Separation of NP-completeness Notions
(Extended Abstract)

A. Pavan* Alan L Selman†

December 6, 2000

Abstract

We use hypotheses of structural complexity theory to separate various NP-completeness notions. In particular, we introduce an hypothesis from which we describe a set in NP that is $\leq_{\mathrm{T}}^{\mathrm{P}}$-complete but not $\leq_{tt}^{\mathrm{P}}$-complete. We provide fairly thorough analyses of the hypotheses that we introduce.

1 Introduction

Ladner, Lynch, and Selman [LLS75] were the first to compare the strength of polynomial-time reducibilities. They showed, for the common polynomial-time reducibilities, Turing ($\leq_{\mathrm{T}}^{\mathrm{P}}$), truth-table ($\leq_{tt}^{\mathrm{P}}$), bounded truth-table ($\leq_{btt}^{\mathrm{P}}$), and many-one ($\leq_{m}^{\mathrm{P}}$), that

$$\leq_{m}^{\mathrm{P}} \prec \leq_{btt}^{\mathrm{P}} \prec \leq_{tt}^{\mathrm{P}} \prec \leq_{\mathrm{T}}^{\mathrm{P}},$$

where $\leq_{r}^{\mathrm{P}} \prec \leq_{s}^{\mathrm{P}}$ means that $\leq_{r}^{\mathrm{P}}$ is *properly stronger* than $\leq_{s}^{\mathrm{P}}$; that is, $A \leq_{r}^{\mathrm{P}} B$ implies $A \leq_{s}^{\mathrm{P}} B$, but the converse does not hold. In each case, the verifying sets belong to $\mathrm{E} = \mathrm{DTIME}(2^n)$. Ladner, Lynch, and Selman raised the obvious question of whether reducibilities differ on NP. If there exist sets A and B in NP (other than the empty set or Σ^*) such that $A \leq_{\mathrm{T}}^{\mathrm{P}} B$ but $A \not\leq_{m}^{\mathrm{P}} B$, then, of course, $\mathrm{P} \neq \mathrm{NP}$ follows immediately. With this in mind, they conjectured that $\mathrm{P} \neq \mathrm{NP}$ implies that $\leq_{\mathrm{T}}^{\mathrm{P}}$ and $\leq_{m}^{\mathrm{P}}$ differ on NP.

In the intervening years, many results have explained the behavior of polynomial-time reducibilities within other complexity classes and have led to a complete understanding of the completeness notions that these reducibilities induce. For example, Ko and Moore [KM81] demonstrated the existence of $\leq_{\mathrm{T}}^{\mathrm{P}}$-complete sets for EXP that are not $\leq_{m}^{\mathrm{P}}$-complete. Watanabe [Wat87] extended this result significantly, showing that $\leq_{1-tt}^{\mathrm{P}}$-, $\leq_{btt}^{\mathrm{P}}$-, $\leq_{tt}^{\mathrm{P}}$-, and $\leq_{\mathrm{T}}^{\mathrm{P}}$-completeness for EXP are mutually different, while Homer, Kurtz, and Royer [KR93] proved that $\leq_{m}^{\mathrm{P}}$- and $\leq_{1-tt}^{\mathrm{P}}$-completeness are identical.

However, there have been few results comparing reducibilities within NP, and we have known very little concerning various notions of NP-completeness. It is surprising that no NP-complete

*Department of Computer Science and Engineering, University at Buffalo, Buffalo, NY 14260. Email: aduri@cse.buffalo.edu

