On the Isomorphism Conjecture for Weak Reducibilities

Manindra Agrawal
School of Mathematics, SPIC Science Foundation
Madras 600 017, India
email : manindra@ssf.ernet.in

Abstract

According to the isomorphism conjecture *all NP-complete sets are polynomial-time isomorphic to each other while according to the* encrypted complete set conjecture *there is a p-one-way function f and an NP-complete set A such that A and $f(A)$ are not polynomial-time isomorphic to each other. In this paper, these two conjectures are investigated for reducibilities weaker than polynomial-time. It is shown that:*

1. *Relative to reductions computed by one-way logspace DTMs, both the conjectures are false.*
2. *Relative to reductions computed by one-way logspace NTMs, the isomorphism conjecture is true.*
3. *Relative to reductions computed by one-way, multi-head, oblivious logspace DTMs, the encrypted complete set conjecture is false.*
4. *Relative to reductions computed by constant-scan logspace DTMs, the encrypted complete set conjecture is true.*

It is also shown that the complete degrees for NP under the latter two reducibilities coincide.

1 Introduction

The *isomorphism conjecture* states that all NP-complete sets are p-isomorphic to each other. It was proposed by Berman and Hartmanis [7] based on their observation that all NP-complete sets known at the time were indeed p-isomorphic to each other. Serious objections were raised against this conjecture by Joseph and Young [17]. They constructed a new type of NP-complete sets, called the *k-creative* sets, some of which did not appear to be p-isomorphic to the standard NP-complete sets. These sets (as pointed out in [22]) had the form $f(A)$ where f is a *p-one-way* function (such functions are 1-1, honest, polynomial-time computable but *not* p-invertible) and A, a paddable NP-complete set. Joseph and Young argued that since p-one-way functions are not p-invertible, there may be *no* p-invertible reduction of A to $f(A)$ which implies that $f(A)$ is not p-isomorphic to A. Based on this, they conjectured that there is a p-one-way function f and a paddable NP-complete set A such that $f(A)$ is not p-isomorphic to A. This conjecture has been referred in the literature as the *encrypted complete set conjecture* [19]. We shall consider an equivalent form of this conjecture which is closer to the spirit behind it, viz., there exists a p-one-way function f and a paddable NP-complete set A such that $A \not\leq^p_{1,li,i} f(A)$. (The equivalence of the two statements follows from a result in [7]: if $A \leq^p_{1,li,i} B$ and $B \leq^p_{1,li,i} A$ then A and B are p-isomorphic.) It is obvious that not both of these conjectures, viz., the isomorphism and the encrypted complete set, can be true simultaneously. So the question is—which of the two conjectures, if any, is true? As both the conjectures imply, amongst other things, P $\neq$ NP, it is a difficult question to answer. This has led to relocation of the conjectures to classes

that are provably larger than P, e.g., EXP, NEXP. As a proof of the encrypted complete set conjecture for these classes would still imply P $\neq$ NP, it remains a difficult problem. However, attempts to *disprove* it, and hence prove the isomorphism conjecture, have not been successful either for any of these classes. There have only been some partial collapsing results—the $\leq^p_m$-complete degree for EXP collapses to $\leq^p_{1,li}$-complete degree [6, 24, 11]; the $\leq^p_m$-complete degree for NEXP collapses to $\leq^p_1$-complete degree [11]. These failures are not surprising in view of certain relativization results—it has been shown [20] that relative to a *random* oracle (see [21] for definition), the encrypted complete set conjecture holds for NP, EXP, NEXP etc. On the other hand, it has been recently shown [9] that relative to a *sp-generic* oracle (see [9] for definition) the isomorphism conjecture is true for all these classes. A good survey of results concerning these two conjectures can be found in [19].

As the conjectures, even when relocated to the higher classes, have proved to be difficult to settle, one may try another approach by relocating them to *weaker reducibilities*. This leads us to the following generalization of the conjectures—for any class of reductions $\mathcal{F}(r)$, let the *r-isomorphism conjecture* be that all $\leq^r_m$-complete sets for NP are r-isomorphic; and the *r-encrypted complete set conjecture* be that there exists an *r-one-way* function f (i.e., f is a 1-1, honest, function in $\mathcal{F}(r)$ such that its inverse does not belong to $\mathcal{F}(r)$) and a $\leq^r_m$-complete set A for NP such that $A \not\leq^r_{1,li,i} f(A)$. These conjectures have been investigated for various reducibilities weaker than polynomial time [13, 5, 1], however, again, no answers have been found. In this paper, we provide, for the first time, answers to the two conjectures for several weak reducibilities. All the results below hold for any class closed under logspace reductions, we restrict ourselves to NP as it is the most interesting one.

We first consider *1-L reductions*, the class of functions computed by logspace bounded DTMs with a *one-way* input head. Complete degrees for these reductions are non-trivial—it has been shown [14] that all *natural* NP-complete sets are complete under 1-L reductions as well (though one can easily construct an NP-complete set which is not $\leq^{1-L}_m$-complete [14]). The structure of complete degrees under 1-L reductions has been investigated before [5, 11, 15, 8, 1], we improve on all the earlier results. We show that even though 1-L-one-way functions exist, the $\leq^{1-L}_m$-complete degree for NP collapses to $\leq^{1-L}_{1,li,i}$-complete degree. Thus the 1-L-encrypted complete set conjecture is false. With such a strong collapse of $\leq^{1-L}_m$-complete degree, one may expect the 1-L-isomorphism conjecture to be true, however, it turns out that this conjecture too is false. Nevertheless, by generalizing the class of reductions to *1-NL*, functions computed by logspace bounded NTMs with a one-way input head, we show that the 1-NL-isomorphism conjecture is true.

Next, we consider another generalization of 1-L reductions: *1-omL reductions*, functions computed by logspace bounded *oblivious* DTMs with *multiple* one-way input-heads. Such TMs can compute several p-one-way functions (defined by using the computation of TMs recognizing languages in UP$-$P), and therefore, are fairly powerful. We show that even the $\leq^{1-omL}_m$-complete degree for NP collapses to a $\leq^{1-omL}_{1,li,i}$-complete degree and therefore, the 1-omL-encrypted complete set conjecture too is false. The last class of reductions we consider are *c-L reductions*, computed by logspace bounded DTMs with their input head allowed a fixed constant number of left-to-right scans of the input tape. These reductions too are a generalization of 1-L reductions and form a proper subclass of 1-omL reductions. We show that the c-L-encrypted complete set conjecture is true. The last two reducibilities exhibit another interesting property: $\leq^{1-omL}_m$- and $\leq^{c-L}_m$-complete degrees for NP *coincide*, and all such complete sets are complete under one-one, length-increasing, c-L reductions with their inverses being 1-omL functions. This provides probably the first example of a reducibility (1-omL), the complete sets under which are also complete under a *strictly weaker* reducibility (c-L).

We begin the paper by first introducing the basic terminology the we use (section 2) and then

in section 3, we define the notion of computation sequence, which plays a crucial part in almost all our proofs. We then move on to giving the results for the four reducibilities in sections 4, 5, 6 and 7. In section 8 we discuss the implications of these results and compare our technique with the earlier ones. Finally, we list some open questions in section 9.

2 Preliminaries

The strings are over $\Sigma = \{0, 1\}$. To avoid confusion between strings and numbers, we write 0's and 1's of a string in boldface. For a string s, $|s|$ denotes its length. For a finite set of strings S, $\|S\|$ denotes the number of strings in S. Set $\Sigma_{=n}$ denotes the set of all strings of length n. For any string s and for any number i, $1 \leq i \leq |s|$, $s[i]$ denotes the i^{th} bit of s.

Our model of computation is Turing machines with a read-only input tape, a write-only output tape and a read-write work tape.

For a resource bound r on TMs, we denote by $\mathcal{F}(r)$ the class of total functions computed by TMs within the resource bound of r. For the class of functions $\mathcal{F}(r)$, we say that f is an *r-computable function*, or simply, an *r function*, if $f \in \mathcal{F}(r)$; and f is *r-invertible* if there is a function $g \in \mathcal{F}(r)$ such that $g(f(x)) = x$ for every x. We say that set $A \leq^r_m$ ($\leq^r_{1,li}$; $\leq^r_{1,li,i}$; $\leq^r_{1,qli,i}$) B if there is a many-one (one-one, length-increasing; one-one, length-increasing and r-invertible; one-one, quadratic length-increasing and r-invertible) r-computable function f reducing A to B. Set A is *$\leq^r_m$-hard* for class $\mathcal{C}$ if for every $B \in \mathcal{C}$, $B \leq^r_m A$. Set A is *$\leq^r_m$-complete* for class $\mathcal{C}$ if A is $\leq^r_m$-hard for $\mathcal{C}$ and $A \in \mathcal{C}$. For the class NP, an NP-*complete* set is a $\leq^p_m$-complete set for NP. The *$\leq^r_m$-complete degree* for $\mathcal{C}$ is defined to be the class of all $\leq^r_m$-complete sets for $\mathcal{C}$. Similarly, one defines these notions for $\leq^r_{1,li}$, $\leq^r_{1,li,i}$ and $\leq^r_{1,qli,i}$ reductions. We say that the set A is *r-isomorphic* to set B if there exists a bijection f between A and B with both f and f^{-1} being r-computable.

A *1-L TM* is a deterministic Turing m/c with a read-only input tape, a write-only output tape and a logspace-bounded work tape such that its input head is *one-way*, i.e., it moves from left to right only. Further, at the beginning of the computation, $\mathbf{1}^{\lceil \log n \rceil}$ is written on the work tape, where n is the length of the input. These functions were first defined in [14] for studying complete sets for DLOG. The class of $\leq^{1\text{-}L}_m$-complete sets for NP is a fairly large one: it was shown in [14] that all known *natural* NP-complete sets are $\leq^{1\text{-}L}_m$-complete as well.

It is worth noting at this point that 1-L functions are not closed under composition as 1-L TMs need $\mathbf{1}^{\lceil \log n \rceil}$ written on the work tape at the beginning of the computation.
(Example: Let $f(x) = x\mathbf{1}^{2^{2\lceil \log |x| \rceil + 1} - |x|}$ if x ends in a $\mathbf{0}$, $f(x) = x\mathbf{0}^{2^{2\lceil \log |x| \rceil} - |x|}$ otherwise; $g(z) = \mathbf{0}x$ if $\lceil \log |z| \rceil$ is odd and $z = x\mathbf{01}^k$ for some $k \geq 0$, $g(z) = \mathbf{1}x$ if $\lceil \log |z| \rceil$ is even and $z = x\mathbf{10}^k$ for some $k \geq 0$. Both f and g are 1-L functions but their composition $g \circ f$—$(g \circ f)(x\mathbf{0}) = \mathbf{0}x$ and $(g \circ f)(x\mathbf{1}) = \mathbf{1}x$—is not as shown in Proposition 4.10 below.)
Nevertheless, the notion of our interest, viz., $\leq^{1\text{-}L}_m$-complete degrees, is well defined.

A *1-NL TM* is a non-deterministic Turing m/c with the rest of the conditions being same as for a 1-L TM. Class $\mathcal{F}$(1-NL) contains total functions that are computed by 1-NL TMs that output the *same* string on all accepting paths. These functions are closed under composition as a 1-NL TM can guess the length of the input and verify it later on.

A *k-L TM*, $k > 0$, is a deterministic Turing m/c with the same conditions as for a 1-L TM except that its input head is allowed a maximum of k left-to-right scans of the input tape. A *c-L TM* is one that is a k-L TM for some $k > 0$. The class of total functions computed by c-L TMs is $\mathcal{F}$(c-L) $= \cup_{k>0}\mathcal{F}(k\text{-L})$. It is clear that the $\mathcal{F}$(c-L) is closed under composition (composition of a k_1-L and k_2-L function is a $k_2.(k_1 + 1)$-L function).

A *1-mL TM* is a deterministic Turing m/c with a read-only input tape, a write-only output tape and a logspace-bounded work tape. It may have *more* than one input heads with all of them

being one-way. The class $\mathcal{F}$(1-mL) is closed under composition. We shall mainly be interested in a subclass of these functions computed by 1-mL TMs that are *oblivious*. This condition implies that the movement of the input heads depends only on the length of the input and not on the particular string of that length. We refer to oblivious 1-mL TMs as *1-omL TMs*. The class $\mathcal{F}$(1-omL) is not closed under composition. An example is—$f(x) = y$ where $x = \mathbf{1}^j\mathbf{0}y\mathbf{01}^k$ for $j, k \geq 0$; $g(y) = \mathbf{1}$ if $y = zz$, $g(y) = \mathbf{0}$ otherwise. Both f and g are 1-omL functions but their composition $g \circ f$ is not. We leave the proof to the reader.

3 Computation sequences of 1-L TMs

Throughout the section, we shall be dealing with Turing machines computing 1-L functions. Without loss of generality, we can assume that these machines have a unique starting state and they halt only when the input head is placed on the cell immediately to the right of the input. In this section, whenever we refer to a TM, we assume it to be of the above kind.

