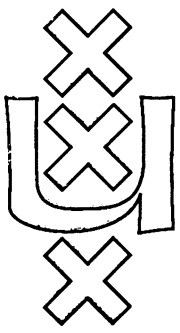
Institute for Language, Logic and Information

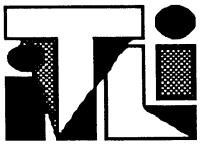
ON ADAPTIVE RESOURCE BOUNDED COMPUTATIONS

Harry Buhrman
Edith Spaan
Leen Torenvliet

ITLI Prepublication Series
for Computation and Complexity Theory CT-89-09



University of Amsterdam



Instituut voor Taal, Logica en Informatie
Institute for Language, Logic and Information

Faculteit der Wiskunde en Informatica
(Department of Mathematics and Computer Science)
Plantage Muidergracht 24
1018TV Amsterdam

Faculteit der Wijsbegeerte
(Department of Philosophy)
Nieuwe Doelenstraat 15
1012CP Amsterdam

ON ADAPTIVE RESOURCE BOUNDED COMPUTATIONS

Harry Buhrman
Edith Spaan
Leen Torenvliet
Department of Mathematics and Computer Science
University of Amsterdam

Received November 1989

Abstract

We investigate differences in computational power of logspace bounded adaptive and non-adaptive oracle machines. Our main results are that for any two sets A and B if $A \leq_T B$ via a logarithmic space bounded machine then $A \leq_{tt} B$ via a $\log^2$ space bounded machine. Under the very strong assumption that $NC_1 \neq LOG/poly$ we demonstrate a difference between the two investigated reductions. We show that there exist recursive sets A and B such that $A \leq_T^{logspace} B$ while $A \not\leq_{tt}^{logspace} B$. Using the natural limit on the number of different queries that can be asked by space bounded machines, we obtain a separation between logarithmic space bounded – and polynomial time bounded Turing reductions. As an exponential time complete set can be encoded in the constructed set we obtain as a corollary differences between the corresponding completeness notions on classes containing exponential time.

1 Introduction

Efficient reducibilities are a central object of study in computational complexity theory. Since the first use of polynomial time bounded Turing reductions by Cook [6] and the introduction of polynomial time bounded many-one reductions by Karp [8], considerable effort has been put in the investigation of properties and the relative strengths of different reductions. In 1975 an extensive survey of different types of reductions and differences between these reductions on exponential time bounded classes was given by Ladner, Lynch and Selman [10]. In 1987 Watanabe [12] building upon earlier work of Berman [2] extended this work by giving proving also almost all differences between complete sets under the different reductions on exponential bounded and larger deterministic time classes.

Logspace reductions have long been studied as strengthenings of polynomial-time reductions. It was realized early-on that many NP-complete sets (with respect to many-one polynomial time reductions) were in fact logspace complete for NP [8,9]. Shortly after the initial work on the isomorphism conjecture by Berman and Hartmanis [3], Hartmanis [7] studied the question of the logspace isomorphism of NP-complete sets and achieved many of the same results as in [3] for logspace reductions.

An important advantage of logspace reductions is that they give rise to the definition and study of complete sets for smaller complexity classes such as NL, CSL and P. The properties of such complete sets have consequences for parallel algorithms and the study of parallel complexity classes. Nonetheless, logspace reductions have been much less studied than have polynomial-time reductions.

Buhrman et al. showed in [5] that the differences obtained by Watanabe for polynomial time bounded reductions on deterministic exponential time could also be obtained for nondeterministic exponential time classes. Moreover the constructions presented in that paper also suffice to demonstrate these differences for logarithmic space bounded reductions.

They differentiated on logarithmic space bounded reductions especially in nondeterministic space classes. They left open however the question of differentiating between logarithmic space bounded Turing reductions and logarithmic space bounded truth-table reductions. The techniques presented in their paper seem inappropriate to attack this problem. As we show in the present paper these two types of reductions are closely interrelated. As a rather straightforward application of Savitch' theorem shows, it is possible to produce in squared logarithmic space a formula which is true if and only if a given logarithmic space bounded oracle machine accepts its input. Therefore in any case the adaptive use of queries cannot give more than a quadratic increase of computational power over non adaptive use of queries. It is at present unknown whether the quadratic overhead also presents a lowerbound for the simulation. We can however under the assumption that $NC_1 \neq LOG/poly$ show that $\leq_T^{logspace} \neq \leq_{tt}^{logspace}$ and therefore elimination of the square would entail $NC_1 = LOG/poly$.