†Department of Computer Science and Engineering, University at Buffalo, Buffalo, NY 14260. Email: selman@cse.buffalo.edu

problem has been discovered that requires anything other than many-one reducibility for proving its completeness. The first result to distinguish reducibilities within NP is an observation of Wilson in one of Selman's papers on p-selective sets [Sel82]. It is a corollary of results there that if $\mathrm{NE} \cap co\text{-}\mathrm{NE} \neq \mathrm{E}$, then there exist sets A and B belonging to NP such that $A \leq_{ptt}^{\mathrm{P}} B$, $B \leq_{tt}^{\mathrm{P}} A$, and $B \not\leq_{ptt}^{\mathrm{P}} A$, where $\leq_{ptt}^{\mathrm{P}}$ denotes positive truth-table reducibility. Regarding completeness, Longpré and Young [LY90] proved that there are $\leq_m^{\mathrm{P}}$-complete sets for NP for which $\leq_{\mathrm{T}}^{\mathrm{P}}$-reductions to these sets are *faster*, but they did not prove that the completeness notions differ. The first to give technical evidence that $\leq_{\mathrm{T}}^{\mathrm{P}}$- and $\leq_m^{\mathrm{P}}$-completeness for NP differ are Lutz and Mayordomo [LM96], who proved that if the p-measure of NP is not zero, then there exists a $\leq_{3\text{-}tt}^{\mathrm{P}}$-complete set that is not $\leq_m^{\mathrm{P}}$-complete. Ambos-Spies and Bentzien [ASB97] extended this result significantly. They used an hypothesis of resource-bounded category theory that is weaker than that of Lutz and Mayordomo to separate nearly all NP-completeness notions for the bounded truth-table reducibilities.

It has remained an open question as to whether we can separate NP-completeness notions without using hypotheses that involve essentially stochastic concepts. Furthermore, the only comparisons of reducibilities within NP known to date have been those just listed.

Here we report some exciting new progress on these questions. Our main new result introduces a strong, but reasonable, hypothesis to prove existence of a $\leq_{\mathrm{T}}^{\mathrm{P}}$-complete set in NP that is not $\leq_{tt}^{\mathrm{P}}$-complete. Our result is the first to provide evidence that $\leq_{tt}^{\mathrm{P}}$-completeness is weaker than $\leq_{\mathrm{T}}^{\mathrm{P}}$-completeness. Let Hypothesis H be the following assertion: There is a UP-machine M that accepts 0^* such that (i) no polynomial time-bounded Turing machine correctly computes infinitely many accepting computations of M, and (ii) for some $\varepsilon > 0$, no $2^{n^{\varepsilon}}$ time-bounded Turing machine correctly computes all accepting computations of M. Hypothesis H is similar to, but seemingly stronger than, hypotheses considered by researchers previously, notably Fenner, Fortnow, Naik, and Rogers [FFNR96], Hemaspaandra, Rothe and Wechsung [HRW97], and Fortnow, Pavan, and Selman [FPS99].

This result is especially interesting because the measure theory and category theory techniques seem to be successful primarily for the nonadaptive reducibilities. We will prove an elegant characterization of the genericity hypothesis of Ambos-Spies and Bentzien and compare it with Hypothesis H. Here, somewhat informally, let us say this: The genericity hypothesis asserts existence of a set L in NP such that no 2^{2n} time-bounded Turing machine can correctly *predict* membership of infinitely many x in L from the initial characteristic sequence $L|x = \{y \in L \mid y < x\}$. That is, L is almost-everywhere unpredictable within time 2^{2n}. Clearly such a set L is 2^{2n}-bi-immune. In contrast, we show that Hypothesis H holds if there is a set L in $\mathrm{UP} \cap \mathrm{co\text{-}UP}$ such that L is P-bi-immune and $L \cap 0^*$ is not in $\mathrm{DTIME}(2^{n^{\varepsilon}})$, for some $\varepsilon > 0$. Thus, we replace "almost-everywhere unpredictable" with P-bi-immunity and we lower the time bound from 2^{2n} to $2^{n^{\varepsilon}}$, but we require L to belong to $\mathrm{UP} \cap \mathrm{co\text{-}UP}$ rather than NP.

We prove several other separations as well, and some with significantly weaker hypotheses. For example, we prove that NP contains $\leq_{\mathrm{T}}^{\mathrm{P}}$-complete sets that are not $\leq_m^{\mathrm{P}}$-complete, if $\mathrm{NP} \cap \mathrm{co\text{-}NP}$ contains a set that is $2^{n^{\varepsilon}}$-bi-immune, for some $\varepsilon > 0$.