Definition 3.1 A *configuration of M of size n* is a partial ID of M on the input of size n. It is written as a 5-tuple $\langle st, in, out, wk, tape\rangle$ where st denotes the state of M; in, out and wk denote respectively the input head, output head and work tape head positions; and $tape$ denotes the contents of the work tape. We refer to the starting configuration of M of size n as C^n_{init} (C^n_{init} may be taken to be $\langle q_0, 1, 1, 1, \mathbf{1}^{\lceil \log n\rceil}\rangle$ where q_0 is the start state) and a final configuration as C^n_{final} (there may be more than one final configurations).

Let $config(M, n)$ denote the set of all configurations of M of size n. The number of such configurations is bounded by a polynomial in n as M is a logspace TM. Let $\|config(M, n)\| \leq q_M(n)$ for some polynomial q_M.

Definition 3.2 A *computation sequence* of M on input x, $|x| = n$, is a sequence of configurations from $config(M, n)$, $C_1, C_2, \ldots, C_m$, such that $C_1 = C^n_{init}$, $C_m = C^n_{final}$ and for every i, $1 \leq i < m$, M moves from C_i to C_{i+1} in a single step when the input is x.

We say that a configuration of $config(M, n)$ is at *level i* if the value of the input head position in the configuration is i.

Definition 3.3 For any pair of configurations C and D of $config(M, n)$ at levels i and j respectively with $i < j$, and for any string z, we say that C *goes to D on z*, and write it as $C \xrightarrow{z} D$, if (1) $|z| = j - i$, and (2) TM M, started on configuration C reaches the configuration D when the string z is written on the bit positions i through $j - 1$ of the input. It is possible that C goes to D on more than one string. We write $C \xrightarrow{I} D$ when C goes to D on every string in the set I.

For any b and k, $k \leq n/b$, we divide every input string of length n into $k + 1$ disjoint adjacent blocks with the first k blocks containing b consecutive bits. We shall be interested in the behavior of M on the boundaries of these blocks (the reason for this would become clear later). Since M is a 1-L TM, its input head can enter any such block only from the left. Consider the computation sequence of M on some input x of length n. In this sequence, for $i > 1$, a configuration from level $(i - 1)b + 1$ such that the previous configuration is from level $(i - 1)b$ denotes the entry of M into the i^{th} block from $(i - 1)^{th}$ block. We call it the *entry* configuration of M into i^{th} block. For the first block, the entry configuration is the starting configuration.

Definition 3.4 A (k, b)-*block restricted computation sequence* of M on input x is the sequence obtained from the computation sequence of M on x by deleting all the configurations, except for the final one, that are not entry configurations of M into any of the $k + 1$ blocks.

It is possible that several strings of length n have the same (k,b)-block restricted computation sequence. Let *Seq* be a (k,b)-block restricted computation sequence on some input x, with $Seq = C_1, \ldots, C_{k+2}$. For $1 \leq i \leq k+1$, denote by $Str(Seq,i)$ the set of strings on which configuration C_i goes to C_{i+1}, i.e., $C_i \stackrel{str(Seq,i)}{\longrightarrow} C_{i+1}$. Then we have,

Proposition 3.5 *Seq is the (k,b)-block restricted computation sequence of M on any string in the set*

$$Str(Seq) = Str(Seq,1) \cdot Str(Seq,2) \cdot \cdots \cdot Str(Seq,k+1).$$

Here '$\cdot$' is the concatenation operation between sets, i.e., $S_1 \cdot S_2 = \{xy \mid x \in S_1 \wedge y \in S_2\}$.

Now we prove a crucial property of the block restricted computation sequences of 1-L TMs. This property—and its versions for other classes of TMs—will form the basis of all the collapsing results in the paper. The intuition behind the property is very simple. Consider computation sequences of a 1-L TM on different strings of size n. Each such sequence will begin with the configuration $C_1 = C^n_{init}$ and will contain configurations from the set $config(M,n)$. Since $\|config(M,n)\| \leq q_M(n)$, by Pigeon–Hole principle, there are two strings of length $b = \lceil \log q_M(n) \rceil + 1$ and a configuration $C_2 \in config(M,n)$ such that on both the strings C^n_{init} goes to C_2. Again, there are two more strings of length b and a configuration C_3 such that on both these strings C_2 goes to C_3. Continuing this way, one can construct a (k,b)-block restricted computation sequence *Seq* with $k \leq n/b$ such that $\|Str(Seq,i)\| \geq 2$ for each $1 \leq i \leq k$. The crucial property of strings in the set $Str(Seq)$ is that the TM M, when working on these strings, 'forgets' the input very quickly—for any $1 \leq i \leq k$, it can not distinguish between the strings in $Str(Seq,i)$ that are written in the i^{th} block, as long as its input head is not scanning the block. We exploit this property to make the function computed by M to be one-one and size-increasing on these strings. The following lemma states the above property formally and in a way that will be useful to us.

Let $b(n) = \lceil \log q_M(n^2) \rceil + 1$ and n_0 be the smallest number such that for any $n \geq n_0$, $n^2 \geq 4.b(n).n + b(n) + 1$.

Lemma 3.6 *Let M be a 1-L TM. There is a logspace procedure that, on input $\mathbf{1}^n$, for $n \geq n_0$, computes (1) a $(2n, 2b(n))$-block restricted computation sequence $Seq = C_1, C_2, \ldots, C_{2n+2}$ of M on strings of size n^2; and (2) $2n$ sets $I_1, \ldots, I_{2n}$ satisfying the following conditions.*

1. *For every i, $1 \leq i \leq 2n$, $\|I_i\| = 2$, $I_i = \{v_i v_i, w_i v_i\}$ and $|v_i| = |w_i| = b(n)$.*
2. *$C_1 \xrightarrow{I_1} C_2 \xrightarrow{I_2} C_3 \cdots \xrightarrow{I_{2n}} C_{2n+1} \stackrel{\mathbf{01}^{b(n)}\mathbf{0}^r}{\longrightarrow} C_{2n+2}$, where $r = n^2 - 4.b(n).n - b(n) - 1$.*

Proof. We give below a logspace procedure for computing the sequence *Seq* and the sets I_i and show that they satisfy all the conditions.

We shall proceed with the construction of *Seq* in stages such that after j^{th} stage, $1 \leq j \leq 2n+1$, C_j is the entry configuration of M into j^{th} block after reading any string from the set $I_1 \cdot \cdots \cdot I_{j-1}$.

Stage 1 : $C_1 = C^{n^2}_{init}$.

Stage j, $1 < j \leq 2n+1$: Find the smallest configuration C and the smallest two strings v and w of length $b(n)$ with $v > w$, such that $C_{j-1} \stackrel{\{vv,wv\}}{\longrightarrow} C$. Let $C_j = C$, $I_j = \{vv, wv\}$ and goto next stage.

Stage $2n+2$: Let C_{2n+2} be the final configuration of the TM such that $C_{2n+1} \stackrel{\mathbf{01}^{b(n)}\mathbf{0}^r}{\longrightarrow} C_{2n+2}$.

The above procedure is clearly computable within logspace as $b(n) = \lceil \log q_M(n^2) \rceil + 1 = O(\log n)$. We now show that two strings vv and wv such that $C_{j-1} \stackrel{\{vv,wv\}}{\longrightarrow} C_j$ will exist for every j. Since $|v| = |w| = b(n)$, there are a total of $2^{b(n)}$ such strings. Since there are at most $q_M(n^2)$ configurations in $config(M, n^2)$, there are at least $2^{b(n)}/q_M(n^2) \geq 2$ such strings, say v and w, such that $C_{j-1} \stackrel{v}{\longrightarrow} D$ and $C_{j-1} \stackrel{w}{\longrightarrow} D$ for some configuration D. Now, M will enter the j^{th} block in the same configuration on both vv and wv. This completes the proof. ■

We shall refer to the strings in the set

$$I_1 \cdot I_2 \cdot \cdots \cdot I_{2n} \cdot \{\mathbf{01}^{b(n)}\mathbf{0}^r\},$$

I_1, I_2, ..., I_{2n} as defined in the lemma above, as the *proper strings of size* n^2.

4 1-L reductions

In this section, we consider complete degrees under 1-L reductions. The isomorphism question for such degrees (and for ones complete under 1-NL reductions) has been considered earlier too. Allender [4] showed that $\leq_m^{1-L}$-complete sets for PSPACE and EXP are p-isomorphic; Ganesan and Homer [11] showed the same result for the class NEXP; Hemachandra and Hoene [15] showed that $\leq_m^{1-L}$- (or, $\leq_m^{1-NL}$-) complete sets for non-deterministic space classes above NLOG are also $\leq_{1,li}^{1-L}$- (resp., $\leq_{1,li}^{1-NL}$-) complete and hence NLOG-isomorphic; Hoene and Burtschick [8] showed that $\leq_m^{1-L}$-complete sets for PSPACE are not 1-L-isomorphic; and finally Agrawal and Biswas [1] showed that $\leq_m^{1-L}$-complete sets for classes closed under logspace reductions are p-isomorphic. Our results, in this and the next section, generalize all the previous ones, both for 1-L and 1-NL reductions.

The section is divided into three subsections. In the first one, we prove our main result, viz., that for any class closed under logspace reductions, $\leq_m^{1-L}$-complete degree collapses to $\leq_{1,qli,i}^{1-L}$-complete degree. In the next subsection we show that there exist 1-L-one-way functions and in the last one we show that the 1-L-isomorphism conjecture does not hold.

4.1 Collapse of complete degrees

In this subsection, we show that all $\leq_m^{1-L}$-hard sets for any class $\mathcal{C}$ closed under logspace reductions, are also $\leq_{1,qli,i}^{1-L}$-hard. The outline of the proof is as follows. Given a $\leq_m^{1-L}$-hard set A and a set $L \in \mathcal{C}$, we first define a 'coded' version of L, called $code(L)$, that belongs to $\mathcal{C}$ and is tailored to exploit the property of 1-L reductions shown in Lemma 3.6. Then, using the 1-L reduction, say f, of $code(L)$ to A, we define a 1-L reduction g_f of L to $code(L)$ such that the composition $f \circ g_f$ is one-one and length-increasing on all strings in $\Sigma_{=n}$ for every n that is an exact power of two and greater than some fixed constant.

Now, a $\leq_{1,qli,i}^{1-L}$-reduction of any set $B \in \mathcal{C}$ to A is obtained in the following way. First define a 'padded' version $\tilde{B}$ of B, $\tilde{B} \in \mathcal{C}$. Next, using the fact that $code(\tilde{B}) \in \mathcal{C}$, we fix a 1-L reduction f of $code(\tilde{B})$ to A. Define reduction g_f of $\tilde{B}$ to $code(\tilde{B})$ as above and finally define a 1-L reduction g_p of B to $\tilde{B}$, using f and g_f, that is one-one, quadratic length-increasing, and maps every string to a string whose size is an exact power of two. Now, it can be shown that the function $f \circ g_f \circ g_p$ is a 1-L reduction of B to A that is one-one, quadratic length-increasing and also 1-L-invertible.

We begin with defining the 'coded' version $code(L)$ of set L. Say that a string u is *duplicated* if $u = vv$ for some v. For any set L, define the set $code(L)$ as:

$$y \in code(L) \text{ iff } y = u_1 u_2 \cdots u_{2n} \mathbf{01}^b \mathbf{0}^r \text{ for } r \geq 0 \text{ satisfying the following conditions—}$$

1. $|u_1| = \cdots = |u_{2n}| = 2.b$,
2. Let $x = x[1] \cdots x[n]$ with $x[i] = \mathbf{1}$ if u_i is duplicated, $\mathbf{0}$ otherwise. Then, *either* for every i, $n+1 \leq i \leq 2n$, u_i is duplicated and $x \in L$, *or* there is a j, $n+1 \leq j \leq 2n$, such that for every i, $n+1 \leq i \neq j \leq 2n$, u_i is duplicated, u_j is not duplicated and $x[j-n] = \mathbf{1}$.

One may say that the first n blocks (of $2b$ bits each) of a string y in $code(L)$ code a string x of length n and the next n blocks represent n 'switches' such that either all the switches are 'on' and $x \in L$ or exactly one switch is 'off' and the corresponding bit of x is $\mathbf{1}$.

The following proposition is obvious.

Proposition 4.1 *For any L, $L \neq \emptyset, \Sigma^*$, $code(L) \leq_m^{log} L$.*

The following lemma captures the required property of the set $code(L)$.