2 Preliminaries

Let $\Sigma = \{0,1\}$. String are elements of Σ^*, and are denoted by small letters $x, y, u, v, \ldots$. For any string x the length of a string is denoted by $|x|$. Languages are subsets of Σ^*, and are denoted by capital letters $A, B, C, \ldots$. Tally sets are subsets of $\{0\}^*$. For any set S

the cardinality of S is denoted by $|S|$. We assume a bijective pairing function computable in logarithmic space and/or polynomial time—as for example described in [1] (pp. 7,8)—which makes it possible not only to encode pairs of strings into a single string but also to encode sequences of *varying* length into a single string.

We assume that the reader is familiar with the standard Turing machine model. An *oracle* machine is a multi-tape Turing machine with an input tape, an output tape, work-tapes and a *query* tape. Oracle machines have three distinguished states QUERY, YES and NO, which are explained as follows. At some stage(s) in the computation the machine may enter the state QUERY and then enters the state YES or enters the state NO depending on the membership of the string currently written on the query tape in a fixed *oracle* set.

Oracle machines appear in the paper in the two flavors: adaptive and non-adaptive. For a non-adaptive machine queries may not be interdependent, whereas an adaptive machine may compute a next query depending on the answer to previous queries. We assume an enumeration of logspace bounded non-adaptive oracle machines and denote this enumeration by $M_1, M_2, \ldots$. Without loss of generality we furthermore assume that M_i operates in space bound $i \times \log n$. The class of languages $\{L | \exists i L = L(M_i)\}$ is denoted LOG. A function f is in the class *poly* iff for some polynomial p:$\forall x.|f(x)| \leq p(|x|)$.

We use $M^A(x)$ to denote the computation of M on input x relative to oracle A. We will use the notation $M^A(x) = 0$ for rejecting –, and $M^A(x) = 1$ for accepting computations. For a Turing machine M, $L(M)$ denotes the set of strings accepted by M. For an oracle machine M and set A, $L(M, A)$ denotes the set of strings accepted by M relative to oracle A. These sets are also called the *language* of M and the language of M^A respectively. The pair $<<a_1, \ldots, a_k>, \alpha>$ is called a *truth-table condition of norm k* if $<a_1, \ldots, a_k>$ is a k-tuple ($k > 0$) of strings, and α is a k-ary Boolean function [10]. The set $\{a_1, \ldots, a_k\}$ is called the *associated set* of the tt-condition. A function f is a *truth-table function* if f is total and $f(x)$ is a truth-table condition for every x in Σ^*.

Let the resource bound b be either polynomial-time (p) or logarithmic-space (*logspace*) and $A_1, A_2 \subseteq \Sigma^*$. We say that:

1. A_1 is b truth-table reducible ($\leq^b_{tt}$-reducible) to A_2 iff there exists a b-bounded tt-function f such that $\alpha(\chi_{A_2}(a_1), \ldots, \chi_{A_2}(a_k)) =$ **true** iff $x \in A_1$, where $f(x)$ is $<<a_1, \ldots, a_k>, \alpha>$ and χ_{A_2} is the characteristic function of the set A_2. As b-bounded functions can be computed by b-bounded Turing machines, the truth table conditions are often modeled by non-adaptive oracle machines.

2. A_1 is b Turing reducible to A_2 ($\leq^b_T$-reducible) to A_2 if there exists a b-bounded deterministic oracle machine such that $A_1 = L(M, A_2)$.

3 A Savitch' theorem for reductions

In this section we show that logspace Turing reductions can be simulated by $\log^2$space truth table reductions. In fact we obtain a more general theorem. We show that $S^2(n)$ space suffices for the computation of a truth table function which evaluates to true on precisely those arguments that are accepted by an $S(n)$ space oracle machine.

Theorem 1 *Let $S(n) \geq \log n$ be a space-constructible function. If $A \leq_T^{space(S(n))} B$ then $A \leq_{tt}^{Space(S^2(n))} B$ for any A and B.*

