989.

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

[Kur83] S. Kurtz. A relativized failure of the Berman-Hartmanis conjecture. Technical Report TR83-001, University of Chicago Department of Computer Science, Chicago, IL, 1983.

[Lon82] T. Long. Strong nondeterministic polynomial-time reducibilities. *Theoretical Computer Science*, 21:1–25, 1982.

[Rus86] D. Russo. Optimal approximations of complete sets. In *Proceedings of the 1st Structure in Complexity Theory Conference*, pages 311–324. Springer-Verlag *Lecture Notes in Computer Science #223*, June 1986.

[Sav70] W. Savitch. Relationships between nondeterministic and deterministic tape complexities. *Journal of Computer and System Sciences*, 4(2):177–192, 1970.

[Sel78] A. Selman. Polynomial time enumeration reducibility. *SIAM Journal on Computing*, 7(4):440–457, 1978.

[Sze88] R. Szelepcsényi. The method of forced enumeration for nondeterministic automata. *Acta Informatica*, 26:279–284, 1988.

[Val76] L. Valiant. The relative complexity of checking and evaluating. *Information Processing Letters*, 5:20–23, 1976.

[Wat85] O. Watanabe. On one-one P-equivalence relations. *Theoretical Computer Science*, 38:157–165, 1985.

[You90] P. Young. Juris Hartmanis: Fundamental contributions to isomorphism problems. In A. Selman, editor, *Complexity Theory Retrospective*, pages 28–58. Springer-Verlag, 1990.