Lemma 4.2 *Let f be a 1-L reduction, computed by TM M, of the set $code(L)$ to some set A. For any $n \geq n_0$ (n_0 as in Lemma 3.6), let $C_1 = C_{init}^{n^2}, C_2, \ldots, C_{2n+2} = C_{final}^{n^2}$ be the $(2n, 2b(n))$-block restricted computation sequence of the TM M on proper strings of size n^2 as computed in Lemma 3.6. Then, for each i, $1 \leq i \leq n$, the TM M, while moving from C_i to C_{i+1}, will output different equal length strings on reading strings v_iv_i and w_iv_i, where $v_iv_i, w_iv_i \in I_i$ (as defined in Lemma 3.6).*

Proof. Since configurations store the position of the output head as well, the output of M will be of the same length on any string while moving from C_i to C_{i+1}. Suppose that for some j, $1 \leq j \leq n$, the output of the TM M is the same while moving from C_j to C_{j+1} on reading strings v_jv_j and w_jv_j. Now consider string

$$\tilde{x} = v_1v_1 \cdots v_{j-1}v_{j-1}v_jv_jv_{j+1}v_{j+1} \cdots v_{n+j-1}v_{n+j-1}w_{n+j}v_{n+j}v_{n+j+1}v_{n+j+1} \cdots v_{2n}v_{2n}\mathbf{01}^{b(n)}\mathbf{0}^r$$

and

$$\tilde{y} = v_1v_1 \cdots v_{j-1}v_{j-1}w_jv_jv_{j+1}v_{j+1} \cdots v_{n+j-1}v_{n+j-1}w_{n+j}v_{n+j}v_{n+j+1}v_{n+j+1} \cdots v_{2n}v_{2n}\mathbf{01}^{b(n)}\mathbf{0}^r$$

Both $\tilde{x}$ and $\tilde{y}$ are proper strings of size n^2. Further, they differ only on the string written in the j^{th} block. Therefore, by our assumption, the output of M will be identical on both of them. However, by the definition of $code(L)$, $\tilde{x} \in code(L)$ and $\tilde{y} \notin code(L)$. This contradicts the fact that M is a reduction of $code(L)$ to A. Therefore, the output of M is indeed different. ■

The above lemma guarantees that if f is a 1-L reduction of $code(L)$ to A then it will be one-one on proper strings of size n^2 that differ only on the first n blocks, for all $n \geq n_0$. Now we compose f with a reduction of L to $code(L)$ that maps strings of size $n = 2^k$ for $k > 0$, to such proper strings of size n^2. Let M be a 1-L TM computing function f. Define function g_f as computed by the following procedure.

> On input x, let $n = 2^{\lceil \log |x| \rceil}$. If $n < n_0$ then output $x(1)\mathbf{1} \cdots x(|x|)\mathbf{1}\mathbf{1}^{2.|x|}\mathbf{01}$ and halt. Otherwise, for each $1 \leq i \leq 2.|x|$, do the following.
>
> Compute the set $I_i = \{v_iv_i, w_iv_i\}$ as in the Lemma 3.6. If $i \leq |x|$, output v_iv_i if $x(i) = \mathbf{1}$, w_iv_i otherwise. If $i > |x|$ then output v_iv_i.
>
> Finally, output $\mathbf{01}^{b(n)}\mathbf{0}^r$ where $r = n^2 - 4.b(n).|x| - b(n) - 1$ and halt.

Lemma 4.3 *g_f is a 1-1, size-increasing, 1-L reduction of L to $code(L)$.*

Proof. That g_f is logspace computable follows from Lemma 3.6, and the fact that $n = 2^{\lceil \log |x| \rceil}$ can be easily computed from $\mathbf{1}^{\lceil \log |x| \rceil}$. The function can be computed by a 1-L TM as the input needs to be scanned only once—bit $x[i]$ is needed only at the time of outputting string $v_i v_i$ or $w_i v_i$. By the construction it is obvious that g_f is 1-1 and size-increasing function. It is a reduction of L to $code(L)$ as the 'switches' in its output are always 'on' and it codes string x. ■

One can see that function g_f maps all strings of size $n \geq n_0$ and $n = 2^k$ for some $k \geq 0$, to a proper string of size n^2. The following lemma shows that it forces f to be one-one and honest on these proper strings.

Lemma 4.4 *Let f be a 1-L reduction, computed by TM M, of the set $code(L)$ to A. For every n, $n = 2^k \geq n_0$ for some $k \geq 0$, and for every x and y, $x \neq y$, $|x| = |y| = n = 2^k$, $f(g_f(x)) \neq f(g_f(y))$ and $|f(g_f(x))| \geq |x|$.*

Proof. For any x and y with $|x| = |y| = n = 2^k$ for some $k \geq 0$, and $n \geq n_0$, $g_f(x)$ and $g_f(y)$ are proper strings of size n^2 with the last $n+1$ blocks having identical strings. Further, if the i^{th} bit of x is different from that of y then the string in the i^{th} block of $g_f(x)$ would be different from that of $g_f(y)$. By Lemma 4.2, it follows that the output of M while scanning i^{th} block of these strings would be different. Therefore, $f(g_f(x)) \neq f(g_f(y))$. Since M will end up in the same final configuration on both x and y, we have $|f(g_f(x))| = |f(g_f(y))|$. And since there are 2^n different strings of size n, $|f(g_f(x))| \geq n = |x|$ as otherwise two such strings will be mapped to same string. ■

Now, we prove the main result of this section.

Theorem 4.5 *For any class $\mathcal{C}$ closed under logspace reductions, every $\leq_m^{1-L}$-hard set for $\mathcal{C}$ is also $\leq_{1,qli,i}^{1-L}$-hard.*

Proof. Let A be a $\leq_m^{1-L}$-hard set for $\mathcal{C}$ and $L \in \mathcal{C}$. Using Lemma 4.4 we can only get a reduction of L to A that is one-one and size-increasing on strings in $\Sigma_{=2^k}$ for large enough k. To get a reduction that is one-one everywhere, we first pad the strings to make their length an exact power of two. Define set $\tilde{L}$ as:

$$\tilde{L} = \{\mathbf{0}^k \mathbf{1} x \mathbf{1} \mathbf{0}^j \mid k, j \geq 0 \wedge x \in L\}$$

It is easy to see that that $\tilde{L} \leq_m^{log} L$ and therefore, $\tilde{L} \in \mathcal{C}$. Therefore, by Proposition 4.1, $code(\tilde{L}) \in \mathcal{C}$. Let f be a 1-L reduction, computed by TM M, of $code(\tilde{L})$ to A. Let $q_M(n) \leq (2n)^c$ for some constant $c > 0$ (where q_M bounds the number of configurations in $config(M, n)$, which, in turn, bounds the running time of TM computing f on input of size n). Define function r as: $r(0) = \max(n_0, 2)$ (n_0 as in Lemma 3.6) and $r(l) = 2c.r(l-1) + c$ for $l > 0$. Define function g_p, reducing L to $\tilde{L}$ as: $g_p(x) = \mathbf{0}^k \mathbf{1} x \mathbf{1} \mathbf{0}^j$ where $k = 2^{\lceil \log |x| \rceil}$, $j = 2^{2.r(l)} - 2^{\lceil \log |x| \rceil} - |x| - 2$, $l = \min_m(2^{r(m)} \geq |x|)$. Therefore, $|g_p(x)| = 2^{2.r(l)}$. Function g_p is a 1-L function—on input x, compute $k = 2^{\lceil \log |x| \rceil}$ and output $\mathbf{0}^k \mathbf{1} x$. Now start from $r(0)$ and calculate $r(1)$, $r(2)$, ... till an $r(m)$ is obtained with $r(m) \geq \lceil \log |x| \rceil$; compute $j = 2^{2.r(l)} - 2^{\lceil \log |x| \rceil} - |x| - 2$ and output $\mathbf{1}\mathbf{0}^j$. This function maps strings of length between $2^{r(l)} + 1$ and $2^{r(l+1)}$ to strings of length $2^{2.r(l+1)}$ and therefore, is quadratic length-increasing. Note that in the computation of $g_p(x)$, the 1-L TM knows only the number $2^{\lceil \log |x| \rceil}$ before scanning x and therefore to make the output length a power of two, it needs to pad $\mathbf{0}$'s at the end as well. Of course, this could have been done by padding $\mathbf{0}$'s *only* at the end, however, to strengthen the isomorphism between $\leq_m^{1-L}$-complete sets, it is necessary to pad $2^{\lceil \log |x| \rceil}$ many $\mathbf{0}$'s at the beginning (see Corollary 4.9 below).

Let function g_f be the reduction of $\tilde{L}$ to $code(\tilde{L})$ as defined above and $h \stackrel{\text{def}}{=} f \circ g_f \circ g_p$. The following claims show that h is the required reduction of L to A.

Claim 4.5.1 *$L \leq_m^{1\text{-}L} A$ via h.*

Proof of Claim 4.5.1. That h is a reduction of L to A follows immediately. To see that h is a 1-L function, we first note that functions g_p, g_f, f are all 1-L functions. Further, we have, $|g_p(x)| = 2^{2.r(l)}$ where l is the smallest number such that $r(l) \geq \lceil \log |x| \rceil$ and $|g_f(g_p(x))| = 2^{4.r(l)}$. These numbers can be computed within logspace using $\mathbf{1}^{\lceil \log |x| \rceil}$ and without scanning the input. Therefore, a 1-L TM for computing h is—on input x, compute $2^{2.r(l)}$ and $2^{4.r(l)}$; run 1-L TMs for g_p, g_f and f by interleaving their computation with $\mathbf{1}^{2.r(l)}$ and $\mathbf{1}^{4.r(l)}$ written on the work tapes of TMs computing g_f and f respectively. □

Claim 4.5.2 *h is quadratic length-increasing.*

Proof of Claim 4.5.2. For every x, $|g_p(x)| \geq 2^{r(0)} \geq n_0$ and further, g_p is quadratic length-increasing. Now it follows from Lemma 4.4 that $|f(g_f(g_p(x)))| \geq |g_p(x)| \geq |x|^2$. □

Claim 4.5.3 *h is one-one.*

Proof of Claim 4.5.3. For any x and y, consider two cases.

Case 1: $|g_p(x)| = |g_p(y)|$. By Lemma 4.4, and since $|g_p(x)| = 2^k$ for some $k \geq 0$, it follows that $h(x) \neq h(y)$.

Case 2: $|g_p(x)| < |g_p(y)|$. Let $|g_p(x)| = 2^{2.r(m)}$ and $|g_p(y)| = 2^{2.r(n)}$ with $n > m$. So, $|h(x)| = |f(g_f(g_p(x)))| \leq q_M(2^{4.r(m)})$ (by the bounds on the output lengths of f, g_f, and g_p) $\leq 2^{4c.r(m)+c}$ (as $q_M(n) \leq (2n)^c$) $\leq 2^{2.r(m+1)-c}$ (by the definition of r) $\leq 2^{2.r(n)-c} \leq 1/2 * |g_p(y)| \leq 1/2 * |f(g_f(g_p(y)))|$ (by Lemma 4.4) $= |h(y)|$. It follows that $h(x) \neq h(y)$.

□

Claim 4.5.4 *h is 1-L-invertible.*

Proof of Claim 4.5.4. We define a 1-L TM M' that, on input x, outputs $h^{-1}(x)$ and halts in an accepting state if the inverse exists, otherwise halts in a rejecting state (this will be useful in computing the isomorphism, see Corollary 4.9 below). It will be a composition of the following two 1-L TMs.

TM $\mathbf{M_1}$: This TM computes the function $(f \circ g_f)^{-1}$ correctly on the range of the function h. The range of function g_p has strings of length $2^{2.r(l)}$ only. So, on input z, M_1 executes the following procedure. Compute an l such that, letting $n = 2^{2.r(l)}$, if $w = f(g_f(\mathbf{1}^n))$ then $\mathbf{1}^{\lceil \log |w| \rceil} = \mathbf{1}^{\lceil \log |z| \rceil}$. There will be *at most* one such l since $|h(x)| \leq 1/2 * |h(y)|$ if $|g_p(x)| < |g_p(y)|$ (from Claim 4.5.3). If there is no such l then output $\mathbf{0}$ and halt in a rejecting state. Otherwise, do the following. We know that function g_f maps strings of size n to proper strings of size n^2 that have the same $(2n, 2b(n))$-block restricted computation sequence, say, $C_1, \ldots, C_{2n+2}$. Compute the set $I_1 = \{v_1v_1, w_1v_1\}$ for this sequence and then the output of M on both these strings while it moves from C_1 to C_2. By Lemma 4.2, these two outputs will be of the same length and different. Check if one of these is a prefix of z. If not then output $\mathbf{0}$ and halt in a rejecting state. Otherwise, output $\mathbf{1}$ or $\mathbf{0}$ respectively if the output on v_1v_1 and w_1v_1 matches. Repeat this process for every I_i, $1 < i \leq n$. Now, check if the rest of the input matches the output of M on string $v_{n+1}v_{n+1} \cdots v_{2n}v_{2n}\mathbf{01}^{b(n)}\mathbf{0}^r$ while moving from C_{n+1} to C_{2n+2}. Reject if not, accept otherwise.

TM $\mathbf{M_2}$: This TM computes the function g_p^{-1} in the following way. On input y, output x if $y = \mathbf{0}^k\mathbf{1}x\mathbf{10}^j$ for some $k, j \geq 0$ (a 1-L TM can extract x out of y by dropping all leading $\mathbf{0}$'s and the first $\mathbf{1}$ of y; then outputting x using the following strategy: the moment a $\mathbf{1}$ is encountered count the number of contiguous zeroes immediately after it, say i, and output $\mathbf{10}^i$ only if there is more input) and accept if $|y| = 2^{2r(l)}$, $2^{r(l-1)} < |x| \leq 2^{r(l)}$ and $k = 2^{\lceil \log |x| \rceil}$ for some l. Reject otherwise.

