Polynomial-Time Isomorphism of 1-L-Complete Sets

Manindra Agrawal

School of Mathematics
SPIC Science Foundation
Madras - 600 017, INDIA
email : manindra@ssf.ernet.in

&

Somenath Biswas

Department of Computer Science and Engineering
Indian Institute of Technology
Kanpur - 208016, INDIA
email : sb@iitk.ernet.in

Running Head : Isomorphism of 1-L-Complete Sets

Mailing address :

Manindra Agrawal

School of Mathematics
SPIC Science Foundation
92, G. N. Chetty Road, T. Nagar
Madras - 600 017, INDIA

Abstract

Let $\mathcal{C}$ be any complexity class closed under log-lin reductions. We show that all sets complete for $\mathcal{C}$ under 1-L reductions are polynomial-time isomorphic to each other. We also generalize the result to reductions computed by *finite-crossing* machines. As a corollary, we show that all sets complete for $\mathcal{C}$ under 2-way DFA reductions are polynomial-time isomorphic to each other.

1 Introduction

The isomorphism conjecture for a class $\mathcal{C}$ is: All $\leq_m^p$-complete sets for $\mathcal{C}$ polynomial-time isomorphic. Resolving this conjecture for various classes is a fundamental problem as its answer for the class $\mathcal{C}$ will provide us with important insight into the structure of the m-complete degree of $\mathcal{C}$. We note that if the conjecture is true for some class $\mathcal{C}$ then it immediately follows that, from the viewpoint of structural complexity theory, all m-complete sets for $\mathcal{C}$ are merely encodings of the same set. A negative answer also has important consequences (see [14] for a survey).

Berman and Hartmanis [6] began the study of the isomorphism conjecture for the class NP. By observing that all known NP-complete sets at that time were indeed polynomial-time isomorphic (in short, p-isomorphic) to each other, they provided strong evidence for the conjecture. Since then, however, some evidences against the conjecture have been found (see, e.g., [13], [10], [15]) and the conjecture is generally believed to be false now (though recently, there has been some positive evidence as well [7]). Yet, the conjecture remains far from settled. In fact, till now, there is no naturally defined unrelativized class for which the isomorphism conjecture has been answered. There are only some partial results for some of the classes, e.g., all $\leq_m^p$-complete sets for E are known to be $\leq_{1,li}^p$-complete [5, 17, 8] and all $\leq_m^p$-complete sets for NE are known to be $\leq_1^p$-complete [8].

As the answer to the isomorphism conjecture remains elusive for natural classes, attempts have been made to answer weaker versions of the conjecture by strengthening completeness under polynomial-time reductions to completeness under weaker reductions. Thus, the isomorphism conjecture for class $\mathcal{C}$ *weakened to reduction* r is: All $\leq_m^r$-complete sets for $\mathcal{C}$ are p-isomorphic. When one weakens the conjecture to logspace reductions, there is no significant progress. The only additional partial result available is that all $\leq_m^{log}$-complete sets for PSPACE are $\leq_{1,li}^{log}$-complete [9]. However, there are several new results for the conjecture weakened to 1-L reductions, a restriction of logspace reductions. 1-L reductions, introduced in [11], are functions computed by logspace TMs with their input head being one-way. These reductions may appear too restrictive, however, it has been observed that all 'natural' NP-complete sets are also complete under 1-L reductions [11]. Allender [4] has shown that $\leq_m^{1-L}$-complete sets for all reasonable deterministic classes above, and including, PSPACE are p-isomorphic. Ganesan and

Homer [8] show the same result for the classes NE and NEXP. Allender [4] also shows that for reasonable classes below PSPACE, $\leq_m^{1-L}$-complete sets are also complete under size-increasing and strongly p-invertible reductions, though these reductions may not be one-one. Recently, Hemachandra and Hoene [12] have improved these results for several classes by showing that for any reasonable nondeterministic space class above NLOG, all $\leq_m^{1-L}$-complete sets are p-isomorphic (in fact, they show such sets to be isomorphic under even nondeterministic logspace reductions). However, these results still fail to answer the isomorphism conjecture weakened to 1-L reductions for some of the most well known and important classes, e.g., DLOG, P, NP etc.

