Bounded Reductions

Harry Buhrman   Edith Spaan
Leen Torenvliet
Departments of Mathematic and Computer Science
University of Amsterdam
Plantage Muidergracht 24 1018 TV Amsterdam
The Netherlands

Abstract

We study properties of resource– and otherwise bounded reductions and corresponding completeness notions on nondeterministic time classes which contain exponential time. As it turns out, most of these reductions can be separated in the sense that their corresponding completeness notions are different. There is one notable exception. On nondeterministic exponential time, 1-truth table and many-one completeness is the same notion.

1 Introduction

Efficient reducibilities and completeness are two of the central concepts of complexity theory. Since the first use of polynomial time bounded Turing reductions by Cook [4] and the introduction of polynomial time bounded many-one reductions by Karp[6], considerable effort has been put in the investigation of properties and the relative strengths of different reductions and corresponding completeness notions. In 1975, Ladner, Lynch and Selman [8] gave an extensive survey of different types of reductions and differences between these reductions on $E\,(=\cup_{c\in I\!N}\mathrm{DTIME}\,(2^{cn}))$. However, they did not present any conclusions concerning any differences in complete sets for these various reductions. In particular, they left open the question of whether these different reductions yield different complete sets. In 1987, Watanabe [10] building upon earlier work of L. Berman [1], proved almost all possible differences between the polynomial-time completeness notions on E and larger deterministic time classes.

The question of differentiating between complete sets for nondeterministic time classes with respect to various bounded reductions was considered by Buhrman, Homer and Torenvliet in [2]. Their paper however, concentrates on differentiating between the notions of completeness defined by many-one, bounded truth-table and standard (unbounded) Turing reductions, but did not consider the case of *bounded* Turing reductions, in both the polynomial time and logarithmic space case on nondeterministic time and space classes.

In this paper, we concentrate on the remaining open problems between notions of bounded reducibilities, and the corresponding completeness notions on *E*, *NE*, *EXP* and *NEXP* (and solve all of these). As all of the considered reductions are bounded by a *constant* number of queries, the proofs are independent of the specific model for truth-table reducibilities, which may present a problem in the case of unbounded-query truth table reducibilities (cf. [3]).

- In section 3, we prove that k-conjunctive and k-disjunctive truth-table completeness are incomparable.

- In section 4, we show that many-one completeness is the same as 1-truth table completeness.

- In section 5, we give a precise relation between k-Turing and ℓ-truth-table completeness: for $k > 1$, k-Turing completeness strictly contains k-truth-table completeness, and for $k < \ell < 2^k - 1$, k-Turing completeness and ℓ-truth-table completeness are incomparable.

2 Preliminaries

2.1 Machines and languages

Let $\Sigma = \{0, 1\}$. Strings 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 $A, B, C, S, \ldots$. The complement of a language A in Σ^*, $\Sigma^* - A$, will be denoted by $\overline{A}$. For any set S, the cardinality of S is denoted by $\|S\|$. We fix a pairing function $\lambda xy.{<}x, y{>}$ computable in polynomial time from $\Sigma^* \times \Sigma^*$ to Σ^*. Without loss of generality, we assume for all y, y' and x with $|y|, |y'| \leq |x|$ that $|{<}y, x{>}| = |{<}y', x{>}|$. 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 goes to the state YES or goes to 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 two flavors: adaptive and non-adaptive. For a non-adaptive machine, queries may not be interdependent, whereas an adaptive machine may compute the next query depending on the answer to previous queries.

Whenever it is obvious that a universal recognizing or transducing machine exists for a class of languages (i.e. the class is recursively presentable), we will assume an enumeration of the acceptors and/or transducers and denote this enumeration by $M_1, M_2, \ldots$. For a Turing machine M, let $L(M)$ denote the set of strings accepted by M. For an oracle machine M, let $L(M, A)$ denote the set of strings accepted by M with oracle A.

