Comparing Reductions to NP-Complete Sets

John M. Hitchcock [*] A. Pavan [†]

Abstract

Under the assumption that NP does not have p-measure 0, we investigate reductions to NP-complete sets and prove the following:

1. Adaptive reductions are more powerful than nonadaptive reductions: there is a problem that is Turing-complete for NP but not truth-table-complete.
2. Strong nondeterministic reductions are more powerful than deterministic reductions: there is a problem that is SNP-complete for NP but not Turing-complete.
3. Every problem that is many-one complete for NP is complete under length-increasing reductions that are computed by polynomial-size circuits.

The first item solves one of Lutz and Mayordomo's "Twelve Problems in Resource-Bounded Measure" (1999). We also show that every many-one complete problem for NE is complete under one-to-one, length-increasing reductions that are computed by polynomial-size circuits.

1 Introduction

A language $L \in$ NP is NP-complete if every language in NP is *reducible* to L. There are several possible interpretations of the word "reducible." Polynomial-time many-one reducible is the most typical meaning, but there are many other reducibilities, each providing a potentially different NP-completeness notion. Are there languages that are NP-complete using one type of reduction but not complete under another type of reduction? Are there two apparently different notions of reductions for which the corresponding completeness notions coincide? We study these questions for several types of reductions.

1.1 Adaptive versus Nonadaptive Reductions

A many-one reduction ($\leq_{\mathrm{m}}^{\mathrm{p}}$) from A to B converts a question about membership in A to an equivalent question about membership in B. Formally, there is a function f such that $x \in A$ if and only if $f(x) \in B$. A variation on this theme is to allow the use of B as an oracle to solve A. Here there is an algorithm M that takes as input an instance x and may ask multiple queries about membership in B before outputting its decision for membership of x in A. There are two basic forms of this type of reduction: adaptive and nonadaptive. In an adaptive reduction (also called a Turing reduction, $\leq_{\mathrm{T}}^{\mathrm{p}}$) M receives the answer for each query before asking its next query – subsequent queries may depend on the answers to previous queries. In a nonadaptive reduction (also called a truth-table reduction, $\leq_{\mathrm{tt}}^{\mathrm{p}}$) M asks all of its queries before receiving any answers.

[*]Department of Computer Science, University of Wyoming. This research was supported in part by NSF grant 0515313.

[†]Department of Computer Science, Iowa State University. This research was supported in part by NSF grant 0430807.

Lutz and Mayordomo [24] showed that if NP does not have p-measure zero (written $\mu_{\mathrm{p}}(\mathrm{NP}) \neq 0$), then adaptive completeness for NP is different from many-one completeness. In fact, they showed this hypothesis yields a problem that is complete for NP under adaptive reductions that make only two queries, but is not complete under many-one reductions. In the conclusion of their paper, Lutz and Mayordomo conjectured that the measure hypothesis would yield separations of other completeness notions between $\leq_{\mathrm{m}}^{\mathrm{p}}$ and $\leq_{\mathrm{T}}^{\mathrm{p}}$ for NP, similar to what is known unconditionally for E and NE [30, 9].

Since then there have been several results in this direction. Ambos-Spies and Bentzien [5] used a genericity hypothesis on NP, an assumption which is implied by the measure hypothesis, to separate essentially all bounded-query completeness notions for NP. It is also known that some of these separations can be obtained under bi-immunity hypotheses [28, 19], which are even weaker assumptions. For a survey of these results see [26].

However, so far a separation of adaptive completeness from nonadaptive completeness for NP has been elusive. This question has been asked in several survey papers [11, 23, 12, 25], most prominently as one of Lutz and Mayordomo's "Twelve Problems in Resource-Bounded Measure," Problem 9:

> Does $\mu_{\mathrm{p}}(\mathrm{NP}) \neq 0$ imply the existence of a problem that is $\leq_{\mathrm{T}}^{\mathrm{p}}$-complete, but not $\leq_{\mathrm{tt}}^{\mathrm{p}}$-complete, for NP?

The only partial result on this problem was by Pavan and Selman [27] who used a strong hypothesis about UP to separate these two completeness notions. We affirmatively answer the above question. Our proof combines the connection between the measure of NP and the NP-machine hypothesis [17] with results about nonadaptive reductions to P-selective sets [10, 29].

1.2 Nondetermistic versus Deterministic Reductions

Adleman and Manders [1] observed that while most problems can be shown to be NP-complete using polynomial-time reductions, some problems resist this approach. To classify such problems, they proposed what are now called *strong nondeterministic many-one reductions.* (Adleman and Manders called these reductions γ-reductions.) If a language that is NP-complete under strong nondeterministic reductions admits an efficient algorithm, then NP = coNP. Therefore, if we believe NP $\neq$ coNP, strong nondeterministic completeness can also be taken as evidence that the problem in hand is intractable.