In this paper, we settle the isomorphism conjecture weakened to 1-L reductions: we show that for any class $\mathcal{C}$ closed under log-lin reductions, all $\leq_m^{1-L}$-complete sets for $\mathcal{C}$ are p-isomorphic. We are also able to generalize the result to functions computed by *finite crossing* TMs (logspace TMs whose input head crosses any cell a constant number of times). As a corollary to this result, we get that for any class $\mathcal{C}$ closed under log-lin reductions, all sets complete for $\mathcal{C}$ under 2-way DFA reductions (computed by TMs that do not have any workspace) are p-isomorphic. This too generalizes some results in [4, 8].

To obtain the above results, we have used a technique which is different from the standard one that diagonalizes over all possible reductions. We argue that our technique is more useful in certain situations.

2 Preliminaries

The strings are over $\Sigma = \{0,1\}$. We denote, by $\Sigma_{=n}$, the set of all strings over Σ of length n. We shall denote, by $x \oplus y$, the XOR of strings x and y, $|x| = |y|$. (XOR of two strings x and y of length n is a string of length n whose i^{th} bit is 1 iff exactly one bit amongst the i^{th} bit of x and the i^{th} bit of y is 1.)

For sets A and B, if B reduces to A via a (one-one, one-one and size-increasing, one-one and size-increasing and polynomial-time invertible) many-one polynomial-time function, then we say that $B(\leq_1^p, \leq_{1,li}^p, \leq_{1,li,i}^p) \leq_m^p A$. For a class $\mathcal{C}$ and set A, we say that A is $(\leq_1^r, \leq_{1,li}^p, \leq_{1,li,i}^p) \leq_m^p$-complete for $\mathcal{C}$ if for every set B in $\mathcal{C}$, $B(\leq_1^p, \leq_{1,li}^p, \leq_{1,li,i}^p) \leq_m^p A$ and $A \in \mathcal{C}$.

A *log-lin* function [16] f is a function computable by a logspace TM such that for all x, $|f(x)| = O(|x|)$. *1-L TMs* are logspace-bounded Turing machines which have an input tape, an output tape and a work tape such that the input head never makes a left move and at the beginning of the computation, $\lceil \log n \rceil$ cells are marked off on the work tape. Functions computed by 1-L TMs are called *1-L functions.* They were introduced in [11] for studying complete sets for DLOG. As above, we say that $B \leq_m^{1\text{–}L} A$ if B reduces to A via a 1-L function, and A is $\leq_m^{1\text{–}L}$-complete for $\mathcal{C}$ if $A \in \mathcal{C}$ and for every $B \in \mathcal{C}$, $B \leq_m^{1\text{–}L} A$.

We shall mainly be dealing with 1-L TMs computing *total* functions in the paper. Without loss of generality, we can assume that these machines have a unique halting state and just before the machine enters this state, it moves the input head to the right of the rightmost input bit. We also assume that a special symbol B is written to the right of the input string on the input tape of M. From now on, whenever we refer to a TM, we assume it to be a 1-L TM of the above kind, unless explicitly stated otherwise.

Definition 2.1 A *configuration of M of size n* is a partial ID of M for input strings 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.

Definition 2.2 The *computation graph of M for input strings of size n*, called G_M^n, is defined as follows:

1. $G_M^n = G(V^n, E^n)$ where edges in E^n are directed and *labelled.* Further, there may be multiple edges between two vertices. These multiple edges are distinguished by their labels. The vertices in V^n are configurations of M of size n.

2. The edge $\langle C, D\rangle \in E^n$ has label $\langle i, o\rangle$, $i \in \Sigma \cup \{B\}, o \in \Sigma^*$, ($C$, D are vertices of V^n) iff there is a sequence of configurations $C_0, C_1, \ldots, C_{k-1}, C_k$ such that

   (a) $C_0 = C$ and $C_k = D$.

   (b) Let the input head position in C_0 be at r. Then for every j, $0 < j < k$, the input head position in C_j is at r. If $r \leq n$ then the input head position in C_k is at $r + 1$. If $r = n + 1$ then the input head position in C_k is at r and the state in C_k is the halting state.

(c) For every j, $0 \leq j < k$, M moves from configuration C_j to configuration C_{j+1} in a single step on reading input i. (The graph may have multiple edges as M may move from C_0 to C_k on reading both 0 and 1. It follows that there may be at most two edges between any pair of vertices.)

(d) String o is the output of M while moving from configuration C_0 to C_k.

Strings i and o are respectively called *input label* and *output label* of the edge. The above condition ensures that every edge $\langle C, D \rangle$ of the graph G_M^n corresponds to the consumption of a single input symbol except possibly when D is the halting configuration.