2.2 Time classes

Let DTIME$(f(n))$ be the class of sets such that $A \in$ DTIME$(f(n))$ iff there exists a Turing machine M for which the running time is bounded by $f(n)$ as $n \to \infty$ (n is the length of the input) and $A = L(M)$. Let NTIME$(f(n))$ be the corresponding nondeterministic class. We define the following classes:

$$\begin{aligned}
\mathit{NEXP} &= \bigcup_{i=1}^{\infty} \mathrm{NTIME}\left(2^{n^i}\right) \\
\mathit{EXP} &= \bigcup_{i=1}^{\infty} \mathrm{DTIME}\left(2^{n^i}\right) \\
\mathit{NE} &= \bigcup_{c=1}^{\infty} \mathrm{NTIME}\left(2^{cn}\right) \\
E &= \bigcup_{c=1}^{\infty} \mathrm{DTIME}\left(2^{cn}\right)
\end{aligned}$$

2.3 Truth tables

The ordered pair $<<a_1, \ldots, a_k>, \alpha>$ $(k > 0)$ is called a *truth-table condition of norm* k if $<a_1, \ldots, a_k>$ is a k-tuple of strings, and α is a k-ary Boolean function [8]. 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^*. If, for all x, $f(x)$ has norm less than or equal to k, then f is called a k-truth-table (k-tt) function. We say that a tt-function f is a *disjunctive (*conjunctive) truth-table (d-tt (c-tt)) function if f is a truth-table condition and its Boolean function is always a disjunction (conjunction) of its arguments.

2.4 Reductions, reducibilities and completeness

Let $A_1, A_2 \subseteq \Sigma^*$. We say that:

1. A_1 is polynomial-time many-one reducible to A_2 ($\leq_m^p$-reducible) iff there exists a function f computable within polynomial-time such that $x \in A_1$ iff $f(x) \in A_2$.

2. A_1 is polynomial-time k-truth-table reducible to A_2 ($\leq_{k-tt}^p$-reducible) iff there exists a polynomial-time bounded k-tt-function f such that $\alpha(\chi_{A_2}(a_1), \ldots, \chi_{A_2}(a_k))$ evaluates to **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.

3. A_1 is polynomial-time Turing reducible ($\leq_T^p$-reducible) to A_2 iff there exists a polynomial-time bounded deterministic oracle machine such that $A_1 = L(M, A_2)$.

4. A_1 is polynomial-time disjunctive (conjunctive) reducible ($\leq_d^p$ ($\leq_c^p$)-reducible) to A_2, iff $A_1 \leq_{tt}^p A_2$ by some d-tt(c-tt)-function. For $k \geq 0$, A_1 is k-disjunctive(conjunctive) reducible ($\leq_{k-d}^p$ ($\leq_{k-c}^p$)) to A_2, if $A_1 \leq_{tt}^p A_2$ by some d-tt(c-tt)-function of norm k.

Let $\leq_r^p$ be any of the above reductions

1. A set A is $\leq_r^p$-hard for some complexity class C iff for all $B \in C$, B is $\leq_r^p$-reducible to A.

2. A set A is $\leq_r^p$-complete for some complexity class C iff A is $\leq_r^p$-hard for C and $A \in C$.

For *NEXP* we use a standard many-one complete set K. $K = \{<i, x, l>|$ machine M_i has an accepting computation on input x within steps$\}$. Note that this set can be recognized in 2^n steps and is therefore also complete for *NE*. For *EXP* and E we use $K = \{<i, x, l>|$ machine M_i is deterministic and accepts x within l steps$\}$.

3 Disjunctive and Conjunctive Truth-table Reductions

Theorem 1 *there exists a set $A \in NEXP$ such that A is $\leq^p_{2-d}$-complete but not $\leq^p_{2-c}$-complete.*

Proof: Let K be the standard $\leq^p_m$-complete set for NE as defined above. To achieve the separation, we construct a set $W \in E$ and a set $A \in NEXP$ such that $W \not\leq^p_{2-c} A$, but $K \leq^p_{2-d} A$. We assume an enumeration of polynomial time 2-conjunctive truth-table reductions $M_1, M_2, \ldots$ where M_i runs in time n^i. We need a set of elements on which to diagonalize. To get such a set, we define a sequence of integers $\{b(n)\}_n$:

$$b(n) = \begin{cases} 1 & \text{if } n \leq 1 \\ 2^{b(n-1)^{n-1}} + 1 & \text{otherwise} \end{cases}$$

We construct A and W in stages. $A \subseteq \{0,1\} \times \Sigma^*$ and $W \subseteq \{0\}^*$. At stage n, we define A_n and decide whether $0^{b(n)} \in W$ or $0^{b(n)} \in \overline{W}$. We let $A_{<n} = \bigcup_{i=0}^{n-1} A_i$ and $A = \bigcup_{n=0}^{\infty} A_n$

Stage 0: $A_0 = W = \emptyset$.

stage n:

Let $A'_n = \{<i, z>|\ z \in K \text{ and } b(n-1)^{n-1} < |<i, z>| \leq b(n)^n \text{ and } i \in \{0,1\}\}$
Simulate M_n on input $0^{b(n)}$. M_n will query at most two strings x and y. Without loss of generality, let x be the largest (in lexicographic order) of the two. M_n accepts iff x and y are both in the oracle set.
There are two cases:

1. $|x| \leq b(n-1)^{n-1}$
2. $b(n-1)^{n-1} < |x| \leq b(n)^n$

In case 1, compute the answers relative to $A_{<n}$ of both x and y and put $0^{b(n)} \in W$ iff M_n rejects. Let $A_n = A'_n$
In case 2, put $0^{b(n)} \in W$ and let $A_n = A'_n - \{x\}$. This ensures that M_n^A rejects on input $0^{b(n)}$.

end of stage n

The remainder of the proof consists of four items. We show that $A \in NEXP$, that $W \in E$, that A is not $\leq^p_{2-c}$-complete, and finally that A *is* $\leq^p_{2-d}$-complete.

- We show first that $A \in NEXP$. To decide $<i,z> \in A$ $(i = 0,1)$, compute n such that $b(n)^n \geq |<i,z>| > b(n-1)^{n-1}$, which can be done in linear time. Simulate machine M_n on input $0^{b(n)}$ and compute x and y. If $<i,z> = x$ reject, else accept iff $z \in K$. All this can be done in nondeterministic exponential time, since simulation of machine M_n on input $0^{b(n)}$ takes time $b(n)^n \leq 2^{n(b(n-1)^{n-1}+1)} \leq 2^{|<i,z>|^2}$, as $b(n) \geq 1$, and $b(n) \geq 2n$, whenever $n \geq 3$.

- Next we show that $W \in E$. On input $0^{b(n)}$, simulate M_n on input $0^{b(n)}$ and compute the queries x and y. Assume again, that x is the larger query in lexicographical order. If $|x| > b(n-1)^{n-1}$ we accept, else we will decide membership of x and y in A, and accept iff M_n rejects. To compute if $x \in A$, determine $n' < n$ such that $b(n'-1)^{n'-1} < |x| \leq b(n')^{n'}$. $x \in A$ iff x is not the largest query asked by $M_{n'}$ and $x \in K$. This takes deterministic time $2^{2^{|x|}} < 2^{b(n)}$. Membership of y in A can be decided by the same algorithm.

- Now assume for a contradiction, that A is $\leq^p_{2-c}$-complete. Note that $0^{b(n)} \in W$ iff M_n rejects. Then there must be a 2-conjunctive truth-table reduction from W to A. Let M_j be the machine witnessing this reduction. Then, by construction, $0^{b(j)}$ is in W iff M_j on input $0^{b(j)}$ rejects. This contradicts the assumption that M_j reduces W to A. This proves that A is not $\leq^p_{2-c}$-complete.

- Finally, we give the $\leq^p_{2-d}$-reduction from K to A. Since in every step, only one of the pairs $<1,x>$ or $<0,x>$ can be deleted, $x \in K$ iff $<0,x> \in A$ or $<1,x> \in A$. Therefore, the following reduction reduces K to A:
$$g(x) = \{<0,x> \vee <1,x>\}$$

$\square$

The same proof technique yields the following theorem.

Theorem 2 *There exists a set $A \in NEXP$ such that A is $\leq^p_{2-c}$-complete, but not $\leq^p_{2-d}$-complete.*

Proof: The proof is almost the same as the previous one. It differs only in case 2 of the diagonalization. Here we put $0^{b(n)}$ not in W and *add* x to $A_n^{'}$. In this way we ensure that $0^{b(n)} \notin W$ iff M_n^A accepts. Note that $x \in K$ iff $<0,x> \in A$ and $<1,x> \in A$. The $\leq_{2-c}^p$-reduction from K to A becomes:

$$g(x) = \{<0,x> \wedge <1,x>\}$$

$\square$

It is easy to see, that the proofs generalize to $\leq_{k-d}^p$-complete sets v.s. $\leq_{k-c}^p$-complete sets (for $k \geq 2$). The theorems solve an open problem from Watanabe [10].

Corollary 3 *For all $k \geq 2$, there exists a set A that is $\leq_{k-tt}^p$-complete for NEXP, but not $\leq_{k-d}^p$ ($\leq_{k-c}^p$)-complete for NEXP.*

This corollary can be strengthened. We are now able to construct a set that is $\leq_{k-tt}^p$-complete, but neither $\leq_{k-d}^p$-complete nor $\leq_{k-c}^p$-complete.