Proof: Let $A = L(M, B)$ with M an $S(n)$ space-bounded oracle machine. The number of different configurations (or instantaneous descriptions (ID's)) of M on a given input of length n is bounded by $2^{c.S(n)}$. We identify these configurations with the numbers $1, 2, \ldots, 2^{c.S(n)}$, where 1 denotes the initial configuration. Let q_i be the string written on the oracle tape if we start the machine in configuration i. Note that different configurations may lead to the same query. Given a computation, we uniquely identify a query asked at time t with the number of the configuration at the earliest time $t' \leq t$ such that no queries are asked in the interval between t' and t. (This will be either the initial configuration or a configuration immediately following a query.) We construct a recursive procedure *writeform*(i, j, t) that works as follows: it writes a formula $\phi(x_1, x_2, \ldots, x_{2^{c.S(n)}})$ on the oracle tape such that $\phi(x_1, x_2, \ldots, x_{2^{c.S(n)}}) =$ **true** iff $\exists r \leq 2^t$ such that queries $q_{j_1}, \ldots, q_{j_r}$ are exactly the queries asked in the computation with answers $a_{j_1}, \ldots, a_{j_r}$. Define a function *reachable* such that *reachable*$(i, j) =$ **true** iff configuration j is reachable from configuration i without asking queries.

```
procedure writeform (i, j, t)
if t = 0 then
    if reachable(i, j) then write ('true')
    else if there exist k, k' such that:
            reachable(i, k), state (k) = QUERY,
            k' reachable in one step from k and reachable (k', j) then
                if state (k') = YES then write ('x_i')
                else write ('¬x_i')
                end if
        else write ('false')
        end if
    end if
else for each ID k such that k = 1 or state (k) ∈ {YES,NO} do
        if not first time then write ('∨')
        write ('(')
        writeform (i, k, t − 1)
        write ('∧')
        writeform (k, j, t − 1)
        write (')')
    end for
end if
end
```

Since the number of queries in a computation is bounded by $2^{c.s(n)}$, we can construct the truth table function as follows : write $q_1, \ldots q_{2^{c.s(n)}}$ on the oracle tape, and write the conjunction of *writeform* $(1, j)$ for each accepting ID j.

for $i := 1$ to $\#IDs$ do
write $(query_i)$
for each accepting ID i do
if not first time then *write* ('$\vee$')
write ('(')
*write*form $(1, i, c.S(n))$
write (')')
end for

$\boxtimes$

The crucial difference between polynomial time bounded Turing reductions and logarithmic space bounded Turing reductions appears in the proof above. The number of possible queries asked of the oracle by a logarithmic space bounded reductor is bounded by a polynomial (The exponential function of its space bound) whereas the total number of queries in the tree of possible computations of a polynomial time bounded Turing reductor is exponential.

We can use this to obtain a real difference between polynomial time bounded Turing reductions and logarithmic space bounded Turing reductions. In particular we construct two sets A and B such that there exists a polynomial time bounded Turing reduction from A to B but by diagonalization against all logarithmic space bounded Turing reductors we ensure that no logarithmic space bounded Turing reduction from A to B can exist

For input x and oracle A let $Q(M_n, A, x)$ be the set of queries made during the computation of M_n^A on input x. Let $Q(M_n, x) = \cup_{A \subseteq \Sigma^*} Q(M_n, A, x)$.

Theorem 2 *There exists sets A and B such that $A \leq_T^p B$ but not $A \leq_T^{logspace} B$*

Proof: Since a logspace oracle machine M on input x can ask only a polynomial number of queries there exists a string of length $|x|$ which will not be queried by M for large enough x . We construct set A and B in stages.
Construction:
Initially $A = B = \emptyset$ and $b(0) = 0$.
stage n:

Let M_n be the n^{th} logspace Turing reduction. Let c be the first natural number such that $2^{nlog(c)} < 2^c$ and $c > b(n-1)$. Compute $Q(M_n, 0^c)$. Since M_n uses $nlog(c)$ space there exists a string y of length c such that $y \notin Q(M_n, 0^c)$.

1. We code y into B by putting the pair $< i, 0^c >$ in B iff the i^{th} bit of $y = 1$.
2. Simulate M_n on input 0^c with B as an oracle.
3. Put 0^c in A iff M_n rejects.
4. Put y in B iff M_n rejects.
5. $b(n) = 2^{nlog(c)} + 1$.

end of stage n

CLAIM 2.1 $A \leq_T^p B$

Proof: Note that 0^c is in A iff y is in B. Now we give an algorithm for the reduction. On input 0^a for some a, query $<0,0^a>,<1,0^a> \ldots <|a|,0^a>$. According to the answers it is possible to recover y ($<i,0^a> \in B$ iff i^{th} bit of $y = 1$). Next we query y to B and accept iff $y \in B$. Clearly this algorithm satisfies the claim ⊠

CLAIM 2.2 $A \not\leq_T^{logspace} B$

Proof: Suppose $A \leq_T^{logspace} B$ by machine M_n. Then there exists a c such that $0^c \in A$ and M_n rejects. (item 3 in the above construction). Since y in this stage of the construction is not queried by M_n the presence or absence in B will not change the computation of M_n furthermore function b ensures that in later stages of the construction this part of B will remain unchanged. ⊠

This completes the proof of Theorem 2. ⊠

With a technique similar to [5] we can also separate the corresponding completeness notions on DTIME $\left(2^{poly}\right)$of $\leq_T^p$ and $\leq_T^{logspace}$. To do this we encode K (a $\leq_m^{logspace}$ -complete set for DTIME $\left(2^{poly}\right)$) in B.