We refer to the starting configuration of M of size n as C_{init}^n (C_{init}^n may be taken to be $\langle q_0, 1, 1, 1, 1^{\lceil \log n \rceil} \rangle$ where q_0 is the start state). We shall also assume that G_M^n does not contain any vertex not reachable from C_{init}^n (this can be ensured by deleting all such non-reachable vertices). It follows that G_M^n is acyclic as the existence of a cycle will imply that M does not halt on some input which clearly is not possible. Thus, for every n, graph G_M^n captures the computation of M on input strings of size n. The number of vertices in G_M^n is bounded by a polynomial in n as M works in logspace. Let $\|V_n\| \leq q(n)$ for some polynomial q.

Definition 2.3 Let p be a path of G_M^n from configuration C_1 to C_2. (A path is an ordered sequence of adjacent edges in the graph.) The *input label* i_p of path p is the concatenation of input labels of the edges in the path p. In other words, i_p is the input read by M while moving from configuration C_1 to C_2 along the path p. Similarly, we define the *output label* o_p of the path p to be the concatenation of output labels of the edges in the path p.

We say that a vertex of G_M^n is at *level* i if the position of the input head at the vertex is i. There are exactly $n + 1$ levels in G_M^n. Let L_i be the set of vertices at level i.

Since vertices at any level i have edges going out to vertices at level $i + 1$ only, if there is a path from a vertex at level i to a vertex at level j then it must have exactly $j - i$ edges. Since every path has a unique input label, the following lemma is immediate.

Lemma 2.4 *For every n, for every k, $k \leq n$, and for each string s of length k, there is a unique path with input label s in G_M^n, starting from C_{init}^n and ending in some vertex at level $k+1$.*

Lemma 2.5 *Let n_0 be smallest number such that $(\forall n) n \geq n_0 \Rightarrow 2^n \geq 2.q(5n)$, where $q(n)$ is the polynomial bounding the number of vertices in G_M^n. For every n, $n \geq n_0$, there is a vertex, $\tilde{C}$, at level $2n+1$ of G_M^{5n} such that there are at least 2^{n+1} different paths from C_{init}^{5n} to $\tilde{C}$.*

Proof. By the above lemma there will be exactly 2^{2n} paths from C_{init}^{5n} to vertices in L_{2n+1}. Since the number of vertices in L_{2n+1} is obviously bounded by the total number of vertices $q(5n)$ in G_M^{5n}, there will a vertex $\tilde{C}$ in L_{2n+1} with at least $2^{2n}/q(5n) \geq 2^{n+1}$ (as $n \geq n_0$) different paths from C_{init}^{5n}. ∎

The above lemma will be the key to the proof in the next section. It essentially says that TM M, after reaching the configuration $\tilde{C}$, does not 'remember' which of the 2^{n+1} strings it has read. By exploiting this property, we force f, the function computed by M, to be 1-1 on such strings.

3 Main Result

We shall use $l(x)$ to denote the position of the string x in the standard lexicographic ordering of strings. For any set L, define

$$code(L) = \{yyx : |y| = 2|x| \wedge x \in L\} \cup \{y_1 y_2 x : |y_1| = |y_2| = 2|x| \wedge l(y_1) > l(y_1 \oplus y_2)\}$$

The following proposition is obvious.

Proposition 3.1 *For any L, $L \neq \emptyset, \Sigma^*$, $code(L)$ reduces to L via a log-lin function.*

We now prove our main theorem.

Theorem 3.2 *Let $\mathcal{C}$ be a class closed under log-lin reductions. If A is a $\leq_m^{1-L}$-complete set for $\mathcal{C}$ then for every $B \in \mathcal{C}$, $B \leq_{1,li,i}^p A$.*

Proof. Let $B \in \mathcal{C}$. Define set D as:

$$D = \{x10^k \mid k \geq 0 \wedge x \in B\}$$

It is easy to see that that D reduces to B via a log-lin function and therefore, $D \in \mathcal{C}$. Further, since $code(D)$ reduces to D via a log-lin function (from Proposition 3.1), $code(D) \in \mathcal{C}$. Let $code(D) \leq_m^{1\text{-}L} A$ via 1-L function f which is computed by TM M.