The body of this extended abstract contains a proof of the main result and selected other proofs that give a flavor of our proof techniques. The Appendix contains additional proofs.

2 Preliminaries

We use standard notation for polynomial-time reductions [LLS75], and we assume that readers are familiar with Turing, $\leq_{\mathrm{T}}^{\mathrm{P}}$, and many-one, $\leq_{m}^{\mathrm{P}}$, reducibilities. A set A is *truth-table* reducible to a set B (in symbols $A \leq_{tt}^{P} B$) if there exist polynomial-time computable functions g and h such that on input x, $g(x)$ is a set of queries $Q = \{q_1, q_2, \cdots, q_k\}$, and $x \in A$ if and only if $h(x, B(q_1), B(q_2), \cdots, B(q_k)) = 1$. The function g is the *truth-table generator* and h is the *truth-table evaluator*. For a constant $k > 0$, A is *k-truth-table* reducible to B ($A \leq_{k\text{-}tt}^{P} B$) if for all x, $\|Q\| = k$, and A is *bounded-truth-table* reducible to B ($A \leq_{btt}^{\mathrm{P}} B$) if there is a constant $k > 0$ such that $A \leq_{k\text{-}tt}^{P} B$. Given a polynomial-time reducibility $\leq_{r}^{\mathrm{P}}$, recall that a set S is $\leq_{r}^{\mathrm{P}}$-complete for NP if $S \in \mathrm{NP}$ and every set in NP is $\leq_{r}^{\mathrm{P}}$-reducible to S.

Recall that a set L is *p-selective* if there exists a polynomial-time computable function $f : \Sigma^* \times \Sigma^* \to \Sigma^*$ such that for all x and y, $f(x,y) \in \{x,y\}$ and $f(x,y)$ belongs to L, if either $x \in L$ or $y \in L$ [Sel79]. The function f is called a *selector* for L.

Given a finite alphabet, let Σ^ω denote the set of all strings of infinite length of order type ω. For $r \in \Sigma^* \cup \Sigma^\omega$, the standard left cut of r [Sel79, Sel82] is the set

$$L(r) = \{x \mid x < r\},$$

where $<$ is the ordinary dictionary ordering of strings with 0 less than 1. It is obvious that every standard left cut is p-selective with selector $f(x,y) = \min(x,y)$.

Given a p-selective set L such that the function f defined by $f(x,y) = \min(x,y)$ is a selector for L, we call f a *min-selector* for L. We will use the following simplified version of a lemma of Toda [Tod91].

Lemma 1 *Let L be a p-selective set with a min-selector f. For any finite set Q there exists a string $z \in Q \cup \{\perp\}$ such that $Q \cap L = \{y \in Q \mid y \leq z\}$ and $Q \cap \overline{L} = \{y \in Q \mid y > z\}$. The string z is called a "pivot" string.*

Now we review various notions related to almost-everywhere hardness. A language L is *immune* to a complexity class $\mathcal{C}$, or *$\mathcal{C}$-immune*, if L is infinite and no infinite subset of L belongs to $\mathcal{C}$. A language L is *bi-immune* to a complexity class $\mathcal{C}$, or *$\mathcal{C}$-bi-immune*, if L is infinite, no infinite subset of L belongs to $\mathcal{C}$, and no infinite subset of $\overline{L}$ belongs to $\mathcal{C}$. A language is DTIME$(T(n))$-*complex* if L does not belong to DTIME$(T(n))$ almost everywhere; that is, every Turing machine M that accepts L runs in time greater than $T(|x|)$, for all but finitely many words x. Balcázar and Schöning [BS85] proved that for every time-constructible function T, L is DTIME$(T(n))$-complex if and only if L is bi-immune to DTIME$(T(n))$.