Adleman and Manders showed that some number-theoretic problems are NP-complete under strong nondeterministic many-one reductions. Chung and Ravikumar [13] showed that certain questions regarding comparator networks are also NP-complete under these reductions. It is not known whether these problems remain complete if we use polynomial-time reductions.

This situation raises the following question: are there languages that are complete under strong nondeterministic reductions, but not complete under polynomial-time reductions? We show that if $\mu_{\mathrm{p}}(\mathrm{NP}) \neq 0$, then the answer to this question is yes, even if we consider polynomial-time adaptive reductions.

1.3 Length-Increasing Reductions

It has been observed that many NP-completeness results hold under very restrictive reductions. For example, SAT is complete under polynomial-time reductions that are one-to-one and length-increasing. In fact, all known many-one complete problems for NP are complete under this type of reduction [8]. This raises the following question: are there languages that are complete under polynomial-time many-one reductions but not complete under polynomial-time, one-to-one, length-increasing reductions?

Berman [7] showed that every many-one complete set for E is complete under one-to-one, length-increasing reductions. Thus for E, these two completeness notions coincide. A weaker result is known for NE. Ganesan and Homer [15] showed that all NE-complete sets are complete via one-to-one reductions that are exponentially honest.

For NP, until very recently there had not been any progress on this question. Agrawal [3] showed that if one-way permutations exist, then all NP-complete sets are complete via one-to-one, length-increasing, p/poly-reductions. Agrawal's result also holds for the NE-complete sets under the same hypothesis.

In this paper, we show that if $\mu_{\mathrm{p}}(\mathrm{NP}) \neq 0$, then all NP-complete sets are complete via length-increasing, p/poly-reductions. We note that the measure hypothesis on NP is apparently incomparable with Agrawal's hypothesis that one-way permutations exist. Regarding NE-completeness, we show that Agrawal's result can be made unconditional. That is, we unconditionally show that all NE-complete sets are complete via one-to-one, length-increasing, p/poly-reductions.

2 Preliminaries

We assume that the reader is familiar with polynomial-time many-one reductions. A many-one reduction f is *polynomially-honest* (or just *honest*) if there is a polynomial p such that $|x| \leq p(|f(x)|)$ for all x. A language A is *polynomial-time Turing reducible* to B ($A \leq_{\mathrm{T}}^{\mathrm{p}} B$) if there is a polynomial-time oracle Turing M such that $A = L(M^B)$. A language A is *polynomial-time truth-table reducible* to a language B ($A \leq_{\mathrm{tt}}^{\mathrm{p}} B$) if there exist polynomial-time computable functions g and h such that for every x, $g(x)$ is a set of queries $\{q_1, \cdots, q_m\}$ and $x \in A$ if and only if $h(x, B(q_1), \cdots, B(q_m)) = 1$. Given a reducibility $\leq_r^\alpha$, a set S in NP is $\leq_r^\alpha$*-complete* for NP if every set in NP is $\leq_r^\alpha$-reducible to S.

2.1 Resource-Bounded Measure

Lutz [22] introduced resource-bounded measure to study the quantitative structure of complexity classes.

A *martingale* is a function $d : \Sigma^* \to \mathbb{Q}$ with the property that for every $w \in \Sigma^*$, $2d(w) = d(w0) + d(w1)$. A martingale d *succeeds* on a language A if

$$\limsup_{n\to\infty} d(A|n) = \infty,$$

where $A|n$ is the length n prefix of A's characteristic sequence.

Intuitively, the martingale d can be viewed as a strategy that bets on the successive bits of the characteristic sequence of A. While betting on the nth bit of the characteristic sequnece, the martingale knows the first $n-1$ bits of the characteristic sequence of A. Initially, the martingale starts with capital $d(\lambda)$. When d is ready to bet on the nth bit, it has capital $d(A|n-1)$. The martingale bets an amount a, $0 \leq a \leq d(A|n-1)$, that the next bit of the characteristic sequence is 0. If the next bit is indeed 0, then the capital of the martingale increases by a, else the capital decreases by a. More precisely, if the next bit of the characteristic sequence is 0, then $d((A|n-1)\cdot 0) = d(A|n-1) + a$, and $d((A|n-1)\cdot 1) = d(A|n-1) - a$. If the next bit is 1, then $d((A|n-1)\cdot 0) = d(A|n-1) - a$ and $d((A|n-1)\cdot 1) = d(A|n-1) + a$. Here $(A|n-1)\cdot b$ denotes the string obtained by concatanating $A|n-1$ with bit b.

Given a time bound $t(n)$, a language L is $t(n)$-*random* [6] if no $O(t(n))$-time computable martingale succeeds on L. A class of languages X has p-measure zero, written $\mu_{\mathrm{p}}(X) = 0$, if there exists a polynomial t such that every language in X is not $t(n)$-random.