TM M' will, on input z, simulate TM M_1 on z and simulate TM M_2 on M_1's output simultaneously. It outputs the output of M_2, accepts if both the TMs accept, rejects otherwise. Since both M_1 and M_2 are 1-L TMs and M_2 does not require the length of the input, M' is also a 1-L TM. □

The above claims show that $L \leq^{1-L}_{1,qli,i} A$ via h. Since L was an arbitrary set of $\mathcal{C}$, and A an arbitrary $\leq^{1-L}_m$-hard set of $\mathcal{C}$, the theorem follows. ■

Corollary 4.6 *For any class $\mathcal{C}$ closed under logspace reductions, $\leq^{1-L}_m$-complete degree for $\mathcal{C}$ collapses to a $\leq^{1-L}_{1,qli,i}$-complete degree.*

Corollary 4.7 *For any class $\mathcal{C}$ closed under logspace reductions, if A is a $\leq^{1-L}_m$-hard set for $\mathcal{C}$ and t is a 1-1, 1-L function then $t(A)$ is also $\leq^{1-L}_m$-hard.*

Proof. As 1-L functions are not closed under composition in general, it is conceivable that $t(A)$ is not $\leq^{1-L}_m$-hard for $\mathcal{C}$ for some $\leq^{1-L}_m$-hard set A and 1-1, 1-L function t. However, this is not possible due to the properties of the constructed 1-1, quadratic length-increasing, 1-L-invertible, 1-L reduction, h, of the set L to A in the proof of above theorem. This reduction has the following easily verifiable property: for all x, $|h(x)| = |h(\mathbf{1}^{2^{\lceil \log |x| \rceil}})|$. Therefore, function $t \circ h$ can be computed by a 1-L TM that on input x, first computes $|h(x)|$ using the above equality and then runs the 1-L TMs of t and h in an interleaved fashion. ■

Corollary 4.8 *The 1-L-encrypted complete set conjecture is false.*

It trivially follows, from the above theorem and a result in [13], that all $\leq^{1-L}_m$-complete sets are logspace-isomorphic. However, one can do better than this and show that these sets are *2-L-isomorphic*, where 2-L functions are computed by logspace TMs that are allowed two left-to-right scans of the input.

Corollary 4.9 *For any class $\mathcal{C}$ closed under logspace reductions, all $\leq^{1-L}_m$-complete sets for $\mathcal{C}$ are 2-L-isomorphic.*

Proof. In [13], it was shown that all sets complete under $\leq^{log}_{1,qli,i}$-reductions are logspace-isomorphic. We shall proceed along exactly the same lines. The construction in [13] is inspired from the one for polynomial-time reducibilities given in [7] which, in turn, is essentially the Cantor-Schröder-Bernstein construction of the isomorphism between two sets. We first give the construction of [13] and then mention the modifications needed to make it work for 1-L reductions.

Let A and B be two sets with $A \leq^{log}_{1,qli,i} B$ via u and $B \leq^{log}_{1,qli,i} A$ via v. For any x, define the *inverse chain* at x to be the sequence x_0, x_1, x_2, …, x_l where $x_0 = x$, $x_i = v^{-1}(x_{i-1})$ if i is odd, $u^{-1}(x_{i-1})$ if i is even, for $1 \leq i \leq l$ and $v^{-1}(x_l)$ ($u^{-1}(x_l)$) does not exist if l is even (odd). Number l is called the *length* of the chain.

Define function t as: $t(x) = u(x)$ if the length of the inverse chain at x is even; $v^{-1}(x)$ otherwise. It is easy to see that t is an isomorphism between A and B. Since both u and v are length squaring and their inverses are computable in logspace, it follows that the length of the inverse chain at x can be computed in logspace by an interleaved simulation of all inverses, and therefore, t can be computed in logspace.

In our case, u and v are 1-1, length-squaring, 1-L-invertible, 1-L functions. Define t as before. We show that the length of the inverse chain at x can be computed in a single pass over the input and therefore t can be computed in two passes (use the second pass to compute $u(x)$ or $v^{-1}(x)$ depending on whether the length is even or odd). In a similar manner, t^{-1} can also be computed.

We have, by the proof of Theorem 4.5, two 1-L TMs M_u and M_v that, on input x, output $u^{-1}(x)$ and $v^{-1}(x)$ respectively if they exists and halt in accepting state. Otherwise the TMs halt in rejecting state. The following procedure to compute the length of the chain suggests itself—on input x, start the computation of M_v on x; the moment some output appears, start the computation of M_u on it; and so on for all the intermediate strings. If at any point, some computation ends in rejecting state (the inverse is not defined), abort all the computations started after it. Eventually, first k computations will end in accepting state and $(k+1)^{th}$ in rejecting state for some $k \geq 0$ and then k will be the length of the chain. However, this procedure does not work as to be able to compute $v^{-1}(x_i)$ or $u^{-1}(x_i)$ in a single pass at any intermediate string x_i, one needs $\mathbf{1}^{\lceil \log |x_i| \rceil}$ also written on the tape. It is for this reason that the function g_p was defined in a somewhat complicated way in the proof of Theorem 4.5. Assume that $u = f_1 \circ g_{f_1} \circ g_{p_1}$ and $v = f_2 \circ g_{f_2} \circ g_{p_2}$ as given in the proof. The above procedure is modified in the following way.

On input x, start the computation of $(f_2 \circ g_{f_2})^{-1}$ on x and count the number of leading zeros. If the inverse exists, there must be exactly $2^{\lceil \log |x_1| \rceil}$ such zeroes. Thus, we get to know $\lceil \log |x_1| \rceil$. Continue with the computation, it must now output x_1. Simultaneously, start the computation of $u^{-1}(x_1)$; it can be computed properly as we know $\lceil \log |x_1| \rceil$. In this way, we can know the length of all intermediate strings before their computation begins. The rest of the procedure remains the same. Thus the entire computation can be performed in a single pass on the input. ■

In [14], it was shown that all natural NP-complete sets are $\leq_m^{1\text{-}L}$-complete as well. It follows that all these sets are 2-L-isomorphic. This is an improvement on the result that they are all logspace-isomorphic [13] while it does not compare with a recent result that they are first-order-isomorphic [3].

4.2 Existence of one-way functions

It is fairly straightforward to see that there are 1-L-one-way functions. Define $f(bx) = xb$ where $b \in \Sigma$. Clearly f is a 1-L function. Moreover, it is a 1-1 and onto function. Now we show that f^{-1} is not a 1-L function.

Proposition 4.10 *Function h, $h(xb) = bx$, $b \in \Sigma$, is not a 1-L function.*

Proof Sketch. Any DTM computing h must read the last bit of the input before outputting any bit. Any such DTM working within logspace will 'forget' most of the string x, for large enough x, by the time it reaches the end of the input and therefore to output $h(xb)$ correctly, it must scan the input once more. ■

4.3 Failure of the isomorphism conjecture

Can we say that $\leq_m^{1-L}$-complete degrees collapse completely? I.e., are all $\leq_m^{1-L}$-complete sets 1-L-isomorphic? The answer is no. It was shown in [8] that $\leq_m^{1-L}$-complete degree for PSPACE does not collapse to a single 1-L-isomorphic degree. We generalize this result for all classes closed under logspace reductions. Our proof is considerably simpler as well.

Theorem 4.11 *For any class $\mathcal{C}$ closed under logspace reductions, $\leq_m^{1-L}$-complete degree for $\mathcal{C}$, if it exists, does not collapse to a single 1-L-isomorphic degree.*

Proof. Assume that for some class $\mathcal{C}$, $\leq_m^{1-L}$-complete degree collapses to 1-L-isomorphic degree. Let L be a $\leq_m^{1-L}$-complete set for $\mathcal{C}$. Let Xb and bX be the set of strings that are obtained by concatenating bit b, $b \in \Sigma$, at the end and beginning respectively, to each string of the set X. Now define, $L_1 = L\mathbf{1}$ and $L_2 = \mathbf{0}L \cup \mathbf{1}\Sigma^*$. Both these sets are $\leq_m^{1-L}$-complete for the class $\mathcal{C}$. Therefore, by our assumption, there is an isomorphism between L_1 and L_2 given by a 1-L function h, say. Since h is a reduction of L_1 to L_2, $h^{-1}(\mathbf{1}\Sigma^*) \subset L\mathbf{1}$. Since h is honest, there exists a polynomial p such that for every x, $p^{-1}(|x|) \leq |h(x)| \leq p(|x|)$. So, the set $h^{-1}(\mathbf{1}\Sigma_{=n})$ contains strings of at most $p(n)$ different sizes and since h is onto, there is an m, $m < p(n)$, such that $h^{-1}(\mathbf{1}\Sigma_{=n}) \cap \Sigma_{=m}\mathbf{1}$ contains at least $2^n/p(n)$ strings. Let $y\mathbf{1} \in h^{-1}(\mathbf{1}\Sigma_{=n}) \cap \Sigma_{=m}\mathbf{1}$ for such an m. Since h is a reduction of L_1 to L_2, $h(y\mathbf{0}) \in \mathbf{0}\Sigma^*$. Therefore, the 1-L TM M_h, computing h, before outputting any bit, must scan the whole input and check if the last bit is $\mathbf{0}$ or $\mathbf{1}$ for all such y's. Now there are $2^n/p(n)$ such y's. This means that when n is large enough so that $2^n/p(n)$ is greater than total number of configurations of M_h on input of size $m+1$ (which is bounded by a polynomial in n), M_h will end up in the same configuration after reading first m bits of two different such input strings $y_1\mathbf{1}$ and $y_2\mathbf{1}$ of size $m+1$. M_h will not have output anything by then and therefore its output on both the strings will be the same. This contradicts the fact that h is one-one. ∎

Corollary 4.12 *The 1-L-isomorphism conjecture if false.*

The above theorem, along with the Corollary 4.9, provides a tight upper and lower bound on the isomorphism of $\leq_m^{1-L}$-complete sets.

5 1-NL reductions

The failure of the 1-L-isomorphism conjecture is due to the inability of 1-L TMs to carry out the Cantor-Schröder-Bernstein kind of construction of the isomorphism as in [13]. Moreover, Theorem 4.11 tells us that a second scan of the input can not be avoided while computing the isomorphism. So now the question is—can one generalize 1-L reductions so that one can carry out the isomorphism construction within these reductions *and* the collapsing result still holds? That would provide us with an example of a reducibility for which the isomorphism conjecture is true. In this section, we answer this question affirmatively for 1-NL reductions.

The section is divided in two subsections. In the first subsection we prove a result similar to the Lemma 3.6 for 1-NL TMs, and in the next one we prove that $\leq_m^{1-NL}$-complete degrees collapse to 1-NL-isomorphic degrees for all classes closed under logspace reductions.

5.1 Computation sequences of 1-NL TMs

We begin with defining the notion of block restricted computation sequences of 1-NL TMs. For a 1-NL TM M, let $\mathit{config}(M, n)$ be the set of configurations on input strings of size n as before.

We also assume that $\|config(M,n)\| \leq q_M(n)$ for some polynomial q_M. One can define (k,b)-block restricted computation sequence of 1-NL TMs in a way similar to that of 1-L TMs. The difference being that (1) for configurations C and D we say $C \xrightarrow{z} D$ if the 1-NL TM moves from configuration C to D on *some* guess path when z is written in the appropriate bit positions of the input tape; and (2) the last configuration of the sequence, instead of a final configuration, is an *accepting* configuration. Due to non-determinism, there may be *several* (k,b)-block restricted computation sequences on a single string.

We now prove the analog of Lemma 3.6. Let $b(n) = \lceil 3.\log q_M(n^2) \rceil + \log n + 4$ (note that the block size here is somewhat larger than the one for 1-L TMs) and n_0 be the least number such that for all $n \geq n_0$, $n^2 \geq 4.b(n).n + b(n) + 1$.

Lemma 5.1 *Let M be a 1-NL TM. There is a non-deterministic logspace procedure that, on input $\mathbf{1}^n$, for $n \geq n_0$, computes (1) a $(2n, 2b(n))$-block restricted computation sequence $Seq = C_1$, C_2, ..., C_{2n+2} of M on strings of size n^2; and (2) $2n$ sets I_1, ..., I_{2n} satisfying the following conditions.*

1. *For every i, $1 \leq i \leq 2n$, $\|I_i\| = 2$, $I_i = \{v_i v_i, w_i v_i\}$ and $|v_i| = |w_i| = b(n)$.*
2. *$C_1 \xrightarrow{I_1} C_2 \xrightarrow{I_2} C_3 \cdots \xrightarrow{I_{2n}} C_{2n+1} \xrightarrow{\mathbf{01}^{b(n)}\mathbf{0}^r} C_{2n+2}$, where $r = n^2 - 4.b(n).n - b(n) - 1$.*

Proof. As in the proof of Lemma 3.6, we first give an NLOG procedure to compute the sequence *Seq* and sets I_1, ..., I_{2n} and then show that they satisfy all the conditions. Both the procedure and the proof of its correctness become more involved due to the presence of non-determinism.