Corollary 4 *For all $k \geq 2$, there exists a set A that is $\leq_{k-tt}^p$-complete for NEXP, but neither $\leq_{k-d}^p$-complete, nor $\leq_{k-c}^p$-complete for NEXP*

Proof: To do this we use the constructions of Theorem 1 at the even stages and the constructions of Theorem 2 at the odd stages $\square$

Corollary 5 *For all $k \geq 2$, there exists a set A that is $\leq_{k-tt}^p$-complete for NEXP, but neither $\leq_{d}^p$-complete, nor $\leq_{c}^p$-complete.*

By standard padding techniques the results in this section go through for *NE*, and since we do not make use of any special properties of nondeterminism, they also hold for *E* and *EXP*. By complementation the results also hold for Co-*NE* and Co-*NEXP*.

4 1-Truth-Table versus Many-One

Another question concerning reductions on *E* and *NE* is whether the notions $\leq_{1-tt}^p$ and $\leq_m^p$ differ for complete sets. From recursion theory it is known (and easy to prove), that these two reductions are the same with respect to *RE* sets if the time bounds on the reductions are omitted. Recently Homer et. al. [5] showed, that these two notions are also the same for *E*-complete sets. They also showed the notions to be identical in the case of RE-complete sets under polynomial time bounded reductions. They left open however the question for NE. We solve this question here. The idea is to first prove that sets in $NE \cap co\text{-}NE$ that are $\leq^p_{1-tt}$-reducible to a complete set are also $\leq^p_m$-reducible to this set. This can be done using a similar technique as in [5]. Once this is done, we are able to reduce the general case (if this is necessary) to this special case.

Lemma 6 *Let T be a $\leq^p_{1-tt}$-complete set for NE. For every set $A \in NE \cap co\text{-}NE$, $A \leq^p_m T$.*

Proof: We assume an enumeration of polynomial time 1-truth-table reductions $M_1, M_2, \ldots$. Without loss of generality, we may assume that M_i runs in time n^i. Let A be any set in $NE \cap co\text{-}NE$. We will construct a set $D \in NE$ with the property that $A \leq^p_{1-tt} D \leq^p_{1-tt} T$ and the $\leq^p_m$-reduction from A to T can be computed from the $\leq^p_{1-tt}$-reduction from D to T. We simulate M_i on input $<i,x>$, and let z be the string queried by M_i. Now there are four possible cases that can occur:

1. M_i accepts iff z is in the oracle set.
2. M_i accepts iff z is *not* in the oracle set.
3. M_i accepts. (M_i is not a $\leq^p_{1-tt}$-reduction on this input)
4. M_i rejects. (M_i is not a $\leq^p_{1-tt}$-reduction on this input)

In case 1 we put the pair $<i,x>$ in D iff $x \in A$
In case 2 we put $<i,x>$ in D iff $x \notin A$
In case 3 we put $<i,x>$ not in D
In case 4 we put $<i,x>$ in D
We first note, that D is in NE. To decide whether $<i,x>$ is in D or not, simulate machine M_i on input $<i,x>$ and find out in which case M_i ends up. The only problem is case 2, but since A is in $NE \cap co\text{-}NE$, we can compute whether x is in the complement of A. Having shown that D is in NE, we infer that D is 1-truth-table reducible to T. Let machine M_h witness this reduction and let z be the string queried by machine M_h on input $<h,x>$. Now we can construct the many one reduction f from A to T:

$$f(x) = z$$

Since machine M_h runs in polynomial time this reduction also runs in polynomial time. Machine M_h can not end up in case 3 or 4 on *any* input $<h,x>$, since this would contradict the fact the M_h is a 1-truth-table reduction from D to T. The following two cases remain:

- Machine M_h is in case 1. Then $x \in A$ iff $<h, x> \in D$ iff M_h accepts iff $z \in T$.
- Machine M_h is in case 2. Then $x \in A$ iff $<h, x> \notin D$ iff M_h rejects iff $z \in T$.