Lutz suggested studying the structure of the class NP under the hypothesis "NP does not have p-measure 0," which is written $\mu_{\mathrm{p}}(\mathrm{NP}) \neq 0$. Since then several believable consequences of this hypothesis have been obtained. For a survey of these results see [23, 25].

2.2 NP-Machine Hypothesis

Our proofs crucially make use of the following hypothesis. Several variants of this hypothesis have been studied earlier [14, 16].

NP-**Machine Hypothesis.** There exists an NP-machine M and $\epsilon > 0$ such that M accepts 0^* and no 2^{n^ϵ}-time-bounded Turing machine computes infinitely many accepting computations of M.

In other words, the hypothesis says that there is no function f computable in time 2^{n^ϵ} such that for infinitely many n, $f(0^n)$ is an accepting computation of $M(0^n)$. It is known that the measure hypothesis implies the NP-machine hypothesis.

Theorem 2.1. (Hitchcock and Pavan [17]) *If $\mu_{\mathrm{p}}(\mathrm{NP}) \neq 0$, then the* NP-*machine hypothesis holds.*

A simple padding argument yields the following.

Observation 2.2. *Assume that the* NP-*machine hypothesis is true and let p be any polynomial. Then there exists an* NP-*machine N that accepts 0^*, and no $2^{p(n)}$-time-bounded machine computes infinitely many accepting computations of N.*

2.3 Reductions to P-selective Sets

A set S is p-selective if there exists a polynomial-time computable function $f : \Sigma^* \times \Sigma^* \to \Sigma^*$ such that for all strings x and y, $f(x, y) \in \{x, y\}$, and if at least one of x and y belongs to S, then $f(x, y)$ belongs to S.

Let P-sel denote the class of p-selective sets. For a reduction $\leq_\tau^\alpha$ and a class $\mathcal{C}$, let

$$R_\tau^\alpha(\mathcal{C}) = \{A \mid (\exists B \in \mathcal{C}) A \leq_\tau^\alpha B\}.$$

Theorem 2.3. (Buhrman and Longpré [10], Wang [29]) $R_{\mathrm{tt}}^{\mathrm{p}}(\text{P-sel})$ *has p-measure 0.*

Let $\leq_{\mathrm{tt}}^{t(n)\restriction\mathrm{p}}$ denote a truth-table reduction that is computable in $t(n)$ time, but where the number and length of the queries is bounded by a polynomial. It is straightforward to extend the arguments in [10] or [29] to show that Theorem 2.3 extends to these reductions when $t(n)$ is linear-exponential.

Theorem 2.4. *For every $c \in \mathbb{N}$, the class $R_{\mathrm{tt}}^{2^{cn}\restriction\mathrm{p}}(\text{P-sel})$ has* p-*measure 0.*

3 Adaptive versus Nonadaptive Reductions

We now present our solution to Problem 9 of Lutz and Mayordomo [25].

Theorem 3.1. *If $\mu_{\mathrm{p}}(\mathrm{NP}) \neq 0$, then there is a problem that is $\leq_{\mathrm{T}}^{\mathrm{p}}$-complete for* NP *but not $\leq_{\mathrm{tt}}^{\mathrm{p}}$-complete.*

Proof. Assume that $\mu_{\mathrm{p}}(\mathrm{NP}) \neq 0$. From Theorem 2.1 and Observation 2.2 we obtain an NP-machine M that accepts 0^* such that no 2^{n^2}-time machine can compute infinitely many of its accepting computations.

For each n, let a_n be the lexicographically maximum accepting computation of $M(0^n)$. Let a be the infinite sequence $a = a_0 a_1 a_2 \ldots$. Let

$$A = \{\langle x, w \rangle \mid x \in \mathrm{SAT} \text{ and } w \text{ is an accepting computation of } M(0^{|x|})\},$$

$$B = L(a) = \{x \in \Sigma^* \mid x < a\},$$

where $<$ is the standard dictionary order. Let

$$C = 0A \cup 1B.$$

Then C is $\leq_{\mathrm{T}}^{\mathrm{p}}$-complete for NP: to decide whether $x \in \mathrm{SAT}$, we can adaptively query B to find $a_{|x|}$ and then ask if $\langle x, a_{|x|} \rangle \in A$.

Suppose that C is $\leq_{\mathrm{tt}}^{\mathrm{p}}$-complete for NP. Then for every $L \in \mathrm{NP}$, $L \leq_{\mathrm{tt}}^{\mathrm{p}} C$ via some reduction (g, h). Fix such an L an (g, h).

Claim 3.2. For all but finitely many x, all queries of $g(x)$ to strings of the form $0\langle y, w \rangle$ must satisfy

- $|y| \leq |x|$, or
- w is not an accepting computation of $M(0^{|y|})$.