We cannot use the procedure described in the proof of Lemma 3.6 here because it is possible that $C_1 \xrightarrow{\{vv,wv\}} C_2$ for some strings v and w but there is no accepting path from C_2. To avoid this problem, we first define the following sets.

$$
\begin{aligned}
L_0 &= \{(\mathbf{1}^n, C, D) \mid C, D \in config(M, n^2), \text{ and } D \text{ is an accepting configuration such that } C \xrightarrow{\mathbf{01}^{b(n)}\mathbf{0}^r} D \} \\
L_1 &= \{(\mathbf{1}^n, C, s, D) \mid C, D \in config(M, n^2) \text{ and } C \xrightarrow{s} D\} \\
L_2 &= \{(\mathbf{1}^n, C, D) \mid C \text{ and } D \text{ are entry configurations for two successive blocks, and the number of strings } s \text{ such that } |s| = b(n) \text{ and } (\mathbf{1}^n, C, ss, D) \in L_1, \text{ is at least } 2.q_M(n^2) \} \\
L_3 &= \{(\mathbf{1}^n, C) \mid C = D_j \text{ is an entry configuration for the } j^{th} \text{ block, there exist } D_{j+1}, \ldots, D_{2n+2} \text{ such that } (\mathbf{1}^n, D_i, D_{i+1}) \in L_2 \text{ for } j \leq i \leq 2n, \text{ and } (\mathbf{1}^n, D_{2n+1}, D_{2n+2}) \in L_0 \}
\end{aligned}
$$

It is easy to see that all these sets are in NLOG. Now, to compute the sequence *Seq* we do the following stage-wise construction.

Stage 1 : $C_1 = C_{init}^{n^2}$.

Stage j, $1 < j \leq 2n+1$: Find the smallest C, C an entry configuration for the j^{th} block, such that $(\mathbf{1}^n, C_{j-1}, C) \in L_2$ and $(\mathbf{1}^n, C) \in L_3$. Now, find the smallest two strings v and w of length $b(n)$, $w < v$, such that $(\mathbf{1}^n, C_{j-1}, vv, C) \in L_1$ and $(\mathbf{1}^n, C_{j-1}, wv, C) \in L_1$. Let $C_j = C$, $I_j = \{vv, wv\}$ and goto next stage.

Stage $2n+2$: Find the smallest accepting configuration C_{2n+2} such that $(\mathbf{1}^n, C_{2n+1}, C_{2n+2}) \in L_0$.

Since NLOG is closed under complementation [16, 23], it follows that the above procedure is also computable within NLOG.

Now we show that the output of the procedure is as desired. Clearly, it is sufficient to show that the configuration C and strings v and w as computed in Stage j, $1 < j \leq 2n+1$, always exist. Suppose that $(\mathbf{1}^n, C_{init}^{n^2}) \in L_3$. Then by the definition of L_3, at every stage j, $1 < j \leq 2n+1$, a configuration $C_j = C$ can be found such that $(\mathbf{1}^n, C_{j-1}, C) \in L_2$ and $(\mathbf{1}^n, C) \in L_3$. By the definition of L_2 we know that there are at least $2.q_M(n^2)$ strings of the form uu with $|u| = b(n)$, such that $C_{j-1} \xrightarrow{uu} C_j$. Therefore, there must be at least $2.q_M(n^2)/q_M(n^2) = 2$ such strings, say vv and ww such that $C_{j-1} \xrightarrow{\{v,w\}} D$ and $D \xrightarrow{\{v,w\}} C_j$ for some configuration D. Thus, $C_{j-1} \xrightarrow{\{vv,wv\}} C_j$ as required.

So, all that we now need to show is that $(\mathbf{1}^n, C_{init}^{n^2}) \in L_3$. For $1 \leq i \leq 2n$, let X_i be the set of all $(2n, 2b(n))$-block restricted computation sequences D_1, D_2, ..., D_{2n+1}, D_{2n+2} on strings of size n^2 such that (1) $D_{2n+1} \xrightarrow{\mathbf{01}^{b(n)}\mathbf{0}^r} D_{2n+2}$, and (2) $(\mathbf{1}^n, D_i, D_{i+1}) \notin L_2$.

Now, let

$$S = \{v_1 v_1 v_2 v_2 \cdots v_{2n} v_{2n} \mathbf{01}^{b(n)} \mathbf{0}^r \mid (\forall i)|v_i| = b(n)\}.$$

Clearly, $\|S\| = 2^{2nb(n)}$. How many strings in S have a $(2n, 2b(n))$-block restricted computation sequence belonging to the set X_i for some i? As there are less than $2q_M(n^2)$ strings of the form vv such that $D_i \xrightarrow{vv} D_{i+1}$ for D_i, D_{i+1} belonging to any computation sequence in X_i (by the definition of set L_2), and there are at most $(q_M(n^2))^2$ possible choices for pair D_i, D_{i+1}, it follows that less than $(q_M(n^2))^2.2q_M(n^2).2^{(2n-1)b(n)}$ strings in S have a $(2n, 2b(n))$-block restricted computation sequence belonging to the set X_i. Therefore, less than $4n.(q_M(n^2))^3.2^{(2n-1)b(n)} \leq 2^{2nb(n)}$ strings in S have a $(2n, 2b(n))$-block restricted computation sequence belonging to the set $\bigcup_{1 \leq i \leq 2n} X_i$. This implies that there is at least one string in S whose $(2n, 2b(n))$-block restricted computation sequence does not belong to the set $\bigcup_{1 \leq i \leq 2n} X_i$. The existence of such a sequence proves that $(\mathbf{1}^n, C_{init}^{n^2}) \in L_3$. ■

The definition of *proper* strings remains the same.

5.2 Collapse of complete degrees

In this subsection we will prove that for every class closed under logspace reduction, 1-NL-isomorphism conjecture is true. Towards this, we first prove a collapse result similar to Theorem 4.5 for $\leq_m^{1-NL}$-complete degrees. The proof structure remains identical to the proof of Theorem 4.5. The definitions of the set $code(L)$ and function g_f remains the same except that the function f is a 1-NL reduction and to compute g_f the procedure given in the proof of Lemma 5.1 is used (with the larger block size $b(n)$ as given in the subsection 5.1). For these definitions, Proposition 4.1 and Lemmas 4.2, 4.3, 4.4 can be proved in the same manner as before with functions f and g_f being 1-NL reductions instead of 1-L reductions.

Theorem 5.2 *For any class $\mathcal{C}$ closed under logspace reductions, every $\leq_m^{1-NL}$-hard set for $\mathcal{C}$ is also $\leq_{1,qli,i}^{1-NL}$-hard.*

Proof Sketch. Let A be a $\leq_m^{1-NL}$-hard set for $\mathcal{C}$ and $L \in \mathcal{C}$. Define $\tilde{L}$ and reduction g_p as in the proof of Theorem 4.5. Let $code(\tilde{L}) \leq_m^{1-NL} A$ via f and $h = f \circ g_f \circ g_p$. That h is a one-one, length-squaring, 1-NL reduction of L to A can be derived in a manner identical to the one in the proof of Theorem 4.5. We now show that h is also 1-NL-invertible.

As in the proof of Claim 4.5.4, we define the inverting 1-NL TM M' as a composition of two 1-NL TMs M_1 and M_2. The definition of M_2 remains the same. TM M_1 also works in an identical manner except that it has to do some extra work when it is computing the output of M

(M computes f) on strings in the set $I_i = \{v_iv_i, w_iv_i\}$ while M moves from C_i to C_{i+1} for any i. The reason is that M, being a non-deterministic TM, may go to more that one configurations from C_i on reading some bit of a string u in the set I_i. And it is possible that from one of these several configurations M may never reach configuration C_{i+1}. Therefore, after computing all these possible configurations of M, M_1 has to choose the one from which M will reach C_{i+1} on string u. To identify such a configuration, M_1 uses the set L_1 defined in the proof of Lemma 5.1. Of course there may be more than one such configurations. M_1 can choose any one of them as M must output identically on all accepting paths. The rest of the procedure for M_1 remains the same. It is easy to verify that M' is a 1-NL TM computing the inverse of h. ■

Using the above Theorem and the construction of isomorphism as given in Corollary 4.9, the following follows easily.

Corollary 5.3 *For any class $\mathcal{C}$ closed under logspace reductions, $\leq_m^{1\text{–}NL}$-complete degree for $\mathcal{C}$ collapses to a single 1-NL-isomorphic degree.*

Proof Sketch. Take any two sets A and B that are $\leq_m^{1\text{–}NL}$-complete for $\mathcal{C}$. By Theorem 5.2, we have that they are reducible to each other via 1-1, length-squaring and 1-NL-invertible reductions. To construct the isomorphism, we use the same construction as in the proof of Corollary 4.9. The function t is defined as before, with functions u and v being 1-1, length-squaring, 1-NL-invertible, 1-NL reductions between A and B. Again, it is easy to see that t can be computed by a non-deterministic logspace TM with two scans of the input—first to compute the length of the inverse chain and second to output. Using non-determinism, one can combine these two scans in one in the following way. At the beginning of the computation on x, *guess* the length of the chain. Now, verify the guess while simultaneously outputting $u(x)$ or $v^{-1}(x)$ based on the guessed length. If a guess turns out to be wrong, abort the computation on that path. Thus, in a single scan, one gets the output. Similarly, t^{-1} can also be computed. ■

Corollary 5.4 *The 1-NL-isomorphism conjecture is true.*

This is the first non-trivial example of a reducibility for which the isomorphism conjecture holds.

6 1-omL reductions

In the previous two sections, we have seen examples of reducibilities for which the encrypted complete set conjecture fails while the isomorphism conjecture fails for one and holds for the other. In this and next section, we study two reducibilities, viz., 1-omL and c-L, such that the encrypted complete set conjecture fails for one and holds for the other. These two reducibilities also exhibit an interesting property—the complete degrees under both are the *same* even though the class of 1-omL functions is larger than the class of c-L functions. There is another reason for investigating the complete degree under 1-omL reductions—these reductions are powerful enough to include several *p-one-way* functions. We first identify such p-one-way functions.

Recall that p-one-way functions exist if and only if P $\neq$ UP [18, 12]. Assuming P $\neq$ UP, one can construct the following 'natural' class of p-one-way functions.

Take any set $A \in$ UP $-$ P. Let M be a polynomial-time NDTM accepting A. A typical encoding of an accepting computation of M on input x is of the form $ID_1ID_2\cdots ID_m$ where $m = p(|x|)$ for some polynomial p, $|ID_i| = m$, ID_1 and ID_m are the starting and accepting IDs respectively of M, and for every i, $1 \leq i < m$, M moves from ID_i to ID_{i+1} in a single step. Define function f_M as: $f_M(x) = \mathbf{1}y$ if x is the above encoding of the accepting computation of M on y; $\mathbf{0}x$ otherwise. Now we have,

Proposition 6.1 *If* P $\neq$ UP*, then f_M is a 1-omL function with f_M^{-1} not computable in polynomial-time.*

Proof Sketch. It is easy to see that f_M is not p-invertible. An accepting computation x of M on input y is of the form $ID_1ID_2\cdots ID_m$, $m = p(|y|)$, and each ID_i itself is of length m. f_M is a 1-omL function since it can be recognized by an 1-omL TM with four input heads—calculate the length of the input using the first head, check if it is equal to $[p(n)]^2$ for some n. If not then output $\mathbf{0}x$. Otherwise, compare successive IDs using the next two heads to ensure that they form a valid computation, the first ID is initial ID and the final ID is an accepting one; if it is indeed an accepting computation then using the last head extract y and output $\mathbf{1}y$ else output $\mathbf{0}x$. In all these calculations, obliviousness of the input heads can be easily maintained. ■

To obtain the collapsing result for 1-omL reductions, we follow the same path as for 1-L and 1-NL reductions. However, there are several obstacles that we have to overcome before we can obtain a lemma similar to Lemma 3.6 for computation sequences of 1-omL TMs. We can assume, without loss of generality, that (1) the j^{th} input head of a 1-omL TM is never ahead of the i^{th} one, for $j > i$, (2) on each move of the TM, at most one of the heads move, and (3) the TM halts only when all its heads have scanned the entire input. A configuration of a 1-omL TM stores the position of all the heads along with the other information. We let $\mathit{config}(M, n)$ denote the set of configurations of a 1-omL TM on input strings of size n with $\|\mathit{config}(M, n)\| \leq q_M(n)$ as before.

The first problem we encounter is in finding an appropriate definition of a configuration *going to* some other configuration on a string. As a configuration for a 1-omL TM stores the positions of all the input heads, the TM may not read a contiguous block of bits from the input tape while moving from a configuration C to another configuration D. In fact, we do not use this notion at all. Instead we define later on a somewhat similar notion for the kind of strings we are interested in.