We need a set of elements on which to diagonalize. To this end we define a sequence of integers $\{u_n\}_n$ by $u_0 = u_1 = 1$, $u_m = 2^{(u_{m-1})^{m-1}} + 1$, for $m > 1$.

Corollary 1 *There is a set B which is $\leq_T^p$ -complete but not $\leq_T^{logspace}$ -complete for* DTIME $\left(2^{poly}\right)$.

Proof: We construct a set B in stages, and simultaneously a set $W \in$ DTIME $\left(2^{poly}\right)$, with the property that W is not $\leq_T^{logspace}$ reducible to B. If we are now able to show that B is $\leq_T^p$ -complete for DTIME $\left(2^{poly}\right)$. by giving a $\leq_T^p$ reduction from K to B, we have proved the desired separation result.

In stage n of the construction of B we do the following: We take the n^{th} logspace Turing reduction and consider all the queries to be made on input 0^{u_n} and choose a string y with length u_n, such that neither y nor any longer strings with y as a prefix are queried. Since there are exponentially many strings and only a polynomial number of strings can be queried, such a y exists. Now we code the bits of y in B, by puting $<0^{u_n},i>$ in B iff the i^{th} bit of y is 1. Furthermore for all $x \in K$ that are in the interval we use currently to diagonalize (i.e. $0^{(u_{n-1})^{n-1}} \leq x \leq 0^{u_n{}^n}$) we put the pair $<y,x>$ in B if $0^{u_n} \leq x \leq 0^{u_n{}^n}$ else we put x in B. In this way it is possible for a Turing reduction on input x to first (if necessary) recover y and then query the oracle B about $<y,x>$ and thus we have a Turing reduction from K to B. After we did this coding, i.e. putting $<0^{u_n},i>$ in B iff the i^{th} bit of y is 1, we look at the behavior of the n^{th} logspace Turing reductor. Thus we compute the answers to the oracle queries and look if this reductor accepts or rejects. Now we put 0^{u_n} in W iff the logspace Turing reductor rejects. By doing this we ensure that W is *not* logspace Turing reducible to B. From this we may conclude that B is not logspace Turing complete for DTIME $\left(2^{poly}\right)$. Making this idea more precise we have the

following construction:
Initially $B = W = \emptyset$
stage n:

Let M_n be the n^{th} logspace Turing reduction. Compute $Q(M_n, 0^{u_n})$ this is the set of all possible queries of M_n on input 0^{u_n}. Let y be the first string (in lexicographic order) with $|y| = u_n$ and for all $v \in \Sigma^*$ $yv \notin Q(M_n, 0^{u_n})$.

1. Put $<0^{u_n}, i>$ in B iff the i^{th} bit of y is 1.
2. Put all the pairs $<y, x>$ with $0^{u_n} \leq x \leq 0^{u_n{}^n}$ and $x \in K$ in B.
3. Put all the $<x>$ with $x \in K$ and $0^{(u_n-1)^{n-1}} \leq x, <0^{u_n}>$ in B.

   Now we determine whether M_n rejects or accepts on input 0^{u_n}, by actually computing the answers to the queries made by M_n according to the previously added strings.
4. Finally we put 0^{u_n} in W iff $M_n(0^{u_n})$ rejects.

end of stage n

CLAIM 2.3 $B \in \text{DTIME}\left(2^{poly}\right)$

Proof: We proof this by giving a deterministic exponential time algorithm for B. All the elements in B have the following form:

1. $<0^{u_n}, i>$
2. $<y, x>$
3. $<x>$ (if $x \in K$)

Since the sequence $\{u_n\}$ is in P, it is easy to determine, on input x, whether x is in $[0^{(u_n-1)^{n-1}}, 0^{u_n}>$ or in $[0^{u_n}, 0^{u_n{}^n}]$. for appropriate n. In case 1 we recover n from u_n, simulate M_n on input 0^{u_n} to recover the string y and check that the i^{th} bit of y is 1. In case 2 we have that $|y| = u_n$ so that we can check y by simulation of M_n as above. We check that $|z| \leq |y|^n$ and that $z \in K$. Finally in case 3 we check that $(u_{n-1})^{n-1} \leq |x| < u_n$ and that $x \in K$. Clearly all these actions can be performed in exponential time. ⊠