Proof of Claim 3.2. Consider the following algorithm.

input 0^n;
for all $x \in \{0,1\}^{<n}$:
    compute $g(x)$;
    for all queries in $g(x)$ that are of the form $0\langle y, w \rangle$, where $|y| = n$:
        if w is an accepting computation of $M(0^n)$
            output w and halt;

This algorithm runs in $O(2^n \cdot \mathrm{poly}(n))$ time, and would compute infinitely many accepting computations of M if the claim is false. □ *Claim 3.2*

Claim 3.3. $L \leq_{\mathrm{tt}}^{2^n \restriction \mathrm{p}} B$.

Proof of Claim 3.3. By Claim 3.2 and making a finite patch to the reduction, we can assume that for all x, all queries of $g(x)$ to strings of the form $0\langle y, w \rangle$ must satisfy $|y| \leq |x|$ or w is not an accepting computation of $M(0^{|y|})$.

- If $|y| \leq |x|$, then we can decide whether $\langle y, w \rangle \in A$ in 2^n time by checking if $y \in \mathrm{SAT}$ in exponential time and whether w is an accepting computation of $M(0^{|y|})$ in polynomial time.
- If $|y| > |x|$, then w is not an accepting of $M(0^{|y|})$, so we know $\langle y, w \rangle \notin A$.

We obtain a reduction to B by answering these queries to A directly. □ *Claim 3.3*

Since B is a left-cut, it is p-selective, so it follows from Claim 3.3 that $\mathrm{NP} \subseteq R_{\mathrm{tt}}^{2^n \restriction \mathrm{p}}(\text{P-sel})$. By Theorem 2.4, this implies $\mu_{\mathrm{p}}(\mathrm{NP}) = 0$, a contradiction. □

4 Nondeterministic versus Deterministic Reductions

Definition. [1, 21] A language A is *strong nondeterministic many-one reducible* to a language B, written $A \leq_{\mathrm{m}}^{\mathrm{SNP}} B$, if there is a nondeterministic polynomial-time machine M such that the following conditions hold.

- On an input x, every path of M either outputs a string y or outputs the special symbol "?". At least one path outputs a string.
- If x belongs to A, then every output y belongs to B, and if x does not belong to A, then every output y does not belong to B.

Adleman and Manders [1] also called this γ-reducibility and denoted it $\leq_\gamma$.

Long [21] showed that the following are equivalent:

- for all A, B, $A \leq_{\mathrm{m}}^{\mathrm{SNP}} B$ implies $A \leq_{\mathrm{T}}^{\mathrm{p}} B$
- every NPMV total function has a polynomial-time refinement.

The latter has been called Proposition Q in [14]. To separate $\leq_{\mathrm{m}}^{\mathrm{SNP}}$-completeness from $\leq_{\mathrm{T}}^{\mathrm{p}}$-completeness for NP, we clearly need a hypothesis that at least implies Q is false. The NP-machine hypothesis fits the bill:

Theorem 4.1. *If the* NP*-machine hypothesis holds, then there is a problem that is* $\leq_{\mathrm{m}}^{\mathrm{SNP}}$*-complete for* NP *but not* $\leq_{\mathrm{T}}^{\mathrm{p}}$*-complete.*

Proof. Assume the NP-machine hypothesis. By Observation 2.2, there exists an NP machine M that accepts 0^* for which no 2^{3n}-time bounded machine can compute infinitely many accepting computations. Consider the following language.

$$A = \{\langle x, a\rangle \mid x \in \mathrm{SAT} \text{ and } a \text{ is an accepting computation of } M(0^{|x|})\}$$

Then $A \in \mathrm{NP}$, and we claim that A is strong nondeterministic many-one complete. Consider a nondeterministic machine N that on input x guesses a string a, and if a is an accepting computation of $M(0^{|x|})$, then it outputs $\langle x, a\rangle$. If a is not an accepting computation of $M(0^{|x|})$, then N outputs "?". Then N is a strong nondeterministic many-one reduction from SAT to A. It follows that A is strong nondeterministic many-one complete for NP.

We will show that A is not Turing complete for NP. Suppose to the contrary that it is Turing complete. Consider the following language S.

$$S = \{\langle 0^n, w\rangle \mid \ w \text{ is a prefix of an accepting computation of } M(0^n)\}$$

Since S is in NP, there is a polynomial-time oracle Turing machine R such that $S = L(R^L)$. Consider the following procedure $\mathcal{A}$ that tries to compute accepting computations of M.

1. Input 0^n.
2. Set $y = \epsilon$.
3. Run $R(\langle 0^n, y0\rangle)$. When R generates a query $q = \langle x, z\rangle$, let $t = |x|$ and do the following:
   (a) If z is not an accepting computation of $M(0^t)$, then continue simulation of R with answer "No".

(b) Else, z is an accepting computation of $M(0^t)$.

(c) If $t \geq n$, then output "Unsuccessful", print z, and halt.

(d) Otherwise, decide whether $\langle x, z\rangle \in L$ by checking whether $x \in$ SAT. Since $t < n$ this takes at most 2^n time. Use this answer to continue the simulation.