A bigger problem awaits us in the definition of proper strings. Recall that the most crucial step in the proofs of Theorems 4.5 and 5.2 was the identification of a subset of strings (referred as proper strings) on which the 1-L (or, 1-NL) TM works in a 'forgetful' manner—for the first $2n$ blocks of proper strings of size n^2, we have two possible choices of strings and for any of the first $2n$ blocks, the 1-L (or, 1-NL) TM does not remember which of these two strings is written in the block while it is scanning a different block. Moreover, the two possible choices of strings for these blocks have the form vv and wv. To make the TM act in a 'forgetful' manner—in other words, to end up in the same configuration on these two strings—we ensured that it ends up in the same configuration on scanning either of v and w. Both of these properties do not appear to hold for 1-omL TMs as they have multiple heads and therefore they can simultaneously scan two different blocks or two different places within a block.

To achieve this 'forgetfulness' property for 1-omL reductions, we change the definition of proper strings by introducing redundant blocks, i.e., blocks that do not have anything useful written on them. The proper strings will be of size n^{2^a}, instead of n^2, where a is the number of input heads. They will have $1/2*(8n)^{2^{a-1}}$ blocks of length $b(n)$ each (for a suitably defined $b(n)$) with $4n$ of them being labelled *live* and the rest *dead*. The live blocks will have two possible choices of strings as before, and the dead ones will just have a sequence of zeros written on it. There will be a further restriction: the $(2i-1)^{th}$ and $(2i)^{th}$ live blocks, for $1 \leq i \leq 2n$, will have the same set of two strings as their possible choices. These two blocks will be used to code a single bit (in the earlier two proofs, a single block was used for this purpose). The idea of introducing dead blocks is to ensure that at any time during the computation, the input heads of the TM are not scanning two different live blocks. And since we code all useful information in live blocks, the 'forgetfulness' property of the TM is restored.

By an *entry* configuration of an input head of M into a block we mean a configuration that denotes the entry by that head into the block. So, corresponding to each block, there will be a entry configurations. We define an *exit* configuration of an input head of M from a block to be the entry configuration of the head into the next block. We will consider computation sequences restricted only to entry and exit configurations of input heads into and from the live blocks. Define a (k, b)-*live-block restricted* computation sequence to be a computation sequence from which all the configurations, except the first and last one, that are not entry or exit configurations of input heads into/from live blocks are deleted.

Now we are ready to prove the analog of Lemma 3.6. Let $b(n) = \lceil 4a.\log q_M(n^{2^a})\rceil+2$, $m(k) = 1/2*(8n)^{2^{k-1}}$, and n_0 be the smallest number such that for all $n \geq n_0$, $n^{2^a} \geq m(a).b(n)+b(n)+1$.

Lemma 6.2 *Let M be a 1-omL TM. There is a logspace procedure that, on input $\mathbf{1}^n$, for $n \geq n_0$, computes (1) a $(4n, b(n))$-live-block restricted computation sequence Seq of M on strings of size n^{2^a}; (2) $m(a)$ sets $I_1, \ldots, I_{m(a)}$; and (3) a division of the input heads of M into d disjoint groups $H_1, H_2, \ldots, H_d$ satisfying the following conditions.*

1. *For every i, $1 \leq i \leq m(a)$, either $\|I_i\| = 2$, $I_i = \{v_i, w_i\}$, $\mathbf{0}^{b(n)} < w_i < v_i$, $|v_i| = |w_i| = b(n)$; or $I_i = \{\mathbf{0}^{b(n)}\}$.*

2. *There are exactly $4n$ I_is with cardinality two (corresponding to live blocks). Let these be $I_{i_1}, \ldots, I_{i_{4n}}$. Then for every j, $1 \leq j \leq 2n$, $I_{i_{2j-1}} = I_{i_{2j}}$.*

3. *On any input string of size n^{2^a}, for every i, $1 < i \leq d$, heads in the group H_i enter the first live block only after all the heads in the group H_{i-1} have exited from the last (i.e., $(4n)^{th}$) live block. Further, for every i, $1 \leq i \leq d$, and for every j, $1 < j \leq 4n$, a head in the group H_i enters the j^{th} live block only after all the heads in H_i have exited from the $(j-1)^{th}$ live block. (It follows from the above two conditions that at no time during the computation, input heads of M scan two different live blocks.)*

4. *Let $r = n^{2^a} - m(a).b(n) - b(n) - 1$. Then Seq is the $(4n, b(n))$-live-block restricted computation sequence of M on every string in the set*

$$S = I_1 \cdot I_2 \cdot \cdots \cdot I_{m(a)} \cdot \{\mathbf{0}\mathbf{1}^{b(n)}\mathbf{0}^r\}.$$

Proof. Note that the condition 3 of the lemma restricts the head movement in a much stronger way than just that they should not scan two different live blocks simultaneously. It is, as we shall see, required to make the proof go through. We proceed with the proof in two parts. First, we show that live blocks satisfying the condition 3 exist and then show that one can choose the assignments to the live blocks and a $(4n, b(n))$-live restricted computation sequence to satisfy the conditions 1, 2, and 4. As for the logspace procedure, it is indicated as we go along.

We now show the first part. We have to choose $4n$ live blocks in a way that condition 3 holds. Since the TM is oblivious, we can find out about its head movement on any string of size n^{2^a} by running it on the string $\mathbf{0}^{m(a).b(n)}\mathbf{0}\mathbf{1}^{b(n)}\mathbf{0}^r$ (the obliviousness of the TM is crucial to the proof). We divide the above string in $m(a)+1$ blocks with the first $m(a)$ blocks of size $b(n)$ each. The last block is of size $1 + b(n) + r$ and is labelled dead. Now we label all of first $m(a)$ blocks live, let $H_1 = \{1\}$ and proceed inductively. Suppose that the first c heads have been divided into groups $H_1, \ldots, H_r$ such that the condition 3 holds for these heads with $m(a+1-c)$ live blocks (instead of $4n$ live blocks). Divide the first $m(a-c) * (1 + m(a-c))$ live blocks into $m(a-c)$ groups of $1+m(a-c)$ successive live blocks each. Since $m(a-c)*(1+m(a-c)) \leq m(a+1-c)$, such a division is possible. Now for each of these groups, check whether by the time head c has entirely scanned all the blocks in the group, head $c+1$ entirely scans the first block of the group. If there is a group for which it is not true, then label the last $m(a-c)$ blocks of that

group live and the rest of the $m(a+1-c)-m(a-c)$ blocks as dead. Also, let $H_{r+1}=\{c+1\}$. On the other hand, if for every group the condition holds, then label the first block of each of $m(a-c)$ groups live and the rest as dead. Also, let $H_r = H_r \cup \{c+1\}$. In either case, we end up with $m(a-c)$ live blocks and the condition 3 holds for the first $c+1$ heads with $m(a-c)$ live blocks. Therefore, in the end, we will have $m(1)=4n$ live blocks with the condition 3 holding for all the heads. To decide, in logspace, if the given block is (eventually) live or not, we just need to note that at any time during the entire process above, all live blocks are equidistant. So a logspace TM can simulate the above process by storing the position of the first live block and the distance between two successive ones.

Now we proceed to show the second part. Since we have fixed the positions of live blocks, we can talk of $(4n, b(n))$-live-block restricted computation sequences. We give below, the construction of a $(4n, b(n))$-live-block restricted computation sequence *Seq* and sets I_1, I_2, ..., $I_{m(a)}$ satisfying the conditions 1, 2, and 4.

Let us assume—for the next two paragraphs—that the $(4n, b(n))$-live-block restricted computation sequence *Seq* as desired exists. We denote by C_i^j and D_i^j respectively the entry and exit configuration, in *Seq*, of the i^{th} input head into and from the j^{th} live block. Let the input heads be divided into disjoint groups H_1, ..., H_d as above with $H_k = \{t_k, t_k+1, \ldots, t_{k+1}-1\}$ for $1 \le k \le d$, $t_1 = 1$ and $t_{d+1} = a+1$. Since the condition 3 of the Lemma is satisfied, we have that in sequence *Seq*, for any $1 \le j \le 4n$, any $1 \le k \le d$, configurations $C_{t_k}^j$, ..., $C_{t_{k+1}-1}^j$, $D_{t_k}^j$, ..., $D_{t_{k+1}-1}^j$, form a consecutive subsequence (not necessarily in the given order) with $C_{t_k}^j$ and $D_{t_{k+1}-1}^j$ as the first and last configuration of the subsequence (ordering of the rest of the configurations is immaterial). We let $\tilde{C}(k,j)$ denote this subsequence. Therefore,

$$Seq = C_{init}, \tilde{C}(1,1), \ldots, \tilde{C}(1,4n), \tilde{C}(2,1), \ldots, \tilde{C}(2,4n), \ldots\ldots, \tilde{C}(d,1), \ldots, \tilde{C}(d,4n), C_{final}$$

where C_{init} and C_{final} are the initial and final configurations respectively. Strictly speaking, the above equality may not hold as in the right hand side, a configuration may get repeated when the j^{th} and $(j+1)^{th}$ live blocks are adjacent ($D_i^j = C_i^{j+1}$ for every i in that case). However, it is more convenient to think in terms of subsequences of configurations like $\tilde{C}(k,j)$, and since it does affect the proof, we will continue to use them.

A string written in the j^{th} live block will be scanned by the TM while passing through the configuration subsequences $\tilde{C}(1,j)$, ..., $\tilde{C}(d,j)$ corresponding to the scan by each of the d group of heads. We say that the subsequence $\tilde{C}(k,j)$ is *valid* for a string s of length $b(n)$, if the TM, started on the configuration $C_{t_k}^j$ with s written on the j^{th} live block, passes through each of the configurations in $\tilde{C}(k,j)$ when zeros are written on all the dead blocks except the last one in which the string $\mathbf{01}^{b(n)}\mathbf{0}^r$ is written. Note that while the TM passes through $\tilde{C}(k,j)$, none of its heads will scan any other live block and therefore, the notion is well defined.

We do a stage-wise construction of the sequence *Seq*. At the end of the stage $2n(k-1)+j$, $1 \le k \le d$, $1 \le j \le 2n$, we will have computed subsequences up to $\tilde{C}(k,2j)$. Since we require that the $(2j-1)^{th}$ and $(2j)^{th}$ live blocks have the same choices of strings, we do the computation of $\tilde{C}(k,2j-1)$ and $\tilde{C}(k,2j)$ in a single stage. Further, since we can not store all the computed subsequences within the available space, we only store the subsequences $\tilde{C}(1,2j-1)$, ..., $\tilde{C}(k,2j-1)$, and $\tilde{C}(1,2j)$, ..., $\tilde{C}(k,2j)$ at the end of stage $2n(k-1)+j$.

Stage $2n(k-1)+j$**,** $1 \le k \le d$**,** $1 \le j \le 2n$: We have subsequences $\tilde{C}(1,2j-2)$, ..., $\tilde{C}(k,2j-2)$ from the previous stage if $j > 1$, and subsequences $\tilde{C}(1,4n)$, ..., $\tilde{C}(k-1,4n)$ if $j=1$ and $k>1$. In either case, we can easily compute the first configurations of $\tilde{C}(1,2j-1)$, ..., $\tilde{C}(k,2j-1)$ by first computing the position of the $(2j-1)^{th}$ live block, then starting the TM on the last configurations of the given subsequences and running it on dead blocks till it enters the live block.

Carry out the simulation as in the Stage $2n(k-2)+j$ if $k > 1$. This will compute $\tilde{C}(1,2j-1)$, ..., $\tilde{C}(k-1,2j-1)$, $\tilde{C}(1,2j)$, ..., $\tilde{C}(k-1,2j)$. We now compute $\tilde{C}(k,2j-1)$ and $\tilde{C}(k,2j)$ as follows. For every possible pair of subsequences $\tilde{C}_*(k,2j-1)$ and $\tilde{C}_*(k,2j)$ denoting some entry and exit configurations of the k^{th} group of heads into/from the $(2j-1)^{th}$ and $(2j)^{th}$ live blocks respectively, do the following. Count the number of strings v, $|v| = b(n)$, on which *all of* $\tilde{C}(1,2j-1)$, ..., $\tilde{C}(k-1,2j-1)$, $\tilde{C}_*(k,2j-1)$, $\tilde{C}(1,2j)$, ..., $\tilde{C}(k-1,2j)$, $\tilde{C}_*(k,2j)$ are valid subsequences. Let $\tilde{C}(k,2j-1)$ and $\tilde{C}(k,2j)$ be that pair $\tilde{C}_*(k,2j-1)$ and $\tilde{C}_*(k,2j)$ for which this number is the maximum (in case of a tie, choose the lexicographically smallest pair). When $k = d$, let $I_{i_{2j-1}} = I_{i_{2j}} = \{vv, wv\}$ where v and w are the smallest two non-zero strings on which all the subsequences $\tilde{C}(1,2j-1)$, ..., $\tilde{C}(d,2j-1)$, $\tilde{C}(1,2j)$, ..., $\tilde{C}(d,2j)$ are valid.

It is straightforward to check that the above procedure can be carried out by a logspace TM and the constructed sequence satisfies the condition 4. To see that it also satisfies the conditions 1 and 2, we have to verify that for every j, $1 \leq j \leq 2n$, there are at least two non-zero strings on which all the subsequences $\tilde{C}(1,2j-1)$, ..., $\tilde{C}(d,2j-1)$, $\tilde{C}(1,2j)$, ..., $\tilde{C}(d,2j)$ are valid.