The outline of the proof is as follows. We first obtain a polynomial-time reduction g_D^M of D to $code(D)$ based on the reduction f of $code(D)$ to the set A. Then, we show that the composition of f and g_D^M is a reduction of D to A that is one-one and size-nondecreasing on $\Sigma_{=n}$ for all large enough n. Finally, we give a reduction g_p of B to D such that $f \circ g_D^M \circ g_p$ is a one-one, size-increasing and p-invertible reduction of B to A.

Let n_0 be as defined in Lemma 2.5. Define function g_D^M as computed by the following polynomial-time TM.

> On input x, let $n = |x|$. If $n < n_0$ then output $1^{4n}x$ and halt. Otherwise, for graph G_M^{5n}, compute the configuration $\tilde{C}$ as in Lemma 2.5. Let I be the set of input labels of paths from C_{init}^{5n} to $\tilde{C}$ (each such label will be of length $2n$) and i_p be the $l(x)^{th}$ largest label, in the lexicographic ordering, in I. (Since $l(x) < 2^{|x|+1} = 2^{n+1} \leq \|I\|$, there will always be such a label in I.) Output $i_p i_p x$ and halt.

Claim 3.2.1 *g_D^M is a 1-1, size-increasing, p-invertible and polynomial-time computable reduction of D to $code(D)$.*

Proof of Claim 3.2.1. That g_D^M is 1-1, size-increasing and a reduction of D to $code(D)$ follows immediately. To compute g_D^M in polynomial-time, we need to compute vertex $\tilde{C}$ and the $l(x)^{th}$ largest label of I in polynomial-time. This can be done easily using dynamic programming techniques. Polynomial-time procedures to compute these values are given in the Appendix. Now, the p-invertibility of g_D^M follows easily: given z, check if $|z| = 5n$ for some n. If yes, then let x be the postfix of z of length n, check if $g_D^M(x) = z$. If yes then output x else reject. □

The following claim shows that $f \circ g_D^M$ is 1-1 and size-nondecreasing on $\Sigma_{=n}$ for any $n \geq n_0$ where n_0 is as defined in Lemma 2.5.

Claim 3.2.2 *For any n, $n \geq n_0$: for any x and y, $|x| = |y| = n$ and $x \neq y$: $f(g_D^M(x)) \neq f(g_D^M(y))$ and $|f(g_D^M(x))| \geq |x|$.*

Proof of Claim 3.2.2. Assume that $f(g_D^M(x)) = f(g_D^M(y))$ for some x and y with $x \neq y$ and $|x| = |y| = n \geq n_0$. Let $\tilde{x} = g_D^M(x)$ and $\tilde{y} = g_D^M(y)$. By the definition of g_D^M, it follows that $\tilde{x} = i_{p_1} i_{p_1} x$ and $\tilde{y} = i_{p_2} i_{p_2} y$, where i_{p_1} and i_{p_2} are respectively the $l(x)^{th}$ and $l(y)^{th}$ largest input labels in the set I, the set of input labels of all paths from C_{init}^{5n} to $\tilde{C}$. Since both the paths p_1 and p_2 end at $\tilde{C}$, it follows that their output labels, respectively o_{p_1} and o_{p_2}, will have the same length (recall that a configuration stores the position of the output head also). By the assumption $f(g_D^M(x)) = f(g_D^M(y))$, it follows that $o_{p_1} = o_{p_2}$.

Since there are at least 2^{n+1} labels in I (as $n \geq n_0$) and $x \neq y$, it must hold that $i_{p_1} \neq i_{p_2}$. Without loss of generality, assume that $l(i_{p_1}) > l(i_{p_2})$. Let $z = i_{p_1} \oplus i_{p_2}$. Define strings, $\tilde{x}_1 = i_{p_1} z 1^n$ and $\tilde{y}_1 = i_{p_2} z 1^n$. Since $i_{p_1} \oplus z = i_{p_2}$ and $i_{p_2} \oplus z = i_{p_1}$, it follows, by the definition of *code*(D), that $\tilde{x}_1 \in$ *code*(D) and $\tilde{y}_1 \notin$ *code*(D). However, since $o_{p_1} = o_{p_2}$ and the input to M, computing f on $\tilde{x}_1$ or $\tilde{y}_1$, after it reaches the configuration $\tilde{C}$ is same, it must be that $f(\tilde{x}_1) = f(\tilde{y}_1)$. This contradicts the fact that f is a reduction of *code*(D) to A. Therefore, $o_{p_1} \neq o_{p_2}$. It also follows that $|o_{p_1}| = |o_{p_2}| \geq n$ as there are 2^n different such output labels (one for each string of length n). This implies, firstly that $f(g_D^M(x)) \neq f(g_D^M(y))$ and secondly that $|f(g_D^M(x))| \geq |x|$. □

Now, to get a 1-1, size-increasing reduction of B to A, we use a simple padding technique. Define function r as: $r(0) = n_0$ and $r(m) = q(5.r(m-1)) + 1$ for $m > 0$ (recall that $q(n)$ bounds the number of vertices of G_M^n and therefore also bounds $|f(x)|$ for any x of size n).