4. If R accepts $\langle 0^n, y0\rangle$, then set $y = y0$. Else set $y = y1$.

5. If y is an accepting computation of $M(0^n)$, then output y and halt. Else, GoTo Step 3.

Observe that the most expensive step in the above computation is Step 3d. This takes 2^n time. Since this step is repeated at most polynomial number of steps, the above algorithm halts in 2^{2n} steps.

Next we make two claims about the behavior of the algorithm $\mathcal{A}$.

Claim 4.2. If $\mathcal{A}(0^n)$ outputs "Unsuccessful" for infinitely many n, then there is a 2^{3n}-time algorithm that outputs infinitely many accepting computations of $M(0^n)$.

Proof of Claim 4.2. Observe that if $\mathcal{A}(0^n)$ outputs "Unsuccessful", then there exists a $t \geq n$ such that $\mathcal{A}(0^n)$ outputs an accepting computation of $M(0^t)$. Thus if there exist infinitely many n for which $\mathcal{A}(0^n)$ outputs "Unsuccessful", then there exists infinitely many t for which there exists $n \leq t$, and $\mathcal{A}(0^n)$ outputs an accepting computation of $M(0^t)$. Now consider the following algorithm: On input 0^t, run $\mathcal{A}(0^j)$, $1 \leq j \leq t$. If any of the runs of $\mathcal{A}$ outputs an accepting computation of $M(0^t)$, then output that accepting computation.

This algorithm outputs an accepting computation of $\mathcal{A}(0^t)$ for infinitely many t. The running time of the algorithm is bounded by $\sum_{j=1}^{t} 2^{2j} \leq 2^{3t}$. This establishes the claim. $\square$ *Claim 4.2*

Claim 4.3. If $\mathcal{A}(0^n)$ does not output "Unsuccessful", then it outputs an accepting computation of $M(0^n)$ in time 2^{2n}.

Proof of Claim 4.3. Observe that $\mathcal{A}(0^n)$ is trying to compute an accepting computation of $M(0^n)$ by doing a prefix search. This is accomplished by running the Turing reduction R, and whenever the reduction generates a query it is attempting to find the answer to the query without actually making the query. Thus if all the queries are answered correctly, it will compute an accepting computation of $M(0^n)$. We argue that $\mathcal{A}(0^n)$ computes all query answers correctly. Let $q = \langle x, y\rangle$ be a query that is generated.

If y is not an accepting computation of M, then q does not belong to A. Thus $\mathcal{A}$ answers the query correctly in 3a. So assume y is an accepting computation of $M(0^t)$. Since $\mathcal{A}(0^n)$ does not output "Unsuccessful", $t < n$. Thus the algorithm reaches Step 3d. In this step, it decides whether $x \in$ SAT by a running a deterministic algorithm for SAT. Thus the query answer is computed correctly in this step.

Thus $\mathcal{A}(0^n)$ computes all query answers correctly. Thus $\mathcal{A}(0^n)$ outputs an accepting computation of $M(0^n)$. Recall that the running time of $\mathcal{A}$ is bounded by 2^{2n}. $\square$ *Claim 4.3*

Now, if $\mathcal{A}(0^n)$ outputs "Unsuccessful" for infinitely many n, then, by Claim 4.2, there is a 2^{3n}-time algorithm that computes infinitely many accepting computations of $M(0^n)$. This contradicts the NP-machine hypothesis. Thus for all but finitely many n, $\mathcal{A}(0^n)$ does not output "Unsuccessful". Thus, by Claim 4.3, for all but finitely many n, $\mathcal{A}(0^n)$ outputs an accepting computation of $M(0^n)$ in time 2^{2n}. This again contradicts the NP-machine hypothesis.

Thus there is no Turing reduction from S to A. Thus A is not Turing complete for NP. $\square$

By Theorem 2.1, we immediately have the following.

Corollary 4.4. *If $\mu_{\mathrm{p}}(\mathrm{NP}) \neq 0$, there is a problem that is $\leq_{\mathrm{m}}^{\mathrm{SNP}}$-complete for* NP *but not $\leq_{\mathrm{T}}^{\mathrm{p}}$-complete.*

5 Length-Increasing Reductions and Polynomial-Size Circuits

In this section we study one-to-one, length-increasing reductions. (All reductions in this section are many-one reductions. We say that a many-one reduction f is *length-increasing* if $|f(x)| > |x|$ for all strings x and that f is *one-to-one* if for all strings $x \neq y$, $f(x) \neq f(y)$.)

Berman proved [7] that every $\leq_{\mathrm{m}}^{\mathrm{p}}$-complete set for E is also is complete under one-to-one, length-increasing reductions. This proof makes essential use of the fact that E is closed under complementation, so it does not go through for nondeterministic classes. As a partial result, Ganesan and Homer [15] showed that every $\leq_{\mathrm{m}}^{\mathrm{p}}$-complete set for NE is complete under one-to-one, exponentially-honest reductions. See also the survey paper [20] by Homer.