For any j, we show, by induction, that for any k, $0 \leq k \leq d$, there will be $2^{b(n)}/[q_M(n^{2^a})]^{4(t_{k+1}-1)}$ strings on which the all the subsequences $\tilde{C}(1,2j-1)$, ..., $\tilde{C}(k,2j-1)$, $\tilde{C}(1,2j)$, ..., $\tilde{C}(k,2j)$ are valid. It is trivially true for $k = 0$. Assume for $k-1$. There are $t_{k+1} - t_k$ heads in the k^{th} group and therefore the subsequences $\tilde{C}(k,2j-1)$ and $\tilde{C}(k,2j)$ have $4.(t_{k+1} - t_k)$ configurations in all (for every head two entry and two exit configurations). By induction hypothesis, there are $2^{b(n)}/[q_M(n^{2^a})]^{4(t_k-1)}$ strings on which all the subsequences up to $(k-1)^{th}$ group of heads are valid. We choose the subsequences for the k^{th} group of heads that are valid on the largest number of strings from the above set and therefore, the number of such strings is at least $1/[q_M(n^{2^a})]^{4.(t_{k+1}-t_k)} * 2^{b(n)}/[q_M(n^{2^a})]^{4(t_k-1)} = 2^{b(n)}/[q_M(n^{2^a})]^{4(t_{k+1}-1)}$. Therefore, there will be at least $2^{b(n)}/[q_M(n^{2^a})]^{4a} \geq 4$ strings on which all the subsequences are valid. This proves the conditions 1 and 2 and therefore the sequence *Seq* is the desired one.

To compute the sets $I_{i_{2j-1}}$ and $I_{i_{2j}}$, where i_{2j-1} and i_{2j} are respectively the positions of the $(2j-1)^{th}$ and $(2j)^{th}$ live blocks, we simply pick up the smallest two non-zero strings respecting every subsequence in Stage $2n(d-1)+j$. ■

The proper strings are the strings in the set S defined in the condition 4 of the above lemma. As the definitions of proper strings has changed, we need a slightly different definition of the set $code(L)$: $y \in code(L)$ iff $y = u_1 u_2 \cdots u_t \mathbf{01}^b \mathbf{0}^r$ for $r \geq 0$ satisfying the following conditions—

1. $|u_1| = \cdots = |u_t| = b$,
2. Let $\{u_{k_1}, \ldots, u_{k_{4n}}\}$ be the set of all non-zero u_is and $x = x(1) \cdots x(n)$ with $x(i) = \mathbf{1}$ if $u_{k_{2i-1}} = u_{k_{2i}}$, $\mathbf{0}$ otherwise. Then *either* for every i, $n+1 \leq i \leq 2n$, $u_{k_{2i-1}} = u_{k_{2i}}$ and $x \in L$, *or* there is a j, $n+1 \leq j \leq 2n$, such that for every i, $n+1 \leq i \neq j \leq 2n$, $u_{k_{2i-1}} = u_{k_{2i}}$, $u_{k_{2j-1}} \neq u_{k_{2j}}$ and $x(j-n) = \mathbf{1}$.

The construction of g_f also changes suitably:

On input x, let $n = 2^{\lceil \log |x| \rceil}$. If $n < n_0$ then output

$$x[1]\mathbf{11}x[1]x[2]\mathbf{11}x[2] \cdots x[|x|]\mathbf{11}x[|x|]\mathbf{1}^{4.|x|}\mathbf{011}$$

and halt. Otherwise, for each $1 \leq j \leq 2.|x|$, do the following.

Compute the positions of $(2j-1)^{th}$ and $(2j)^{th}$ live blocks, say i_{2j-1} and i_{2j}, and the set $I_{i_{2j-1}} = I_{i_{2j}} = \{w_j, v_j\}$. Output $\mathbf{0}^{(i_{2j-1}-i_{2j-2}-1).b(n)} v_j \mathbf{0}^{(i_{2j}-i_{2j-1}-1).b(n)} v_j$ if $j > |x|$ or $x[i] = \mathbf{1}$, $\mathbf{0}^{(i_{2j-1}-i_{2j-2}-1).b(n)} w_j \mathbf{0}^{(i_{2j}-i_{2j-1}-1).b(n)} v_j$ otherwise (letting $i_0 = 0$).

Finally, output $\mathbf{0}^{(m(a)-i_{4|x|}).b(n)}\mathbf{01}^{b(n)}\mathbf{0}^r$ where $m(a) = 1/2*(8n)^{2^{a-1}}$ and $r = m(a).b(n) - b(n) - 1$, and halt.

Function g_f, in fact, is a 1-L function. Analogues of Proposition 4.1, and Lemmas 4.2, 4.3, and 4.4 continue to hold as the output of M while any of the d group of heads are scanning a live block is independent of the assignment of other live blocks when the TM is working on proper strings. We now prove the collapsing result for 1-omL reductions.

Theorem 6.3 *For any class $\mathcal{C}$ closed under logspace reductions, every $\leq_m^{1\text{–}omL}$-hard set for $\mathcal{C}$ is also $\leq_{1,qli,i}^{1\text{–}omL}$-hard.*

Proof Sketch. Let A be a $\leq_m^{1\text{–}omL}$-hard set for $\mathcal{C}$ and $L \in \mathcal{C}$. Set $\tilde{L}$ and function g_p are defined analogously. Let $h = f \circ g_f \circ g_p$, where $code(\tilde{L}) \leq_m^{1\text{–}omL} A$ via f. It is clear that h is a one-one, and length-squaring reduction of L to A. To show that it a 1-omL computable as well as 1-omL-invertible function, we have to do a little more work as the class of 1-omL functions is not closed under composition.

Define a TM to be *bi-oblivious* if, in addition to the input head(s), movement of its output head also depends only on the input length. It is easy to see that the composition of a 1-omL function with a bi-oblivious 1-mL function is a 1-omL function. Now to see that h is a 1-omL function, it is sufficient to note that the functions g_f and g_p can be computed by a bi-oblivious 1-L TM.

Finally, to invert the function h, we again define a TM M' that is a composition of the following two TMs.

TM $\mathbf{M_1}$: This TM, as before, computes the function $(f \circ g_f)^{-1}$ correctly on the range of the function h. Let the function f be computed by TM M with a input heads. TM M_1 also has a input heads and it executes the following procedure. Given input z, first find out the size, say n, of the input string on which the size of the output of $f \circ g_f$ is $|z|$. (Again, the padding by function g_p ensures that there can be at most one such n.) If none, then reject. Otherwise, compute the division of input heads of the TM M (computing f) into d disjoint groups (on strings of size n^{2^a}) as in Lemma 6.2. Now, say that an output bit is *produced* by the k^{th} group of heads of TM M if the bit is outputted while the TM is moving between the configurations $C^1_{t_k}$ and $C^1_{t_{k+1}}$ (we take $C^1_{t_{d+1}}$ to be the final configuration). If a bit is outputted while M moves from the initial configuration to C^1_1, we say it is produced by the first group. It follows that the output can thus be divided in d disjoint contiguous blocks corresponding to the output produced by each of the d group of heads. Calculate the positions of these d blocks of bits in the input z—these positions will remain the same for all strings of length $|z|$—and place one head (of M_1) at the beginning of the each such block. Now compute the configuration subsequences $\tilde{C}(1,1)$, $\tilde{C}(2,1)$, …, $\tilde{C}(d,1)$, $\tilde{C}(1,2)$, $\tilde{C}(2,2)$, …, $\tilde{C}(d,2)$, and the sets I_{i_1} and I_{i_2}, $I_{i_1} = I_{i_2} = \{v, w\}$, for the first two live blocks and then simulate all the heads on the two assignments vv and wv to these blocks to decide if the output of M on any of these two assignments matches with the prefix of each of the d groups in the given input. If none then reject; otherwise output **1** or **0** accordingly. Do it for every successive pair of live blocks. This procedure computes the inverse in logspace with d one-way input heads. Of course, the value of d varies with the size of z, however, it is always bounded by a. The function g_p ensures that the output length of $f \circ g_f \circ g_p$ is different for different input lengths. Thus, TM M_1 is bi-oblivious as its output head moves by exactly one bit after simulation for every pair of blocks.

TM $\mathbf{M_2}$: This TM remains identical to the one defined in the proof of Theorem 4.5. It would be an oblivious 1-L TM.

The composition of the above two TMs computes the inverse of h and the TM computing it is a 1-omL TM. ■

Immediate corollaries:

Corollary 6.4 *For any class $\mathcal{C}$ closed under logspace reductions, $\leq_m^{1\text{–}omL}$-complete degree for $\mathcal{C}$ collapses to $\leq_{1,qli,i}^{1\text{–}omL}$-complete degree.*

Corollary 6.5 *For any class $\mathcal{C}$ closed under logspace reductions, if A is a $\leq_m^{1\text{–}omL}$-hard set for $\mathcal{C}$ and t is a 1-1, 1-omL function then $t(A)$ is also $\leq_m^{1\text{–}omL}$-hard.*

Proof. To prove this, we just need to observe that the function h of the above theorem is a bi-oblivious 1-mL function. Function $g_f \circ g_p$, as noted above, is a bi-oblivious 1-L function. Since all the heads of TM M, computing f, enter and exit from each live block in the same configuration on any string of a fixed length n in the range of $g_f \circ g_p$, and each live block is $O(\log n)$ bits long, there is a 1-mL TM that computes h by 'spacing' the output of M on the live blocks so that the output head movement does not depend on the string written on the block. This makes the TM bi-oblivious. Now, it follows that $t \circ h$ is a 1-omL function. ■

Corollary 6.6 *The 1-omL-encrypted complete set conjecture is false.*

Corollary 6.7 *For any class $\mathcal{C}$ closed under logspace reductions, all $\leq_m^{1\text{–}omL}$-complete sets for $\mathcal{C}$ are logspace-isomorphic.*

Is the 1-omL-isomorphism conjecture true? We suspect not, as to compute the length of f^{-1}-g^{-1} chain as in [13], it appears that a polynomial number of input heads are needed instead of just a constant.

7 c-L reductions

In this section, we study yet another reducibility, viz., c-L. It behaves differently from the above three ones in that the c-L-encrypted complete set conjecture is true. We first show that this reducibility is closely related to 1-omL reducibility.

Corollary 7.1 *For any class $\mathcal{C}$ closed under logspace reductions, every $\leq_m^{1\text{–}omL}$-hard set for $\mathcal{C}$ is also $\leq_{1,qli}^{c\text{–}L}$-hard via a function that is 1-omL-invertible.*

Proof Sketch. In fact the reduction h constructed in the proof of Theorem 6.3 is a c-L function. This follows from the Lemma 6.2 and the constructions of functions g_f and g_p in the proof above. Lemma 6.2 allows us to construct function g_f such that the a heads of the TM M, computing function f on the range of g_f, can be divided into d disjoint groups, $d \leq a$, d varying with the input length, satisfying the following properties.

1. For each k, $1 < k \leq d$, heads in the k^{th} group start scanning the live blocks only after the heads in the $(k-1)^{th}$ group have scanned all the live blocks.

2. For each k, $1 \leq k \leq d$, and for each live block, all the heads in the k^{th} group scan it before any head in the group starts scanning the next one.

Now a a-L TM can compute the function f on the range of g_f in the following way—in the k^{th} scan of the input, compute the output of M while its k^{th} group of heads are scanning the live blocks. This can be done by a single head as, for each live block, the TM can write its contents in its work tape and simulate all the heads in the k^{th} group on it. The above two properties ensure that no other live block is scanned during this simulation. Therefore, the function $f \circ g_f$ is a c-L function and so is the function $h = f \circ g_f \circ g_p$ (remember that c-L functions are closed under composition and both g_f and g_p are 1-L functions). ■

The above result immediately raises the following question: is the class of c-L functions properly contained in the class of 1-omL functions? If so, then we have an interesting collapse of $\leq_m^{1\text{–}omL}$-complete degrees to complete degrees under a strictly weaker class of reductions. The following proposition shows that it is indeed so.

Proposition 7.2 $\mathcal{F}$(c-L) $\subset \mathcal{F}$(1-omL).

Proof Sketch. Any c-L function f can be computed by an *oblivious* c-L TM in the following way—suppose that a TM, say M, computing f works for at most $p(n)$ steps on input strings of size n. Then a TM that takes exactly $p(n)$ steps to simulate steps of M between two consecutive movements of the input head will be an oblivious c-L TM computing f since the movement of the input head is one-way. Now, this oblivious TM can be simulated by a 1-omL TM—put one head for each scan of the input. Therefore, $\mathcal{F}$(c-L) $\subseteq \mathcal{F}$(1-omL).

By Proposition 7.6 proved below, the inverse of the function $t(x) = xx$ is computable by a 1-omL TM but not by a c-L TM. This completes the proof. ■

Thus, as a reduction from complete sets, 1-omL functions are no more powerful than c-L functions!