Given a time bound $T(n)$, a language L is *$T(n)$-printable* if there exists a $T(n)$ time-bounded Turing machine that, on input 0^n, prints all elements of $L \cap \Sigma^{=n}$ [HY84]. A set S is *$T(n)$-printable-immune* if S is infinite and no infinite subset of S is $T(n)$-printable.

In order to compare our hypotheses with the genericity hypothesis we describe time-bounded genericity [ASFH87]. For this purpose, we follow the exposition of Ambos-Spies, Neis, and Terwijn [ASNT96]. Given a set A and string x, $A|x = \{y \mid y < x \text{ and } y \in A\}$. Let $\Sigma^* = \{z_n\}_n$, where z_n is the n-th string in lexicographic order. We identify the initial segment $A|z_n$ with its characteristic sequence; i.e., $A|z_n = A(z_0) \cdots A(z_{n-1})$. A *condition* is a set $C \subseteq \Sigma^*$. *A meets C* if for some x, the characteristic sequence $A|x \in C$. C is *dense along A* if for infinitely many strings x there exists

$i \in \{0,1\}$ such that the concatenation $(A|x)i \in C$. Then, the set A is DTIME$(t(n))$-*generic* is A meets every condition $C \in$ DTIME$(t(n))$ that is dense along A. To simplify the notation, we say that A is $t(n)$-*generic* if it is DTIME$(t(n))$-generic.

Finally, we briefly describe the Kolmogorov complexity of a finite string. Later we will use this in an oracle construction. The interested reader should refer to Li and Vitanyi [LV97] for an in-depth study. Fix a universal Turing machine U. Given a string x and and a finite set $S \subseteq \Sigma^*$, the *Kolmogorov complexity of x with respect to S* is defined by

$$K(x|S) = \min\{|p| \mid U(p,S) = x\}.$$

If $S = \emptyset$, then $K(x|S)$ is called the *Kolmogorov complexity* of x, denoted $K(x)$. We will use time-bounded Kolmogorov complexity $K^t(x)$ also. For this definition, we require that $U(p)$ runs in at most $t(|x|)$ steps.

3 Separation Results

Let Hypothesis H be the following assertion:

Hypothesis H: There is a UP-machine M that accepts 0^* such that

1. no polynomial time-bounded Turing machine correctly computes infinitely many accepting computations of M, and
2. for some $\varepsilon > 0$, no 2^{n^ε} time-bounded Turing machine correctly computes all accepting computations of M.

Theorem 1 *If Hypothesis H is true, then there exists a* $\leq^{\mathrm{P}}_{\mathrm{T}}$*-complete language for* NP *that is not* $\leq^{\mathrm{P}}_{tt}$*-complete for* NP.

Proof. Let M be a UP-machine that satisfies the conditions of Hypothesis H. For each $n \geq 0$, let a_n be the unique accepting computation of M on 0^n, and let $l_n = |a_n|$. Define the language

$$L_1 = \{\langle x, a_n\rangle \mid |x| = n, \text{ and } x \in \mathrm{SAT}\}.$$

Define the infinite string $a = a_1 a_2 \ldots$, and define

$$L_2 = L(a) = \{x \mid x < a\}$$

to be the standard left-cut of a.

We define $L = L_1 \oplus L_2$ to be the disjoint union of L_1 and L_2. We will prove that L is $\leq^{\mathrm{P}}_{\mathrm{T}}$-complete for NP but not $\leq^{\mathrm{P}}_{tt}$-complete.

Lemma 2 *L is* $\leq^{\mathrm{P}}_{\mathrm{T}}$*-complete for* NP.

Proof. It is clear that L belongs to NP. The following reduction witnesses that SAT$\leq^{\mathrm{P}}_{\mathrm{T}}L$: Given an input string x, where $|x| = n$, use a binary search algorithm that queries L_2 to find a_n. Then, note that $x \in$ SAT if and only if $\langle x, a_n\rangle$ belongs to L_1. ∎

Lemma 3 *L is not $\leq_{tt}^{\mathrm{P}}$-complete for* NP.