So in both cases $x \in A$ iff $z \in T$. $\square$

Now for all sets in *NE* if a set is 1-truth-table reducible to a complete set T via say machine M_j there are strings that are accepted if the query is in T. Those strings are already many-one reducible to T. The other strings (i.e. the strings that get accepted by a query in the complement of T) form a set that is in $NE \cap co\text{-}NE$ and by Lemma 6 they are many-one reducible to T via some other reduction. We state:

Theorem 7 *Every $\leq^p_{1-tt}$-complete set for NE is also $\leq^p_m$-complete.*

Proof: Let A be a set in *NE*, T a 1-truth-table complete set in *NE* and let M_j witness the reduction from A to T. On any input M_j can end up in one of the following four situations:

1. M_j queries z and accepts iff $z \in T$
2. M_j queries z and accepts iff $z \notin T$
3. M_j accepts
4. M_j rejects

We now split set A in two subsets A_1 and A_2.

$$\begin{aligned} A_1 &= \{x \mid x \in A \text{ and machine } M_j \text{ is not in case } 2\} \\ A_2 &= \{x \mid x \in A \text{ and machine } M_j \text{ is in case } 2\} \end{aligned}$$

CLAIM 8 *A_2 is in $NE \cap co\text{-}NE$.*

Proof: We need to show that there is a *NE* predicate for A_2 and for the complement of A_2.

$$x \in A_2 \text{ iff machine } M_j \text{ in case 2 and } x \in A$$
$$x \notin A_2 \text{ iff machine } M_j \text{ not in case 2 or } z \in T$$

It is clear that both predicates are NE. □

Now we can construct the many-one reduction from A to T: On input x simulate machine M_j on input x. If M_j is in case 1, then output z. If M_j in case 2, then x is in A iff x is in A_2. Since A_2 is in $NE \cap co\text{-}NE$ there is, by Lemma 6, a many-one reduction from A_2 to T say g. Now output $g(x)$. If M_j is in case 3, output a fixed element $t_0 \in T$, and if M_j is in case 4, output a fixed element $t_1 \notin T$. The entire construction can be carried out in polynomial time. □

The construction can be generalized to a recursion theoretic setting. We relax the time bounds and end up with recursive reductions. We now have the following equivalent reductions $\leq_m^{rec}$ for a many-one reduction and $\leq_{1-tt}^{rec}$ for a 1-truth-table reduction in exactly the same way as the above theorem was proven we can prove the following:

Corollary 9 *let Σ_k be the k^{th} level of the arithmetic hierarchy as defined in [9]. For all k, if A is $\leq_{1-tt}^{rec}$-complete for Σ_k, then A is $\leq_m^{rec}$-complete for Σ_k.*

It would be interesting to prove the same result for the class NP. The problem is that the technique used in lemma 6 is not applicable for sets in NP. Under the strong assumption that $P = NP \cap co\text{-}NP$ however, we can prove it.

Corollary 10 *If $P = NP \cap co\text{-}NP$ then every $\leq_{1-tt}^{p}$-complete set for NP is $\leq_m^p$-complete*

5 Bounded Turing versus bounded Truth-Table

We now turn our attention to bounded Turing reductions. Informally, these are Turing reductions where for any input x, the number of queries asked is bounded by a constant k. Note that by definition, every k-truth table reduction is a k-Turing reduction. It is well known, that every k-Turing reduction can be simulated by $(2^k - 1)$-truth-table reduction. A natural question one can ask is: "What is the relation between k-Turing reductions versus ℓ-truth-table reductions?" In the previous section, it was proven that for nondeterministic-exponential-time complete sets: many-one = 1-truth-table = 1-Turing. In this section we prove that k-Turing reductions are more powerful than k-truth-table reductions for $k > 1$, and that for $k < \ell < 2^k - 1$, k-Turing and ℓ-truth table reductions are incomparable. These results hold also for the corresponding completeness notions on *NEXP*.

Definition 11 *Let $Q(M, x, A)$ be the set of strings, queried in the computation of polynomial time oracle machine M with oracle A on input x. We say that $B \leq^p_{k-T} A$ if there exists a polynomial time oracle machine M such that $B = L(M, A)$ and for all $x, \|Q(M, x, A)\| \leq k$.*

Theorem 12 *For every k, there exists a set D in NEXP that is $\leq^p_{k-T}$-complete, but not $\leq^p_{(2^k-2)-tt}$-complete.*

As an example of the techniques used, we first prove the degenerate case $k = 2$, i.e. we will construct a set $D \in NEXP$, such that D is $\leq^p_{2-T}$-complete but not $\leq^p_{2-tt}$-complete.