Corollary 7.3 *For any class $\mathcal{C}$ closed under logspace reductions, set A is $\leq_m^{1\text{–}omL}$-hard set for $\mathcal{C}$ iff A is $\leq_m^{c\text{–}L}$-hard for $\mathcal{C}$ iff A is $\leq_{1,qli}^{c\text{–}L}$-hard for $\mathcal{C}$ via a function that is 1-omL-invertible.*

Can we obtain a further collapse by making the reductions in the above corollary c-L-invertible? Surely, it would be possible if the c-L-encrypted complete set conjecture is false. However, it turns out that, unlike the previous three reducibilities, c-L-encrypted complete set conjecture is true. Towards this, we first define c-L-annihilating functions in the same spirit as [20].

Definition 7.4 A function f is *c-L-annihilating* function if it is a 1-1, length-increasing, c-L function such that every subset of the range of f that is recognized by a c-L TM is sparse.

Now, the following proposition follows.

Proposition 7.5 *If c-L-annihilating functions exist then the c-L-encrypted complete set conjecture is true.*

Proof. Suppose the conjecture is false. Let A be a $\leq_m^{c\text{–}L}$-complete set for NP and f be a c-L-annihilating function. Define $B = \mathbf{1}A \cup \mathbf{0}\Sigma^*$. Set B too is $\leq_m^{c\text{–}L}$-complete for NP. Consider the set $f(B)$. Since the conjecture is false, there is a 1-1, length-increasing, c-L function g reducing B to $f(B)$ such that g is c-L-invertible as well. Define the set C as: $x \in C$ iff $g^{-1}(x)$ exists and belongs to $\mathbf{0}\boldsymbol{\Sigma}^*$. C is a non-sparse set recognizable by a c-L TM as well as a subset of the range of f. A contradiction. ■

It is easy to show that c-L-annihilating functions exist. Define $t(x) = xx$.

Proposition 7.6 *t is a c-L-annihilating function.*

Proof. Function t is clearly a 1-1, length-increasing, c-L function. Let the c-L TM M recognize a subset S of its range. Let M be a k-L TM with $q_M(n)$ being the number of configurations in $config(M, 2n)$. We show that $\|S_{=2n}\| \leq [q_M(n)]^{2k}$. Consider the $(1, n)$-block restricted computation sequences on input strings of size $2n$. There will be two configurations in these sequences corresponding to each pass of the input by the input head, and therefore, $2k + 2$ configurations in all (including initial and final configurations). How many different such sequences exist? Clearly, not more than $[q_M(n)]^{2k}$ (we can take initial and final configurations to be fixed). If there are two strings in S, say xx and yy, that share the same $(1, n)$-block restricted computation sequence then strings xy and yx would also belong to S. But this is not possible since $x \neq y$ and S is a subset of the range of t. Therefore, S can contain at most $[q_M(n)]^{2k}$ strings of size $2n$. Note that S will have no string of odd length. Therefore, S is sparse. Since the TM M was arbitrary, it follows that t is a c-L-annihilating function. ■

Corollary 7.7 *c-L-encrypted complete set conjecture is true.*

8 Discussion

The motivation for relocating the conjectures to weaker reducibilities, or to higher classes, has been that an answer of the relocated conjectures may shed some light on the answer of the conjectures in their original form. So, what, if any, is the implication of the above results in this sense? At a first glance, they do not seem to favor any conjecture as all three possible answers to the two conjectures have been shown to exist for different reducibilities—both are false for 1-L while the isomorphism conjecture is true for 1-NL and the encrypted complete set conjecture is true for c-L reductions. However, on a closer look these results appear to support the p-isomorphism conjecture. To see this, we first identify two properties of a reducibility r.

Simple reducibility. Reducibility r is *simple* for class $\mathcal{C}$ if every $\leq^r_m$-complete set for $\mathcal{C}$ is also $\leq^r_{1,li}$-complete.

Easily invertible reducibility. Reducibility r is *easily invertible* for class $\mathcal{C}$ if—given that the inverse of every one-one, length-increasing function in $\mathcal{F}(r)$ is computable by a *nondeterministic* TM working within a resource bound of s, $s \geq r$ (i.e., the inverting TM is allowed to use more resources but not less)—every $\leq^r_{1,li}$-complete set for $\mathcal{C}$ is also $\leq^r_{1,li}$-complete via reductions whose inverses are computable by *deterministic* TMs working within a resource bound of s.

It is straightforward to see that the p-isomorphism conjecture holds if and only if the polynomial-time reducibility is both simple and easily invertible for NP as the inverse of any one-one length-increasing polynomial-time function is computable by an NDTM working in polynomial-time. We shall refer to the conjecture that reducibility r is both simple and easily-invertible for class $\mathcal{C}$ as the *r-complete degree conjecture for $\mathcal{C}$*. We now show that for reducibilities 1-L, 1-omL, and c-L—for which we have been unable to prove the isomorphism conjecture—the r-complete degree conjecture for NP either holds or is very likely to hold.

That the conjecture holds for reducibilities 1-L and 1-omL follows directly from Theorems 4.5 and 6.3 respectively. In fact, as shown in Proposition 6.1, there are 1-omL functions whose inverses are computable only by polynomial-time NDTMs and yet the $\leq^{1-omL}_{1,li}$-complete sets for NP are also $\leq^{1-omL}_{1,li,i}$-complete. So, for 1-omL reductions, the inverses are stronger than desired.

The interesting case is that of c-L reductions. It can be shown—by a direct adaptation of the proof of Proposition 7.6—that there are one-one, length-increasing c-L functions whose inverses are *not* computable by non-deterministic c-L TMs ($t(x) = xx$ is one such function). In fact, there seems no better way to invert a general c-L function than by a non-deterministic 1-mL TM—keep one head for the output produced during each scan of the c-L TM; guess the length of the output produced during each scan and position the heads accordingly; now guess the input to the c-L TM bit-by-bit and verify. Therefore—as by Corollary 7.1 the $\leq^{c-L}_{1,li}$-complete sets for NP are also $\leq^{c-L}_{1,li}$-complete via reductions that are 1-omL-invertible—the c-L reducibility too appears to be easily invertible for NP. It is also simple for NP (direct from Corollary 7.1).

The 1-NL-complete degree conjecture is clearly true (follows from Theorem 5.2). The only point to note is that in the definition of the easily invertible property we require the resource bound on the inverting TM to be at least as much as r which in case of 1-NL TMs is 1-NL (we count non-determinism also as a resource). And so the conclusion that a deterministic TM must compute the inverse within a resource bound of s is satisfied as non-determinism is still allowed in the resource bound s.

Thus, for all the four reducibilities that we have considered the r-complete degree conjecture for NP (in fact for any class closed under logspace reductions) is true (or very likely true). So, our results can be interpreted as providing evidence for the p-isomorphism conjecture provided one believes that the complete degrees for these weak reducibilities have similar 'structure' as the complete degree for polynomial-time reducibility. However, this is far from clear. The reducibilities that we considered have the special feature of scanning the input tape only constantly many times and that too in only one direction. The proofs make heavy use of this property and do not work for any non-constant number of scans.

In view of the above, more investigation is needed—particularly of r-complete degree conjectures for various other reducibilities—before we can positively conclude anything about the p-isomorphism conjecture. For some other reducibilities also the answer to the above conjecture is known. In [2] it is shown that the 2DFA-complete degree conjecture for NP is most likely true (2DFA denotes the class of reductions computed by two-way DFA transducers), and it can be easily inferred from the results in [3] that FOP-complete degree conjecture for NP is true (FOP denotes the class of first order projections, see [3] for definition). However, these two classes of reductions are extremely weak and so do not add much to the evidence(?) already provided above for p-complete degree conjecture being true for NP.

Finally, a few words on the technique. The technique that we use is essentially a refinement of the one used in [1]. In [1], the function g_f, mapping L to $code(L)$, 'codes' the entire string in one go whereas here we do the coding 'bit-by-bit'. This allows us to obtain a stronger collapse of complete degrees—in [1] it could only be shown that all $\leq^{1-L}_m$-complete sets for NP are p-isomorphic. Further, our results show that this technique is more useful than the standard diagonalization one at least for weak reductions like 1-L, 1-NL etc. In [15] it was shown, using the diagonalization technique, that $\leq^{1-L}_m$- (and $\leq^{1-NL}_m$-) complete sets for non-deterministic space classes above NLOG are also $\leq^{1-L}_{1,li}$- (resp. $\leq^{1-NL}_{1,li}$-) complete. We have been able to improve this result in two ways—one, we show that the $\leq^{1-L}_m$- and $\leq^{1-NL}_m$-complete sets are also respectively $\leq^{1-L}_{1,li,i}$- and $\leq^{1-NL}_{1,li,i}$-complete; and two, this result holds for all classes closed under logspace reductions.

9 Open questions

As we have observed above, it would be interesting to investigate the r-complete degree conjecture for NP for various reducibilities r, the most important ones being of course, logspace and polynomial-time. We list here a few reducibilities for which the answer to the complete degree

conjecture appears more tractable.

1. We have not been able to answer the conjecture for the more natural class of 1-mL reductions. The obliviousness condition appears crucial for our proof to work. As shown in section 6, the inverses of one-one, length-increasing 1-mL reductions can only be computed by non-deterministic polynomial-time TMs in general. So, to prove the 1-mL-complete degree conjecture for NP, we have to show that all $\leq_m^{1-mL}$-complete sets for NP are also $\leq_{1,li}^{1-mL}$-complete via reductions that are p-invertible.

2. We have concentrated on functions computed by TMs with one-way input head(s) as they are provably weaker than polynomial-time functions. However, there are other such classes of functions, e.g., *uniform-AC^0* or *first-order* functions [10]. It is easy to show that there are p-one-way functions that are first-order computable (the p-one-way 1-omL function exhibited in section 6 is also first-order computable), so to prove the complete degree conjecture for NP for these reductions we have to show that all $\leq_m^{fo}$-complete sets for NP are also $\leq_{1,li}^{fo}$-complete via p-invertible reductions.

Acknowledgement: I would like to thank Professor Somenath Biswas for several suggestions and corrections in an earlier draft. I am also grateful to Professor Steve Homer for providing constant encouragement and support.

References

[1] M. Agrawal and S. Biswas. Polynomial isomorphism of 1-L-complete sets. In *Proceedings of the Structure in Complexity Theory Conference*, pages 75–80, 1993.

[2] M. Agrawal and S. Venkatesh. The isomorphism problem for 2-DFA reductions. Technical Report TCS-94-4, School of Mathematics, SPIC Science Foundation, Madras, India, 1994.

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

[4] E. W. Allender. Isomorphisms and 1-L reductions. In *Proceedings of the Structure in Complexity Theory Conference*, pages 12–22. Springer Lecture Notes in Computer Science 223, 1986.

[5] E. W. Allender. Isomorphisms and 1-L reductions. *J. Comput. Sys. Sci.*, 36(6):336–350, 1988.

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

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

[8] H. Burtschick and A. Hoene. The degree structure of 1-L reductions. In *Proceedings of Math. Foundation of Computer Science*, pages 153–161. Springer Lecture Notes in Computer Science 629, 1992.

[9] S. Fenner, L. Fortnow, and S. Kurtz. The isomorphism conjecture holds relative to an oracle. In *Proceedings of Annual IEEE Symposium on Foundations of Computer Science*, pages 30–39, 1992. To appear in SIAM J. Comput.

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

[11] K. Ganesan and S. Homer. Complete problems and strong polynomial reducibilities. In *Proceedings of the Symposium on Theoretical Aspects of Computer Science*, pages 240–250. Springer Lecture Notes in Computer Science 349, 1988.

[12] J. Grollmann and A. Selman. Complexity measures for public-key cryptosystems. In *Proceedings of Annual IEEE Symposium on Foundations of Computer Science*, pages 495–503, 1984.

[13] J. Hartmanis. On log-tape isomorphisms of complete sets. *Theoretical Computer Science*, pages 273–286, 1978.

[14] J. Hartmanis, N. Immerman, and S. Mahaney. One-way log-tape reductions. In *Proceedings of Annual IEEE Symposium on Foundations of Computer Science*, pages 65–72, 1978.

[15] L. A. Hemchandra and A. Hoene. Collapsing degrees via strong computation. In *Proceedings of the International Colloquium on Automata, Languages and Programming*, pages 393–404. Springer Lecture Notes in Computer Science 510, 1991.

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

[17] D. Joseph and P. Young. Some remarks on witness functions for nonpolynomial and noncomplete sets in NP. *Theoretical Computer Science*, 39:225–237, 1985.

[18] K. Ko. On some natural complete operators. *Theoretical Computer Science*, 37:1–30, 1985.

[19] S. Kurtz, S. Mahaney, and J. Royer. The structure of complete degrees. In A. Selman, editor, *Complexity Theory Retrospective*, pages 108–146. Springer-Verlag, 1988.

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

[21] S. Kurtz, S. Mahaney, and J. Royer. Average dependence and random oracles. In *Proceedings of the Structure in Complexity Theory Conference*, pages 306–317, 1992.

[22] A. L. Selman. A survey of one-way functions in complexity theory. *Mathematical Systems Theory*, 25:203–221, 1992.

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

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