Proof. Assume that L is $\leq_{tt}^{\mathrm{P}}$-complete for NP. Define the set

$$S = \{\langle 0^n, i\rangle \mid \text{ the } i\text{-th bit of } a_n = 1\ \}.$$

Clearly, S belongs to NP. Thus, by our assumption, there is a $\leq_{tt}^{\mathrm{P}}$-reduction $\langle g, h\rangle$ from S to L. Given this reduction, we will derive a contradiction to Hypothesis H.

Consider the following procedure $\mathcal{A}$:

1. input 0^n;
2. compute the sets $Q^j = g(\langle 0^n, j\rangle)$, for $1 \leq j \leq l_n$. Let $Q = \bigcup\{Q^j \mid 1 \leq j \leq l_n\}$;
3. Let Q_1 be the set of all queries in Q to L_1 and let Q_2 be the set of all queries in Q to L_2 ($Q = Q_1 \cup Q_2$);
4. If Q_1 contains a query $\langle x, a_t\rangle$, where $t \geq n^{\varepsilon}$, then output "Unsuccessful" and Print a_t, else output "Successful".

Observe that this procedure runs in polynomial time. We treat two cases, namely, either $\mathcal{A}(0^n)$ is unsuccessful, for infinitely many n, or it is successful, for all but finitely many n.

Claim 1 *If the procedure $\mathcal{A}(0^n)$ is unsuccessful for infinitely many n, then there is a polynomial time-bounded Turing machine that correctly computes infinitely many accepting computations of M, thereby contradicting Clause 1 of Hypothesis H.*

Proof. If $\mathcal{A}(0^n)$ is unsuccessful, then it outputs a string a_t such that $t \geq n^{\varepsilon}$. Hence, if $\mathcal{A}(0^n)$ is unsuccessful for infinitely many n, then for infinitely many t there exists an n, where $n \leq t^{1/\varepsilon}$, and $\mathcal{A}(0^n)$ outputs a_t. The following procedure uses this observation to compute infinitely many accepting computations of M in polynomial time.

input: 0^t
for $j = 1$ to $j = t^{1/\varepsilon}$ **do**
    if $\mathcal{A}(0^j)$ outputs a_t
        then output a_t and halt.

The procedure runs in polynomial time because the procedure $\mathcal{A}(0^j)$ runs in polynomial time. ∎

Claim 2 *If $\mathcal{A}(0^n)$ is successful for all but finitely many n, then there is a $2^{n^{\varepsilon}}$ time-bounded Turing machine that correctly computes all accepting computations of M, thereby contradicting Clause 2 of Hypothesis H.*

Proof. We will demonstrate a procedure $\mathcal{B}$ such that for each n, if $\mathcal{A}(0^n)$ is successful, then $\mathcal{B}$ on input 0^n outputs the accepting computation of M on 0^n in $2^{n^{\varepsilon}}$ time.

If $\mathcal{A}(0^n)$ is successful, then no member of the set Q_1 is of the form $\langle x, a_t\rangle$ where $t \geq n^{\varepsilon}$. We begin our task with the following procedure $\mathcal{C}$ that for each query $q = \langle y, z\rangle$ in Q_1 decides whether $q \in L_1$.

1. input $q = \langle y, z \rangle$;

2. If $z \neq a_t$ for some t, then $\langle y, z \rangle$ does not belong to L_1; (This can be determined in polynomial time.)

3. if $z = a_t$, where $t \leq n^{\varepsilon}$, then $\langle y, z \rangle$ belongs to L_1 only if $|y| = t$ and y belongs to SAT. (Since $t \leq n^{\varepsilon}$ this step can be done in time $2^{n^{\varepsilon}}$).