Proof: Let $M_1, M_2, \dots,$ be an enumeration of the 2-truth-table reductions, where M_i runs in time n^i. Let K be the standard $\leq^p_m$-complete set for *NE* and let $\{b(n)\}_n$ the sequence defined in the proof of Theorem 1. We will construct sets $D \in NEXP$ and $W \in E$, such that $W \not\leq^p_{2-tt} D$, and $K \leq^p_{2-T} D$. W and D will be constructed in stages. At each stage n, we will define a set D_n and we let $D = \bigcup_{n=0}^{\infty} D_n$.

To ensure that $K \leq^p_{2-T} D$, we have to exploit the fact that a 2-Turing reduction can ask 3 queries in its *entire* query tree, while a 2-truth-table reduction can ask at most 2 queries in its entire query tree. We will ensure that $D \subseteq \{0, 1, 2\} \times K$, and use the following 2-Turing reduction M_T to reduce K to D:

On input x, first query $<0, x>$. If the answer is YES, query $<1, x>$, and accept iff the answer is YES. If the answer to query $<0, x>$ is NO, query $<2, x>$ and accept iff the answer is YES.

For every 2-truth-table reduction, and for every x, at least one of the strings $<0, x>, <1, x>, <2, x>$ is not queried on input $0^{b(n)}$, where n is such that $<0, x> \leq b(n)^n$. This provides enough freedom to diagonalize against the 2-truth-table reductions, while still keeping $K \leq^p_{2-T} D$ by M_T.

At stage $0 : D_0 = W = \emptyset$

stage n:

Let $D_n = \{<i,x>|\ x \in K \text{ and } b(n-1)^{n-1} < |<i,x>| \leq b(n)^n \text{ and } 0 \leq i \leq 2\}$.

Simulate M_n on input $0^{b(n)}$. If M_n queries strings of length $\leq b(n-1)^{(n-1)}$, compute the answers to those strings. I.e., compute the membership of those strings in $\bigcup_{i=0}^{n-1} D_i$.

Let Q be the set of (at most two) queries $\in \{0,1,2\} \times \Sigma^*$ with length $> b(n-1)^{(n-1)}$. Let $i_0 \in \{0,1,2\}$ be a number such that Q contains no string of the form $<i_0,x>$. Now we take the following action, depending on the value of i_0:

$i_0 = 0$: For every y occurring as second member in a pair of Q do
$D_n := (D_n \setminus \{<2,y>\}) \cup \{<1,y>\}$

$i_0 = 1$: For every y occurring as second member in a pair of Q do
$D_n := (D_n \setminus \{<2,y>\}) \cup \{<0,y>\}$

$i_0 = 2$: For every y occurring as second member in a pair of Q do
$D_n := (D_n \setminus \{<0,y>\}) \cup \{<1,y>\}$

Now we are able to compute whether M_n accepts or rejects the input $0^{b(n)}$, when given the oracle $\bigcup_{i=0}^{n} D_i$. The fact that no strings of length $\leq b(n)^n$ (i.e. strings that can be queried by M_n on input $0^{b(n)}$) are defined inside or outside D at subsequent stages ensures, that the computation on this input is the same as with oracle D

Put $0^{b(n)}$ in W iff M_n rejects on input $0^{b(n)}$.

end of stage n

We can use an argument similar to the one used in the proof of Theorem 1, to prove that $D \in NEXP$, $W \in E$ and W is not $\leq^p_{2-tt} D$. It remains to prove, that D is $\leq^p_{2-T}$-hard for $NEXP$. Our 2-Turing reduction M_T accepts x iff either $(<0,x> \in D \wedge <1,x> \in D)$ or $(<2,x> \in D \wedge <0,x> \notin D)$.

We have the following possibilities for $D \cap \{<0,x>,<1,x>,<2,x>\}$

$x \in K$: $\{<0,x>,<1,x>,<2,x>\}$ or $\{<0,x>,<1,x>\}$ or $\{<1,x>,<2,x>\}$.

$x \notin K$: $\emptyset$ or $\{<0,x>\}$ or $\{<1,x>\}$.

Thus, M_T accepts x iff $x \in K$ as required. $\square$

For this proof, it was essential that a 2-Turing reduction can ask more queries in its entire query tree than can a 2-truth-table reduction. Since a

k-Turing reduction can ask $2^k - 1$ queries in its entire query tree, while a $2^k - 2$ truth-table reduction can ask at most $2^k - 2$ queries in its entire oracle tree, we can use a generalization of the previous construction, to obtain a set D that is $\leq^p_{k-T}$-complete, but not $\leq^p_{(2^k-2)-tt}$-complete, thus proving Theorem 12.