Agrawal [3] showed that if one-way permutations exist, then many-one complete sets for NP and NE are complete via one-to-one, length-increasing, p/poly reductions. (A p/poly reduction is computed by a nonuniform family of polynomial-size circuits, one for each input length.) We now show that Agrawal's result for NE can be made unconditional. Our original proof [18] of this used the fact coNE $\subseteq$ NE/poly to apply Berman's technique. The following simpler proof was described to us by Fortnow (personal communication, 2006) and instead uses the result of Ganesan and Homer.

Theorem 5.1. *Every $\leq_{\mathrm{m}}^{\mathrm{p}}$-complete set for* NE *is complete under one-to-one, length-increasing,* p/poly *reductions.*

Proof. Let A be an arbitrary $\leq_{\mathrm{m}}^{\mathrm{p}}$-complete set for NE and let K be the standard complete set. By [15], there is a one-to-one $\leq_{\mathrm{m}}^{\mathrm{p}}$-reduction f from K to A. Let $\mathrm{pad} : \Sigma^* \times \Sigma^* \to \Sigma^*$ be a one-to-one, polynomial-time computable padding function such that for all strings x and r, $x \in K$ if and only if $\mathrm{pad}(x, r) \in K$. Because f and pad are one-to-one, for each n there is some $r_n \in \Sigma^{n+1}$ such that $|f(\mathrm{pad}(x, r_n))| > |x|$ for all $x \in \Sigma^n$. We use this r_n as our advice to define the one-to-one, length-increasing, p/poly reduction $g(x) = f(\mathrm{pad}(x, r_n))$ from K to A. □

Next we will show that if NP does not have p-measure zero, then all NP-complete sets are complete via length-increasing, p/poly reductions. In the proof we will consider whether a language R has the following property.

Property 5.2. There is a 2^{cn}-time computable function f such that for every n, $f(0^n)$ either outputs $\perp$ or outputs a tuple $\langle a, b, u, v\rangle$. For infinitely many n, $f(0^n) \neq \perp$. Whenever $f(0^n) = \langle a, b, u, v\rangle$, the following hold:

- $|a| = |b| = n$.
- $R(a)R(b) \neq uv$, and uv is either 00 or 11.

Informally, f either finds two strings such that at least one of them is in R, or finds two strings such that at least one of them does not belong to R.

Lemma 5.3. *If R has Property 5.2, then R is not n^c-random.*

Proof. We describe a martingale d that can win an infinite amount of money while betting on R. Let $d(n)$ denote the amount of money that the martingale has before it starts betting on strings of length n. Before starting betting on strings of length n, the martingale runs $f(0^n)$. If $f(0^n) = \perp$, then d does not bet on any string of length n. Suppose $f(0^n) = \langle a, b, u, v\rangle$. Without loss of generality we can assume $a < b$. Consider the case $uv = 00$. In this case at least one of a and b must be in R. The martingale bets 1/3rd of its amount on $a \in R$. If a really belongs to R, then d does not bet on any other string of length n. So if $a \in R$, then $d(n+1) = 4d(n)/3$. However, if $a \notin R$, then d is left with capital $2d(n)/3$. However, since at least one of a and b must be in R, b must belong to R. Now d bets all its money on $b \in R$. Thus in this case also $d(n+1) = 4d(n)/3$. The case $uv = 11$ is handled via a symmetric argument.

Since $f(0^n) \neq \perp$ for infinitely many n, for infinitely many n, $d(n+1) \geq 4d(n)/3$. Thus $d(n)$ approaches infinity as n tends to ∞. Since f runs in 2^{cn}-time, d runs in time $O(n^c)$. Thus R is not n^c-random. □

Now we are ready to prove the theorem regarding complete sets for NP.

Theorem 5.4. *If* $\mu_{\mathrm{p}}(\mathrm{NP}) \neq 0$*, then every* NP*-complete language is complete under length-increasing,* p/poly *reductions.*

Proof. Let L be any NP-complete language. We show that there is a p/poly, length-increasing reduction from SAT to L. We first define an intermediate language S such that SAT is p/poly, length-increasing reducible to S, and S is honest polynomial-time reducible to L. Combining these two reductions and using the paddability of SAT we obtain the desired reduction from SAT to L. Let $L \in \mathrm{DTIME}(2^{n^k})$.

If NP does not have p-measure 0, then there is an n^4-random language R in NP. The randomness of R implies that both R and $\overline{R}$ have at least one string at each length. Let

$$S = \{\langle x, y, z\rangle \mid |x| = |y| = |z| \text{ and } \mathtt{MAJ}\{x \in R, y \in \mathrm{SAT}, z \in R\} = 1\}.$$

Here $\mathtt{MAJ}\{\phi, \psi, \tau\} = 1$ if a majority of ϕ, ψ, and τ are true.