CLAIM 2.4 $W \in \text{DTIME}\left(2^{poly}\right)$

Proof: Since $B \in \text{DTIME}\left(2^{poly}\right)$it is easy to see that W is also $\in \text{DTIME}\left(2^{poly}\right)$. ⊠

Next we prove that B is not $\leq_T^{logspace}$ -complete for $\text{DTIME}\left(2^{poly}\right)$, by showing that W is not $\leq_T^{logspace}$ B. Suppose W is logspace Turing reducible to B via M_n. Now on input 0^{u_n} machine M_n^B accepts iff 0^{u_n} is not in W, a contradiction with the fact that M_n is a $\leq_T^{logspace}$ reduction from W to B. To complete the proof we show that B is $\leq_T^p$ -complete for $\text{DTIME}\left(2^{poly}\right)$by giving a $\leq_T^p$ reduction from K to B. On input w there are two possibilities. First x is in the interval $[0^{(u_n-1)^{n-1}}, 0^{u_n}>$. If this is the case we simply query

B about x and accept if $x \in B$. Second x is in the interval $[0^{u_n}, 0^{u_n{}^n}]$. Now we we recover y by querying B about $<0^{u_n}, i>(0 \leq i \leq u_n)$. Next we query B about $<y, x>$ and accept if $<y, x> \in B$. This completes the proof. ⊠

Standard padding techniques give that the set B can be used for also obtaining a separation of the completeness notions on $DEXT$. A similar approach as taken in [5] yields that the entire construction can be carried out for a complete set in $NEXT$. A proof of this will appear in the final paper.

4 Logarithmic adaptivity

We now turn our attention to the relation between logarithmic space bounded Turing reductions and logarithmic space bounded truth table reductions. It is already known that limiting the size of the truth table in any way gives a separation from the general truth table reducibility and therefore a forteriori a separation from Turing reductions [4]. Moreover since said separations can be proved on $S(n)$ space bounded classes for $S(n) > \log n$ this gives a separation of the corresponding completeness notions.

Since the polynomial size of the set of possible queries asked by a logspace bounded Turing reductor does not allow for a separation as in Theorem 2 we need rather a strong assumption to achieve the separation. One of the key observations in the proof below is that although a logarithmic space bounded truth table reductor may ***ask*** all the same queries of the oracle that are asked by the polynomial size Turing reductor, it has no room to *store* these answers, and repetition of queries is not allowed. For a tally set B let $B \upharpoonright n$ be a string $\in 0,1^n$ such that: $(B \upharpoonright n)_i = 1$ iff $0^i \in B(1 \leq i \leq n)$, where $(B \upharpoonright n)_i$ is the i^{th} bit of the string $(B \upharpoonright n)$.

Theorem 3 *If $NC_1 \neq LOG/poly$ then $\leq_{tt}^{logspace} \neq \leq_T^{logspace}$.*

Proof: The proof is given by a series of three lemmas and a diagonalization. First we show that the assumption leads to a set in $LOG - NC_1$.

Lemma 3.1 *If $NC_1 \neq LOG/poly$ then $\exists A \in LOG$ such that $A \notin NC_1$.*

Proof: Let $B \in LOG/poly, B \notin NC_1$. Then there exist $A \in LOG, f \in poly$ such that : $x \in B \iff <x, f(|x|)> \in A$. Suppose $A \in NC_1$. Fix n and look at the $|<x, f(n)>|$-th circuit for A with inputs $x_1, \ldots, x_n, y_1 \ldots, y_{|f(n)|}$. We construct circuit C_n for B with inputs $x_1, \ldots, x_n$ as follows : set input y_i to true if $f(n)_i = 1$, and to false otherwise. Size $(C_n) = q(|x| + f(|x|)) \leq q(n + p(n))$. Depth $(C_n) = c.\log(|x| + f(|x|) \leq c.\log(n + p(n))$. Therefore $B \in NC_1$ which contradicts our assumption. ⊠

This observation can be extended to the construction of a set A in LOG such that the set consisting of the suffixes of the strings in A is also not in NC_1.