Proof: Let $M_1, M_2, \ldots,$ be an enumeration of the $2^k - 2$-truth-table reductions, where M_i runs in time n^i. Let K be a standard $\leq^p_m$-complete set for *NE* and let $\{b(n)\}_n$ the sequence defined in the proof of Theorem 1. We construct set D and W in stages; $D = \bigcup_{n=0}^{\infty} D_n$. We will ensure that $D \subseteq \{0, \ldots, 2^k - 2\} \times \Sigma^*$, and use the following k-Turing reduction M_T to reduce K to D.

On input x, first query $<0, x>$. For the j-th query $<i, x>$, where $j < k$ do the following: if the answer is YES, query $<2i+1, x>$, else query $<2i+2, x>$. Accept iff the k-th query asked gets answer YES.

At stage $0: D_0 = W = \emptyset$

stage n:

Let $D_n = \{<i, x>| \; x \in K$ and $b(n-1)^{n-1} < |<i, x>| \leq b(n)^n$ and $0 \leq i \leq 2^k - 2\}$.

Simulate M_n on input $0^{b(n)}$. If M_n queries strings of length $\leq b(n-1)^{(n-1)}$, compute the answers to those strings.

Let Q be the set of queries $\in \{0, \ldots, 2^k - 2\} \times \Sigma^*$ with length $> b(n-1)^{(n-1)}$. Let $i_0 \in \{0, \ldots, 2^k - 2\}$ be such that Q contains no string of the form $<i_0, x>$.

Consider the following tree of depth k, where the nodes are labeled $0, \ldots, 2^k - 2$: The root has label 0, and for each node at depth $< k$ with label i, the left child has label $2i+1$, and the right child label $2i+2$. We will identify roots with their labels, and we will visualize this tree as a query tree, such that on input x the node labeled i corresponds to query $<i, x>$. If our Turing reduction receives a YES answer to a query represented at some node, then the next query computed is represented by its left child and if it receives a NO answer, then the next query computed is represented by its right child.

For every y that occurs as second member in a pair of Q and and for every $i \in \{0, \ldots 2^k - 2\}, i \neq i_0$, we take the following action:

1. if i occurs on the path from the root to i_0 then

if i_0 is in the left subtree of i then $D_n := D_n \cup \{<i,y>\}$
if i_0 is in the right subtree of i then $D_n := D_n \setminus \{<i,y>\}$

2. if i occurs to the left of the path from the root to i_0 then $D_n := D_n \cup \{<i,y>\}$

3. if i occurs to the right of the path from the root to i_0 then $D_n := D_n \setminus \{<i,y>\}$

4. if i is in the left subtree of i_0 then $D_n := D_n \cup \{<i,y>\}$

5. if i is in the right subtree of i_0 then $D_n := D_n \setminus \{<i,y>\}$

Now we are able to compute whether M_n accepts or rejects.
Put $0^{b(n)}$ in W iff M_n rejects on input $0^{b(n)}$.

end of stage n

We can use a similar argument as in the proof of theorem 1, to prove that $D \in \mathit{NEXP}$, $W \in E$ and $W \not\leq^p_{(2^k-2)-tt} D$. It remains to prove that D is $\leq^p_{k-T}$-hard for *NEXP*.

Recall that our k-Turing reduction M_T works as follows: On input x, first query $<0,x>$. For each query $<i,x>$ at depth $< k$ do the following: if the answer is YES, query $<2i+1,x>$, else query $<2i+2,x>$. Accept iff the last query asked gets answer YES. View this reduction as a tree of depth k, where the nodes are labeled by the queries, and a YES (resp. NO) answer to a query corresponds to taking the left (resp. right) branch.

If $x \in K$, then M_T, on input x, takes either the leftmost path in its oracle tree (if $i_0 = 0$), or the leftmost path through the node labeled i_0 which corresponds to query $<i_0,x>$. In either case we accept.

If $x \notin K$, then M_T, on input x, takes either the rightmost path in its oracle tree, or the rightmost path through $<i_0,x>$. In either case we reject.