It is clear that S is NP. For every n, fix two strings a_n and b_n of length n such that $a_n \in R$ and $b_n \notin R$. Consider the following reduction from SAT to S: Given an input y of length n the reduction outputs $\langle a_n, y, b_n\rangle$. Now $y \in \mathrm{SAT} \Leftrightarrow \langle a_n, y, b_n\rangle \in S$. The reduction takes a_n and b_n as advice. It is clear that this reduction is length-increasing. Therefore we have established that SAT is p/poly, length-increasing reducible to S.

Since S is in NP and L is NP-complete, there is a many-one reduction f from S to L. We now argue that f must be a honest reduction on strings of form $\langle x, y, z\rangle$ where $|x| = |y| = |z|$.

Claim 5.5. Let $T = \{\langle x, y, z\rangle \mid |x| = |y| = |z|\}$. For all but finitely many strings $w = \langle x, y, z\rangle$ from T, $|f(w)| \geq |x|^{1/k}$.

Proof of Claim 5.5. Consider the following set

$$U = \left\{ w = \langle x, y, z\rangle \in T \,\middle|\, |f(w)| < |x|^{1/k} \right\}.$$

We show that if U is infinite, then R has Property 5.2.

Recall that L can be decided in time 2^{n^k}. Thus if a string $w = \langle x, y, z\rangle$ belongs to U, then the membership of $f(w)$ in L can be decided in time $2^{|f(w)|^k} < 2^{|x|}$. Since f is a many-one reduction from S to L, for every string $w = \langle x, y, z\rangle$ in U, its membership in S can be computed in time $2^{|x|}$.

Define a function g as follows. In input 0^n, cycle through all tuples $w = \langle x, y, z\rangle$, $|x| = |y| = |z| = n$, and check if $w \in U$ by computing $f(w)$. If none of the w's are in U, then output $\perp$. Else, let $w = \langle x, y, z \rangle$ be the first string that belongs to U. Compute the membership of w in S. We first consider the case $w \in S$. In this case,

$$\mathtt{MAJ}\{x \in R, y \in \mathrm{SAT}, z \in R\} = 1.$$

Thus it can not be the case that both x and z are out of R. Then g outputs $\langle x, z, 0, 0 \rangle$. Similarly, if $w \notin S$, then it cannot be the case that both x and z are in R. Then g outputs $\langle x, z, 1, 1 \rangle$.

Observe that the running time of g is bounded by $O(2^{3n})$. If U is infinite, then for infinitely many n, $g(0^n) \neq \perp$. So, if U is infinite, then R has Property 5.2, and by Lemma 5.3, R is not n^3-random. Since R is n^4-random, U is finite.

Thus for all but finitely many strings from T, $|f(w)| \geq |x|^{1/k}$. □ *Claim 5.5*

Now consider the following reduction h from SAT to L: On input y of length n, output $f(\langle a_n, y, b_n \rangle)$. By Claim 5.5, $|f(\langle a_n, y, b_n \rangle)| \geq n^{1/k}$. Thus h is an honest, p/poly-reduction from SAT to L. Since SAT is paddable, there exists a reduction from SAT to itself that maps strings of length n to strings of length at least n^k. Combining this reduction with h, we obtain a a length-increasing, p/poly-reduction from SAT to L. Thus L is complete via length-increasing, p/poly reductions. □

6 Conclusion

We now know that the measure hypothesis separates nearly all polynomial-time completeness notions for NP. It would be interesting to separate completeness notions for NP under weaker hypotheses such as "NP is hard on average". Can we separate Turing completeness from many-one completeness under a hypothesis that is weaker than the measure hypothesis? More specifically, can we achieve the separation under the NP-machine hypothesis?

Theorem 4.1 gives evidence that when we give more resources to the reductions, we obtain a richer class of complete sets. What happens when we decrease the resource bound of the reductions? Agrawal et al. [4, 2] showed that NC^0-completeness and AC^0-completeness for NP coincide whereas AC^0-completeness and AC^0[mod 2]-completeness for NP differ. It would be interesting to extend these results to other resource bounds.

The results of Agrawal [3] and our results in Section 5 indicate that complete sets for NP and NE are complete under one-to-one, length-increasing reductions. However these reductions need polynomial-size advice. Can we eliminate the advice?

Acknowledgments. We thank the anonymous referees for very helpful comments and corrections. We also thank the anonymous reviewers from ICALP 2006 for their comments on a preliminary version of this paper. We thank Lance Fortnow for the simpler proof of Theorem 5.1.

References

[1] L. Adleman and K. Manders. Reducibility, randomness, and intractability. In *Proceedings of the 9th ACM Symposium on Theory of Computing*, pages 151–163, 1977.

[2] A. Agrawal, E. Allender, R. Impagliazzo, T. Pitassi, and S. Rudich. Reducing the complexity of reductions. *Computational Complexity*, 10(2):117–138, 2001.