Lemma 3.2 *If $NC_1 \neq LOG/poly$ then $\exists A \in LOG$ such that $\forall k, \forall a \in \{0,1\}^k : \{y | ay \in A\} \notin NC_1$.*

Proof: By Lemma 3.1, there exists a set $B \in LOG, B \notin NC_1$. Define $A := \{0^n y : |y| = n, y \in B\}$. Suppose $\exists k, a \in \{0,1\}^k$ such that $\{y : ay \in A\} \in NC_1$. Then $\{0^{n-k} y : |y| = n, y \in B\} \in NC_1$. But these circuits can be viewed as circuits for B as well. ⊠

Let A be a set as in Lemma 3.1. For each tally set B, define the tally language L_B as follows: $0^n \in L_B$ iff $B \upharpoonright n \in A$. Then for any tally set B: $L_B \leq_T^{logspace} B$.

The machine M reducing L_B to B works as follows: On input 0^n, M starts simulating the program for A as if started on $B \upharpoonright n$. If M needs to read the i^{th} input bit it asks a query 0^i of the oracle B for $i \leq n$ or proceeds as if it has read a blank.

Since A operates in logarithmic space, and the necessary queries of length $\leq n$ only involve a counter, it is now clear that M operates in logarithmic space. (The fact that the simulated machine may leave the input to the right reading blanks for an indeterminate period does not lead to complications here.)

Next we demonstrate the existence of a tally set B such that $L_B \not\leq_{tt}^{logspace} B$. The proof consists of a "room to diagonalize" lemma and the diagonalization itself.

Lemma 3.3 $\forall f \in logspace \forall k, \forall a \in \{0,1\}^k : \exists tally B : B \upharpoonright k = a \wedge L_B \not\leq_{tt}^{logspace} B$ *via f.*

Proof: Suppose $\exists f, k, a \in \{0,1\}^n$ such that $\forall$ tally $B : B \upharpoonright k = a \Rightarrow L_B \leq_{tt}^{logspace} B$ via f.

This leads to polynomial size formulas for $\{y : ay \in A\}$. We argue that it is no loss of generality to assume these formulas are also on arguments of length $\leq$ the length of the input and therefore (cf. [11]) $\{y : ay \in A\} \in NC_1$, contradicting the properties of A.

Suppose f on input 0^a generates the condition $\phi\left(\chi_B(0^{b_1}), \chi(0^{b_2}), \ldots, \chi(0^{b_c})\right)$, which is of polynomial size since it is generated by a logspace bounded machine. Then for $i = 1, \ldots, c : b_i \leq m$. Since $0^m \in L_B$ depends only on $B \upharpoonright m$ and the condition is assumed to hold for *any* tally set $\chi_B(0^d)$ *cannot* influence the value of the condition if $d > m$. ⊠

Lemma 3.3 states that for any candidate truth table function f and initial segment of the tally sets C and D we can find extensions of C and D such that f cannot generate the truth table reduction from C to D. This can be used to obtain tally sets for which no function can generate the reduction:

Construct the following tally set C:

stage 0: $C := \emptyset; k := 0;$

stage s: (Diagonalizing against f_s).
$\exists D : D \upharpoonright k = C \upharpoonright k, L_D \not\leq_{tt}^{logspace} D$ via f_s.
Take some $x, |x| > k$ such that x is counterexample for reduction.
$k := 2^{|x|}, C := D^{\leq k}$

Now $L_C \not\leq_{tt}^{logspace} C$ as required. ⊠

The reader will have noted that the proof of this theorem presents another difference between logspace and polynomial time bounded reductions. Although it is straightforward to show that for any tally set B and set A if $A \leq_T^p B$ then also $A \leq_{tt}^p B$ (cf. [1] exc. 4.6.10) the separation presented above was done with a tally oracle set. The main difference between polynomial space bounded oracle machines and polynomial time bounded oracle machines is again that although the logarithmic space bounded machine may *ask* all possible questions, it cannot remember the answers long enough to profit from them in the course of the computation. This is perhaps an important breakpoint in the relation between time and space bounded computations.

Bibliography

[1] Balcázar J.L., J. Díaz & J. Gabarró. *Structural Complexity I.* W. Brauer, G. Rozenberg & A. Salomaa (eds.) EATCS Monographs on Theoretical Computer Science 11 (1988) Springer Verlag.

[2] Berman L. *On the structure of complete sets.* Proc. 17th IEEE conference on Foundations of Computer Science (1976) pp76-80.

[3] Berman, L. & J. Hartmanis. *On isomorphisms and density of NP and other complete sets.* SIAM J. on Computing 1 (1977) pp. 305–322.