Thus, $\mathcal{C}$ decides membership in L_1 for all queries q in Q_1. Therefore, if for each query q in Q_2, we can decide whether q belongs to L_2, then the evaluator h can determine whether each input $\langle 0^n, j \rangle$, $1 \leq j \leq l_n$, belongs to S. That is, if for each query q in Q_2, we can decide whether q belongs to L_2, then we can compute a_n. We can accomplish this using a standard proof technique for p-selective sets [HNOS96, Tod91]. Namely, since L_2 is a standard left-cut, by Lemma 1, there exists a pivot string z in $Q_2 \cup \{\perp\}$ such that $Q_2 \cap L_2$ is the set of all strings in Q_2 that are less than or equal to z. We do not know which string is the pivot string, but there are only $\|Q_2\|$ choices, which is a polynomial number of choices. Thus, procedure $\mathcal{B}$ on input 0^n proceeds as follows to compute a_n: For each possible choice of pivot and the output from procedure $\mathcal{C}$, the evaluator h computes a possible value for each j-th bit of a_n. There are only a polynomial number of possible choices of a_n, because there are only a polynomial number of pivots. $\mathcal{B}$ verifies which choice is the correct accepting computation of M on 0^n, and outputs that value. Finally, we have only to note that the entire process can be carried out in $2^{n^{\varepsilon}}$ steps. This completes the proof of our Claim, and of the Theorem as well. ∎

Let Hypothesis H' be the following assertion:

Hypothesis H': There is an NP-machine M that accepts 0^* such that for some $0 < \varepsilon < 1$, no $2^{n^{\varepsilon}}$ time-bounded Turing machine correctly computes infinitely-many accepting computations of M.

Theorem 2 *If Hypothesis* H' *is true, then there exists a Turing complete language for* NP *that is not* $\leq_m^{\mathrm{P}}$*-complete for* NP.

Proof. The proof builds on the ideas of the previous argument, but here we will just give a brief outline of how the proof proceeds. For each $n \geq 0$, let a_n be the lexicographically maximum accepting computation of M on 0^n. Let $|a_n| = l_n$. Let $a = a_1 a_2 a_3 \cdots$. Define

$$L_1 = \{\langle x, u \rangle \mid |x| = n,\ u \text{ is an accepting computation of } M \text{ on } 0^m, n = m^{\varepsilon}/2, \text{ and } x \in \mathrm{SAT}\},$$

$$L_2 = L(a) = \{x \mid x < a\},$$

and $L = L_1 \oplus L_2$.

It is easy to see, as in the previous argument, that L is $\leq_{\mathrm{T}}^{\mathrm{P}}$-complete for NP. In order to prove that L is not $\leq_m^{\mathrm{P}}$-complete, we define the set

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

which belongs to NP, and assume there is a $\leq_m^{\mathrm{P}}$-reduction from S to L. Then, we show that this assumption leads to a contradiction of Hypothesis H'. ∎

The following results give fine separations of polynomial time reducibilities in NP from significantly weaker hypotheses. Moreover, they follow readily from results in the literature.

Theorem 3 *If there is a tally language in* UP − P*, then there exist two languages* L_1 *and* L_2 *in* NP *such that* $L_1 \leq_{tt}^{\mathrm{P}} L_2$*,* $L_2 \leq_{\mathrm{T}}^{\mathrm{P}} L_1$*, but* $L_1 \not\leq_{btt}^{\mathrm{P}} L_2$*.*[1]

Proof. Let L be a tally language in UP − P. Let R be the polynomial-time computable relation associated with the language L. Define

$$L_1 = \{\langle 0^n, y\rangle \mid \exists w, R(0^n, w) \text{ and } y \leq w\}$$

and

$$L_2 = \{\langle 0^n, i\rangle \mid \exists w, R(0^n, w) \text{ and } i\text{-th bit of } w \text{ is one}\}.$$

It is clear that L_1 is $\leq_{tt}^{\mathrm{P}}$-reducible to L_2. To see that L_2 is $\leq_{\mathrm{T}}^{\mathrm{P}}$-reducible to L_1, implement a binary search algorithm that accesses L_1 to determine the unique witness w such that $R(0^n, w)$, and then find the i-th bit.