Define function g_p, reducing B to D as: $g_p(x) = x10^{k-1}$ where $k = r(l) - |x|$ and $l = \min_m(r(m) > |x|)$. Function g_p can be computed in polynomial-time: start from $r(0)$ and calculate $r(1)$, $r(2)$, ... till an $r(m)$ is obtained with $r(m) > |x|$; compute k and output $x10^{k-1}$. This function maps strings of length between $r(l)$ and $r(l+1) - 1$ to strings of length $r(l+1)$ and therefore, is size-increasing.

Let $h \stackrel{\text{def}}{=} f \circ g_D^M \circ g_p$. We claim that h is the required reduction of B to A. It is clearly a polynomial-time computable reduction of B to A.

Claim 3.2.3 *h is size-increasing.*

Proof of Claim 3.2.3. For every x, $|g_p(x)| \geq r(0) = n_0$ and further, g_p is size-increasing. Now it follows from the Claim 3.2.2 that $|f(g_D^M(g_p(x)))| \geq |g_p(x)| > |x|$. $\square$

Claim 3.2.4 *h is one-one.*

Proof of Claim 3.2.4. Take any two strings x and y. If $|g_p(x)| = |g_p(y)|$ then, by Claim 3.2.2, it follows that $h(x) \neq h(y)$. Now consider the case when $|g_p(x)| < |g_p(y)|$. Let $|g_p(x)| = r(m)$ and $|g_p(y)| = r(n)$ with $n > m$. So, $|h(x)| = |f(g_D^M(g_p(x)))| \leq q(5.r(m))$ (by size bounds of f, g_D^M and g_p) $< r(m+1)$ (by the definition of r) $\leq r(n) = |g_p(y)| \leq |f(g_D^M(g_p(y)))|$ (by Claim 3.2.2) $= |h(x)|$. Therefore, $h(x) \neq h(y)$. $\square$

Claim 3.2.5 *h is p-invertible.*

Proof of Claim 3.2.5. Both the functions g_D^M and g_p are p-invertible. The p-invertibility of f when restricted to the range of $g_D^M \circ g_p$ follows from a result by Allender [4] showing that all 1-1, honest 1-L functions can be inverted in polynomial-time. In the following we give a more direct proof of the invertibility of $f \circ g_D^M$.

Given any z, with $|z| = m$, for each n, $n_0 \leq n \leq m$ do the following. Compute the graph G_M^{5n} and the vertex $\tilde{C}$. Check if there exists a path in G_M^{5n} from the vertex C_{init}^{5n} to some vertex in level $5n+1$ and passing through vertex $\tilde{C}$ whose output label is z (by Claim 3.2.2, there can exist at most one such path). If yes, then consider the subpath, say p, of this path, from C_{init}^{5n} to $\tilde{C}$. Calculate the position of the input label i_p of p in the lexicographic ordering of the input labels of all the paths from C_{init}^{5n} to $\tilde{C}$. Let i_p be the t^{th} largest such label and $t = l(x)$ for some string x. Check if $f(i_p i_p x) = z$. If yes and $|x| = n$ then output x and stop.

If there is no such x for any n then there is no inverse of z. It is easy to see that the above procedure can be carried out in time polynomial in $|z|$ using the procedures defined in the Appendix. $\square$

From the above claims it follows that $B \leq_{1,li,i}^{p} A$ via h. This completes proof of the theorem. ■

Corollary 3.3 *For any class $\mathcal{C}$ closed under log-lin reductions, all 1-L complete sets for $\mathcal{C}$ are p-isomorphic.*

Proof. In [6], it was shown that all sets reducible to each other via $\leq^p_{1,li,i}$-reductions are p-isomorphic. The corollary follows from the above theorem. ■

Corollary 3.4 *1-L complete sets for classes* DLOG*,* NLOG*,* P*,* NP*,* Σ^p_k*,* PSPACE*,* E*,* NE *etc. are p-isomorphic.*

Proof. As all these classes are closed under log-lin reductions, the corollary follows. ■

It has been observed [11] that all 'natural' NP-complete sets are complete under 1-L reductions as well. Therefore, it follows that they are all p-isomorphic to each other, a fact which had earlier been shown using paddability [6].