[4] Buhrman H. & L. Torenvliet. *A comparison of reductions on nondeterministic space* Proc. 4th CSN conference (1989) pp89–102.

[5] Buhrman H., S. Homer & L. Torenvliet. *Honest reductions, completeness and nondeterministic complexity classes.* Report CT-89-08, University of Amsterdam, Dept. of Computer Science.

[6] Cook, S. A. *The complexity of theorem-proving procedures.* Proc. 3d ACM Symp. on Theory of Computing, Assoc. for Computing Machinery, New York (1971) pp. 151–158.

[7] Hartmanis, J. *On the logtape isomorphism of complete sets* Theoretical Computer Science 7 (1978) pp. 273–286.

[8] Karp, R.M. *Reducibility among combinatorial problems.* Complexity of Computer Computations, R.E. Miller & J.W. Thatcher eds. Plenum N.Y. pp. 85–103.

[9] Jones, N. *Space bounded reducibilities among combinatorial problems* J. Comp. System Sci. 11 (1975) pp. 68–85.

[10] Ladner, R.E., N. Lynch & A.L. Selman. *A comparison of polynomial time reducibilities.* Theoretical Computer Science 1 (1975) pp. 103–123.

[11] Spira P.M. *On time-hardware complexity tradeoffs for Boolean functions.* Proceedings of 4th Hawaii Symposium on System Sciences, Western Periodicals Company, North Hollywood (1971) pp. 525–527.

[12] Watanabe, O. *A comparison of polynomial time completeness notions.* Theoretical Computer Science 54 (1987) pp. 249–265.

The ITLI Prepublication Series

1986

86-01 The Institute of Language, Logic and Information
86-02 Peter van Emde Boas — A Semantical Model for Integration and Modularization of Rules
86-03 Johan van Benthem — Categorial Grammar and Lambda Calculus
86-04 Reinhard Muskens — A Relational Formulation of the Theory of Types
86-05 Kenneth A. Bowen, Dick de Jongh — Some Complete Logics for Branched Time, Part I Well-founded Time, Forward looking Operators
86-06 Johan van Benthem — Logical Syntax

1987

87-01 Jeroen Groenendijk, Martin Stokhof — Type shifting Rules and the Semantics of Interrogatives
87-02 Renate Bartsch — Frame Representations and Discourse Representations
87-03 Jan Willem Klop, Roel de Vrijer — Unique Normal Forms for Lambda Calculus with Surjective Pairing
87-04 Johan van Benthem — Polyadic quantifiers
87-05 Víctor Sánchez Valencia — Traditional Logicians and de Morgan's Example
87-06 Eleonore Oversteegen — Temporal Adverbials in the Two Track Theory of Time
87-07 Johan van Benthem — Categorial Grammar and Type Theory
87-08 Renate Bartsch — The Construction of Properties under Perspectives
87-09 Herman Hendriks — Type Change in Semantics: The Scope of Quantification and Coordination

1988

Logic, Semantics and Philosophy of Language:

LP-88-01 Michiel van Lambalgen — Algorithmic Information Theory
LP-88-02 Yde Venema — Expressiveness and Completeness of an Interval Tense Logic
LP-88-03 — Year Report 1987
LP-88-04 Reinhard Muskens — Going partial in Montague Grammar
LP-88-05 Johan van Benthem — Logical Constants across Varying Types
LP-88-06 Johan van Benthem — Semantic Parallels in Natural Language and Computation
LP-88-07 Renate Bartsch — Tenses, Aspects, and their Scopes in Discourse
LP-88-08 Jeroen Groenendijk, Martin Stokhof — Context and Information in Dynamic Semantics
LP-88-09 Theo M.V. Janssen — A mathematical model for the CAT framework of Eurotra
LP-88-10 Anneke Kleppe — A Blissymbolics Translation Program

Mathematical Logic and Foundations:

ML-88-01 Jaap van Oosten — Lifschitz' Realizabiility
ML-88-02 M.D.G. Swaen — The Arithmetical Fragment of Martin Löf's Type Theories with weak Σ-elimination
ML-88-03 Dick de Jongh, Frank Veltman — Provability Logics for Relative Interpretability
ML-88-04 A.S. Troelstra — On the Early History of Intuitionistic Logic
ML-88-05 A.S. Troelstra — Remarks on Intuitionism and the Philosophy of Mathematics

Computation and Complexity Theory:

CT-88-01 Ming Li, Paul M.B.Vitanyi — Two Decades of Applied Kolmogorov Complexity
CT-88-02 Michiel H.M. Smid — General Lower Bounds for the Partitioning of Range Trees
CT-88-03 Michiel H.M. Smid, Mark H. Overmars Leen Torenvliet, Peter van Emde Boas — Maintaining Multiple Representations of Dynamic Data Structures
CT-88-04 Dick de Jongh, Lex Hendriks Gerard R. Renardel de Lavalette — Computations in Fragments of Intuitionistic Propositional Logic
CT-88-05 Peter van Emde Boas — Machine Models and Simulations (revised version)
CT-88-06 Michiel H.M. Smid — A Data Structure for the Union-find Problem having good Single-Operation Complexity
CT-88-07 Johan van Benthem — Time, Logic and Computation
CT-88-08 Michiel H.M. Smid, Mark H. Overmars Leen Torenvliet, Peter van Emde Boas — Multiple Representations of Dynamic Data Structures
CT-88-09 Theo M.V. Janssen — Towards a Universal Parsing Algorithm for Functional Grammar
CT-88-10 Edith Spaan, Leen Torenvliet, Peter van Emde Boas — Nondeterminism, Fairness and a Fundamental Analogy
CT-88-11 Sieger van Denneheuvel, Peter van Emde Boas — Towards implementing RL

Other prepublications:

X-88-01 Marc Jumelet — On Solovay's Completeness Theorem

1989

Logic, Semantics and Philosophy of Language:

LP-89-01 Johan van Benthem — The Fine-Structure of Categorial Semantics
LP-89-02 Jeroen Groenendijk, Martin Stokhof — Dynamic Predicate Logic, towards a compositional, non-representational semantics of discourse
LP-89-03 Yde Venema — Two-dimensional Modal Logics for Relation Algebras and Temporal Logic of Intervals
LP-89-04 Johan van Benthem — Language in Action
LP-89-05 Johan van Benthem — Modal Logic as a Theory of Information
LP-89-06 Andreja Prijatelj — Intensional Lambek Calculi: Theory and Application
LP-89-07 Heinrich Wansing — The Adequacy Problem for Sequential Propositional Logic

Mathematical Logic and Foundations:

ML-89-01 Dick de Jongh, Albert Visser — Explicit Fixed Points for Interpretability Logic
ML-89-02 Roel de Vrijer — Extending the Lambda Calculus with Surjective Pairing is conservative
ML-89-03 Dick de Jongh, Franco Montagna — Rosser Orderings and Free Variables
ML-89-04 Dick de Jongh, Marc Jumelet, Franco Montagna — On the Proof of Solovay's Theorem
ML-89-05 Rineke Verbrugge — Σ-completeness and Bounded Arithmetic
ML-89-06 Michiel van Lambalgen — The Axiomatization of Randomness
ML-89-07 Dirk Roorda — Elementary Inductive Definitions in HA: from Strictly Positive towards Monotone
ML-89-08 Dirk Roorda — Investigations into Classical Linear Logic

Computation and Complexity Theory:

CT-89-01 Michiel H.M. Smid — Dynamic Deferred Data Structures
CT-89-02 Peter van Emde Boas — Machine Models and Simulations
CT-89-03 Ming Li, Herman Neuféglise, Leen Torenvliet, Peter van Emde Boas — On Space efficient Simulations
CT-89-04 Harry Buhrman, Leen Torenvliet — A Comparison of Reductions on Nondeterministic Space
CT-89-05 Pieter H. Hartel, Michiel H.M. Smid Leen Torenvliet, Willem G. Vree — A Parallel Functional Implementation of Range Queries
CT-89-06 H.W. Lenstra, Jr. — Finding Isomorphisms between Finite Fields
CT-89-07 Ming Li, Paul M.B. Vitanyi — A Theory of Learning Simple Concepts under Simple Distributions and Average Case Complexity for the Universal Distribution (Prel. Version)
CT-89-08 Harry Buhrman, Steven Homer Leen Torenvliet — Honest Reductions, Completeness and Nondeterminstic Complexity Classes
CT-89-09 Harry Buhrman, Edith Spaan, Leen Torenvliet — On Adaptive Resource Bounded Computations

Other Prepublications:

X-89-01 Marianne Kalsbeek — An Orey Sentence for Predicative Arithmetic
X-89-02 G. Wagemakers — New Foundations: a Survey of Quine's Set Theory
X-89-03 A.S. Troelstra — Index of the Heyting Nachlass
X-89-04 Jeroen Groenendijk, Martin Stokhof — Dynamic Montague Grammar, a first sketch
X-89-05 Maarten de Rijke — The Modal Theory of Inequality