Observe that L_2 is a sparse set. Ogihara and Watanabe [OW91] call L_1 the *left set* of L, and they and Homer and Longpre [HL94] proved for every L in NP that if the left set of L is $\leq_{btt}^{\mathrm{P}}$-reducible to a sparse set, then L is in P. Hence $L_1 \not\leq_{btt} L_2$. ∎

We now prove that Turing and truth-table reducibilities also differ in NP under the same hypothesis.

Theorem 4 *If there is a tally language in* UP − P*, then there exist two languages* L_1 *and* L_2 *in* NP *such that* $L_1 \leq_{\mathrm{T}}^{\mathrm{P}} L_2$ *but* $L_1 \not\leq_{tt}^{\mathrm{P}} L_2$*.*

Proof. Hemaspaandra et al. [HNOS96] proved that the hypothesis implies existence of a tally language L in UP − P such that L is not $\leq_{tt}^{\mathrm{P}}$-reducible to any p-selective set. In the same paper they also showed, given a tally language L in NP − P, how to obtain a p-selective set S such that L is $\leq_{\mathrm{T}}^{\mathrm{P}}$-reducible to S. Combing the two results we obtain the theorem. ∎

4 Analysis of the Hypotheses

This section contains a number of results that help us to understand the strength of Hypotheses H and H$'$.

4.1 Comparisons With Other Complexity-Theoretic Assertions

We begin with some equivalent formulations of these hypotheses, and then relate them to other complexity-theoretic assertions. The question of whether P contains a P-printable-immune set was studied by Allender and Rubinstein [AR88], and the equivalence of items 1 and 3 in the following

[1] The class of all languages that are $\leq_{\mathrm{T}}^{\mathrm{P}}$-equivalent to L_1 is a *noncollapsing* degree.

theorem is similar to results of Hemaspaandra, Rothe, and Wechsung [HRW97] and Fortnow, Pavan, and Selman [FPS99]. The second item is similar to the the characterization of Grollmann and Selman [GS88] of *one-one, one-way* functions with the addition of the attribute *almost-always one-way* of Fortnow, Pavan, and Selman.

Theorem 5 *The following statements are equivalent:*

1. *There is a language L in* P *that contains exactly one string of every length such that L is* P*-printable-immune and, for some $\varepsilon > 0$, L is not 2^{n^ε}-printable.*
2. *There exists a polynomial-bounded, one-one, function $f : 0^* \to \Sigma^*$, such that f is almost-everywhere not computable in polynomial time, for some $\varepsilon > 0$, f is not computable in time 2^{n^ε}, and the graph of f belongs to* P.
3. *Hypothesis H is true*

Theorem 6 *The following statements are equivalent*

1. *There is a language L in* P *that contains* at least *one string of every length such that, for some $\varepsilon > 0$, L is 2^{n^ε}-printable-immune.*
2. *There is polynomial-bounded, multivalued function $f : 0^* \to \Sigma^*$ such that every refinement of f is almost-everywhere not computable in 2^{n^ε}-time, and the graph of f belongs to* P.
3. *Hypothesis H' holds.*

Next we compare our hypotheses with the following complexity-theoretic assertions:

1. For some $\varepsilon > 0$, there is a P-bi-immune language L in UP$\cap$co-UP such that $L \cap 0^*$ is not in DTIME(2^{n^ε}).
2. For some $\varepsilon > 0$, there is language L in UP$\cap$co-UP such that L is not in DTIME(2^{n^ε}).
3. For some $\varepsilon > 0$, there is a 2^{n^ε}-bi-immune language in NP$\cap$co-NP.

Theorem 7 *Assertion 1 implies Hypothesis H and Hypothesis H implies Assertion 2.*

Proof. Let L be a language in UP$\cap$co-UP that satisfies Assertion 1. Define M to be the UP-machine that accepts 0^* as follows: On input 0^n, nondeterministically guess a string w. If w either witnesses that 0^n is in L or witnesses that 0^n is in $\overline{L}$, then accept 0^n. It is immediate that M satisfies the conditions of Hypothesis H.