4 Discussion

In most of the previous results that show complete sets for a certain class to be 1-1 complete, e.g. [5, 17, 8, 12], the following technique is employed in their proofs. To ensure that the reduction from a set B to a complete set A is one-one, an intermediate set I is constructed so that a composition of the reduction from B to I and the reduction from I to A is one-one. The set I is constructed by diagonalizing over all possible reductions to (effectively) ensure that no many-one function can be a reduction of I to A (see, e.g., [8, 12]). However, it has the drawback of making the set I 'hard'. For example, to diagonalize over the class of 1-L reductions superlogarithmic space is needed thus forcing the set I to go beyond DLOG (and possibly even NP) even when B belongs to DLOG. This is one of the reasons why the results in [12] hold only for space classes beyond NLOG.

We have used a different technique in our proof. The outline of our technique is the same as above except that the set I ($= \mathit{code}(D)$) is constructed differently. The set $\mathit{code}(D)$ contains only 'potentially' diagonalizable information. The reduction of B to $\mathit{code}(D)$ exploits this information to force its composition with the reduction of *code*(D) to A to be one-one (and size-increasing). It has the advantage of making the set *code*(D) 'easy enough' to make it fall in the logspace degree of B. This allows our result to hold for any class closed under logspace reductions, e.g., DLOG, NP etc. This technique has been used by Allender [4] as well, though its full potential has not been realized there.

In our technique part of the diagonalization work is transferred to the reduction g $(= g_D^M \circ g_p)$ of B to *code*(D). So, one would expect that now the reduction g will become 'hard'. Indeed, g needs polynomial-time to compute (Claim 3.2.1), and therefore, the one-one reduction of B to A is a polynomial-time function even though A is $\leq_m^{1\text{–}L}$-complete. This was not the case with the earlier technique, where the one-one reduction of B to A remains a 1-L function when A is $\leq_m^{1\text{–}L}$-complete [12]. Therefore, there appears to be a trade-off between the complexities of the set I and the reduction of B to I[1].

From the discussion above, it is clear that our technique is more useful in obtaining isomorphism results than the usual diagonalization technique, at least in the case of weak reductions like 1-L.

5 A generalization

The result in the section 3 can be generalized to functions computed by *finite crossing* machines. A *finite crossing* machine is a logspace bounded TM such that its input head can cross any cell only a constant number of times. TMs computing 1-L functions form a subset of finite crossing TMs as their input head is allowed to cross any cell exactly once. We refer to total functions computed by finite crossing machines as *f-L functions*.

Theorem 5.1 *Let $\mathcal{C}$ be a class closed under log-lin reductions. Then all sets complete for $\mathcal{C}$ under f-L reductions are p-isomorphic to each other.*

The proof of this theorem builds on the proof for 1-L reductions. We omit the details, which can be found in [2].

[1]However, surprisingly, this is not so. In a recent work [1] it is shown that one can construct the set *code*(D) and reduction g such that g remains a 1-L function. In fact, it is shown there that not only is the one-one reduction of B to A a 1-L function, it is 1-L-invertible as well, thus providing a very strong collapse of 1-L-complete degrees.

The class of f-L functions contains the class of total functions computed by 2-way DFA machines. A *2-way DFA* machine is a TM that has no workspace. Therefore, the input head of any such machine computing a total function cannot cross any cell more than a constant number of times (equal to the number of states of the TM). Therefore, we have

Corollary 5.2 *Let $\mathcal{C}$ be a class closed under log-lin reductions. Then all sets complete for $\mathcal{C}$ under 2-way DFA reductions are p-isomorphic to each other.*

The above corollary generalizes some earlier results on 2-way DFA functions: Allender [4] had shown that all complete sets for E under 2-way DFA reductions are p-isomorphic while Ganesan and Homer [8] did the same for the class NE.

6 Conclusion

Our results here generalize the results of Allender [4] and Ganesan and Homer [8]. In comparison with the results of Hemachandra and Hoene [12], our results are weaker but hold for a larger number of classes ([12] shows that for any reasonable nondeterministic space class above NLOG, $\leq_m^{1-L}$-complete sets are also complete under one-one and size-increasing 1-L reductions).