Thus, M_T is a reduction from K to D. □

Now we will construct a set D in *NEXP* that is $\leq^p_{(k+1)-tt}$-complete, but not $\leq^p_{k-T}$-complete. A $\leq^p_{k-T}$-reduction can be represented as a binary tree of depth k, where every node in the tree represents a query. The reduction starts with the query represented by the root and proceeds after obtaining an anwer to each query as follows. If the answer to the query is YES, we proceed to the left branch otherwise to the right branch. The leaves of the tree are labeled with the quality accept or reject. The idea of the oracle construction is to force the $\leq^p_{k-T}$-reduction into one branch by leaving out all the queries (if possible) of that branch. Since there are only k queries on one branch, there remains the freedom to code an extra pair of K into D that can be queried by a $\leq^p_{(k+1)-tt}$-reduction.

Theorem 13 *There exists a set D in NEXP that is $\leq^p_{(k+1)-tt}$-complete, but not $\leq^p_{k-T}$-complete.*

Proof: We only give the construction for $k = 4$. Let $M_1, M_2, \ldots$ be an enumeration of $\leq^p_{4-T}$ reductions. Let K be the standard $\leq^p_m$-complete set for *NE* and $\{b(n)\}_n$ the sequence defined in the proof of theorem 1. Again we use a stage construction.

stage n:

$D'_n := \{<i,x>|\ x \in K$ and $b(n-1)^{(n-1)} < |<i,x>| \leq b(n)^n$ and $0 \leq i \leq 4\}$

Simulate M_n on input $0^{b(n)}$, and compute the answers to the queries of length $< b(n-1)^{n-1}$. Now evaluate the branch where all the other queries receive the answer NO. Let Q' be the set of the queries that are large (of length $\geq b(n-1)^{(n-1)}$).

Put $0^{b(n)}$ in W iff M_n rejects

$D_n := D'_n \backslash Q'$

end of stage n

Note that for every $x : x \in K$ iff $<i,x> \in D$ for some i. The 5-truth-table reduction from K to D becomes:

$$g(x) = \{<0,x> \vee \ldots <4,x>\}$$

□

Corollary 14 *If $k < \ell < 2^k - 1$, then $\leq^p_{k-T}$ and $\leq^p_{\ell-tt}$ are incomparable with respect to complete sets for NEXP.*

As before the results go through for *NE*,*E* and *EXP*.

6 Conclusions

In the previous sections, we proved that almost all reductions on *NE*, *E*, *EXP* and *NEXP* are incomparable except those where inclusion is trivial.

As a consequence, the extended Berman Hartmanis conjecture for those reductions fails. It follows, that for example the sets in the degree of 2-truth-table complete sets are not p-isomorphic. An interesting step would be to disprove the extended conjecture for the degree of many-one complete sets. Perhaps the techniques discussed here could lead towards results in that direction.

The proof of the non-separation of many-one and 1-truth-table reductions fails for NP. The problem is, that it is not known whether the universal polynomial time function is computable in NP. For all well behaved classes that contain the universal polynomial time function, this non-separation result is true.

One area of great interest would be to separate the various polynomial time reductions on classes between P and $PSPACE$, and in particular to do this for NP.

All the previous obtained results go through with respect to logspace reductions for nondeterministic and deterministic space classes that contain that universal logspace function. It would be interesting to prove a similar result for $NLOGSPACE$.

Acknowledgements We would like to thank Steven Homer and Peter van Emde Boas for fruitful discussions.

References

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

[2] Buhrman H., S. Homer & L. Torenvliet. *Completeness for nondeterministic complexity classes* Mathematical Systems Theory **24** (1991) 179–200.

[3] Buhrman H., E. Spaan & L. Torenvliet. *The relative power of logspace and polynomial time reductions*. To appear.

[4] Cook, S. A. *The complexity of theorem-proving procedures*. Proc. 3d ACM Symp. on Theory of Computing, Assoc. for Computing Machinery, New York (1971) pp151–158.

[5] Homer S, S. Kurtz & J. Royer *A note on many-one and 1-truth table complete languages* To appear in Theoretical Computer Science.

[6] Karp, R.M. *Reducibility among combinatorial problems*. Complexity of Computer Computations, R.E. Miller & J.W. Thatcher eds. Plenum N.Y. pp85–103.

[7] Ladner R.E. & N.A. Lynch. *Relativization of Questions About Log Space Computability*. Mathematical Systems Theory 10 (1976) pp19–32.

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

[9] Soare, R.I. *Recursively Enumerable Sets and Degrees*. Perspectives in Mathematical Logic. Springer-Verlag (1987) pp60–61.

[10] Watanabe, O. *A comparison of polynomial time completeness notions*. Theoretical Computer Science 54 (1987) pp249–265.