[3] M. Agrawal. Pseudo-random generators and structure of complete degrees. In *17th Annual IEEE Conference on Computational Complexity*, pages 139–145, 2002.

[4] M. Agrawal, E. Allender, and S. Rudich. Reductions in circuit complexity: An isomorphism theorem and a gap theorem. *Journal of Computer and System Sciences*, 57(2):127–143, 1998.

[5] K. Ambos-Spies and L. Bentzien. Separating NP-completeness notions under strong hypotheses. *Journal of Computer and System Sciences*, 61(3):335–361, 2000.

[6] K. Ambos-Spies, S. A. Terwijn, and X. Zheng. Resource bounded randomness and weakly complete problems. *Theoretical Computer Science*, 172(1–2):195–207, 1997.

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

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

[9] H. Buhrman, S. Homer, and L. Torenvliet. Completeness notions for nondeterministic complexity classes. *Mathematical Systems Theory*, 24(1):179–200, 1991.

[10] H. Buhrman and L. Longpré. Compressibility and resource bounded measure. *SIAM Journal on Computing*, 31(3):876–886, 2002.

[11] H. Buhrman and L. Torenvliet. On the structure of complete sets. In *Proceedings of the Ninth Annual Structure in Complexity Theory Conference*, pages 118–133. IEEE Computer Society, 1994.

[12] H. Buhrman and L. Torenvliet. Complete sets and structure in subrecursive classes. In *Logic Colloquium '96*, volume 12 of *Lecture Notes in Logic*, pages 45–78. Association for Symbolic Logic, 1998.

[13] M. J. Chung and B. Ravikumar. Strong nondeterministic Turing reduction—a technique for proving intractability. *Journal of Computer and System Sciences*, 39(1):2–20, 1989.

[14] S. Fenner, L. Fortnow, A. Naik, and J. Rogers. Inverting onto functions. *Information and Computation*, 186(1):90–103, 2003.

[15] K. Ganesan and S. Homer. Complete problems and strong polynomial reducibilities. *SIAM Journal on Computing*, 21(4):733–742, 1992.

[16] L. Hemaspaandra, J. Rothe, and G. Wechsung. Easy sets and hard certificate schemes. *Acta Informatica*, 34(11):859–879, 1997.

[17] J. M. Hitchcock and A. Pavan. Hardness hypotheses, derandomization, and circuit complexity. In *Proceedings of the 24th Conference on Foundations of Software Technology and Theoretical Computer Science*, pages 336–347. Springer-Verlag, 2004.

[18] J. M. Hitchcock and A. Pavan. Comparing reductions to NP-complete sets. Technical Report TR06-039, Electronic Colloquium on Computational Complexity, 2006.

[19] J. M. Hitchcock, A. Pavan, and N. V. Vinodchandran. Partial bi-immunity, scaled dimension, and NP-completeness. *Theory of Computing Systems*. To appear.

[20] S. Homer. Structural properties of complete sets for exponential time. In L. A. Hemaspaandra and A. L. Selman, editors, *Complexity Theory Retrospective II*, pages 135–153. Springer-Verlag, 1997.

[21] T. J. Long. On γ-reducibility versus polynomial time many-one reducibility. *Theoretical Computer Science*, 14:91–101, 1981.

[22] J. H. Lutz. Almost everywhere high nonuniform complexity. *Journal of Computer and System Sciences*, 44(2):220–258, 1992.

[23] J. H. Lutz. The quantitative structure of exponential time. In L. A. Hemaspaandra and A. L. Selman, editors, *Complexity Theory Retrospective II*, pages 225–254. Springer-Verlag, 1997.

[24] J. H. Lutz and E. Mayordomo. Cook versus Karp-Levin: Separating completeness notions if NP is not small. *Theoretical Computer Science*, 164(1–2):141–163, 1996.

[25] J. H. Lutz and E. Mayordomo. Twelve problems in resource-bounded measure. *Bulletin of the European Association for Theoretical Computer Science*, 68:64–80, 1999. Also in *Current Trends in Theoretical Computer Science: Entering the 21st Century*, pages 83–101, World Scientific Publishing, 2001.

[26] A. Pavan. Comparison of reductions and completeness notions. *SIGACT News*, 34(2):27–41, June 2003.

[27] A. Pavan and A. Selman. Separation of NP-completeness notions. *SIAM Journal on Computing*, 31(3):906–918, 2002.

[28] A. Pavan and A. Selman. Bi-immunity separates strong NP-completeness notions. *Information and Computation*, 188(1):116–126, 2004.

[29] Y. Wang. NP-hard sets are superterse unless NP is small. *Information Processing Letters*, 61(1):1–6, 1997.

[30] O. Watanabe. A comparison of polynomial time completeness notions. *Theoretical Computer Science*, 54:249–265, 1987.