One can consider some other natural classes of weak reductions and address the isomorphism conjecture weakened to them. For example, in [3], it has been shown that all complete sets under *first order projections* for reasonable classes are first order isomorphic. One may try to generalize this result to prove it for all first order computable (equivalently, *uniform-AC^0*) reductions or at least, for all *uniform-NC^0* reductions.

Acknowledgment: We thank the anonymous referees for many helpful suggestions and improvements.

References

[1] M. Agrawal. On the isomorphism problem for weak reducibilities. In *Proceedings of the Structure in Complexity Theory Conference*, pages 338–355, 1994.

[2] M. Agrawal and S. Biswas. Isomorphism of f-L-complete sets. Technical report, Indian Institute of Technology, Kanpur, India, 1993.

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

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

[7] 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.

[8] 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.

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

[10] J. Hartmanis and L. Hemchandra. One-way functions and the non-isomorphism of NP-complete sets. *Theoretical Computer Science*, 81(1):155–163, 1991.

[11] 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.

[12] 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.

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

[14] 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.

[15] 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.

[16] L. J. Stockmeyer. *The Complexity of Decision Problems in Automata Theory and Logic*. PhD thesis, Massachusetts Institute of Technology, 1974.

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

Appendix

```
procedure number-of-paths;

begin
/* $P[u,v]$ is the number of distinct paths from $u$ to $v$ */
for all $u, v \in V^{5n}$ do      /* initialization */
        if $u = v$ then
                $P[u,v] = 1$;
        else
                $P[u,v] = 0$;

for all $i = 1, 5n$ do
        /* compute the number of paths between every pair of vertices
         * that are $i$ levels apart in $G_M^{5n}$ */
        for all $j = 1, 5n - i + 1$ do
                /* compute the number of paths between vertices at level $j$ and $j + i$ */
                for all $u \in L_j$ and $v \in L_{j+i}$ do
                        /* compute $P[u,v]$ */
                        Let $v_1, v_2 \in L_{j+1}$ such that $\langle u, v_1 \rangle, \langle u, v_2 \rangle \in E^{5n}$
                            with input label 0 and 1 respectively;
                        $P[u,v] = P[v_1,v] + P[v_2,v]$;
end;
```

Figure 1: Procedure to compute the number of paths between vertices of G_M^{5n}

```
function configuration;

begin
number-of-paths;
/* At this point the array $P$ is defined */
Find the lexicographically smallest configuration $\tilde{C} \in L_{2n+1}$
    such that $P[C^{5n}_{init}, \tilde{C}] \geq 2^{2n}/q(5n)$;
return $\tilde{C}$;
end;
```

Figure 2: Function to compute the configuration $\tilde{C}$ of G^{5n}_M

```
function pathno(k);

begin
number-of-paths;
/* At this point the array $P$ is defined */
$\tilde{C}$ = configuration;
/* $P[C^{5n}_{init}, \tilde{C}] \geq 2^{2n}/q(5n)$ */
$I = \epsilon$;          /* length 0 prefix of the label of the $k^{th}$ largest path */
$v = C^{n}_{init}$;     /* first vertex on the $k^{th}$ largest path */
$l = k$;          /* $l^{th}$ largest path from $v$ is to be chosen */

for all $i = 1, 2n$ do
        /* Invariants: $v$ is the $i^{th}$ vertex on the $k^{th}$ largest path;
         *                $I$ is the length $i-1$ prefix of the label of the $k^{th}$ largest path;
         *                $k^{th}$ largest path is the $l^{th}$ largest path from $v$ */
        Let $v_1, v_2 \in L_{i+1}$ such that $\langle v, v_1\rangle, \langle v, v_2\rangle \in E^{5n}$ with input label 0 and 1 respectively;
        if $P[v_1, \tilde{C}] \geq l$ then        /* $l^{th}$ largest path occurs in the $\langle v, v_1\rangle$ branch */
                $I = I \cdot 0$;
                $v = v_1$;
        else     /* $l^{th}$ largest path occurs in the $\langle v, v_2\rangle$ branch */
                $I = I \cdot 1$;
                $v = v_2$;
                $l = l - P[v_1, \tilde{C}]$;

return $I$;
end;
```

Figure 3: Function to compute the input label of k^{th} largest path from C^{5n}_{init} to $\tilde{C}$