To prove the second implication, let M a UP-machine that satisfies the conditions of Hypothesis H. Let a_n denote the unique accepting computation of M on 0^n and define

$$L = \{\langle 0^n, x\rangle \mid x \leq a_n\}.$$

It is clear that $L \in$ UP$\cap$co-UP. If $L \in$ DTIME(2^{n^ε}), then a binary search algorithm can correctly compute a_n, for every n, in time 2^{n^ε}. This would contradict Hypothesis H. Hence, $L \notin$ DTIME(2^{n^ε}). ∎

Corollary 1 *If, for some* $\varepsilon > 0$, $\mathrm{UP} \cap \text{co-UP}$ *has a* 2^{n^ε}*-bi-immune language, then* $\leq_{\mathrm{T}}^{\mathrm{P}}$*-completeness is different from* $\leq_{tt}^{\mathrm{P}}$*-completeness for* NP.

Theorem 8 *Assertion (3) implies Hypothesis* $H^{'}$.

Corollary 2 *If, for some* $\varepsilon > 0$, $\mathrm{NP} \cap \text{co-NP}$ *has a* 2^{n^ε}*-bi-immune language, then* $\leq_{\mathrm{T}}^{\mathrm{P}}$*-completeness is different from* $\leq_{m}^{\mathrm{P}}$*-completeness for* NP.

4.2 Comparisons with Genericity

The genericity hypothesis of Ambos-Spies and Bentzien [ASB97], which they used successfully to separate NP-completeness notions for the bounded-truth-table reducibilities, states that "NP contains an n^2-generic language". Our next result enables us to compare this with our hypotheses.

We say that a deterministic oracle Turing machine M is a *predictor* for a language L if for every input word x, M decides whether $x \in L$ with oracle $A|x$. L is *predictable in time* $t(n)$ if there is a $t(n)$ time-bounded predictor for L. We define a set L to be *almost-everywhere unpredictable in time* $t(n)$ if every predictor for L requires more than $t(n)$ time for all but finitely many x. This concept obviously implies $\mathrm{DTIME}(t(n))$-complex almost everywhere, but the converse does not hold:

Theorem 9 EXP *contains languages that are* $\mathrm{DTIME}(2^n)$*-complex but* not *almost-everywhere unpredictable in time* 2^n.

Now we state our characterization of $t(n)$-genericity.

Theorem 10 *A decidable language* L *is* $t(n)$*-generic if and only if it is almost-everywhere unpredictable in time* $t(2^n - 1)$.

Proof. Assume that L is not almost-everywhere unpredictable in time $t(2^n - 1)$, and let M be a predictor for L that for infinitely many strings x runs in time $t(2^n - 1)$. Define a condition C so that the characteristic sequence

$$(L|x)\overline{x} \in C \Leftrightarrow M \text{ with oracle } L|x \text{ runs in time } t(2^{|x|} - 1) \text{ on input } x,$$

where $\overline{x} = \neg(M \text{ accepts } x)$. Then, C is dense along L because M correctly predicts whether $x \in L$ for infinitely many x. It is easy to see that $C \in \mathrm{DTIME}(t(n))$. However, L is not $t(n)$-generic because we defined C so that L does not meet C.

Assume that L is not $t(n)$-generic, and let $C \in \mathrm{DTIME}(t(n))$ be a condition that is dense along L such that L does not meet C. Let T be a deterministic Turing machine that halts on all inputs and accepts L. Define a predictor M for L to behave as follows on input x with oracle $A|x$: If $(A|x)1 \in C$, then M rejects x, and if $(A|x)0 \in C$, then M accepts x. If neither holds, then M determines membership in L by simulating T on x. Since L does not meet C, M is a predictor for L. Since C is dense along L and L does not meet C, for infinitely many x, either $(A|x)1 \in C$ or $(A|x)0 \in C$, and in each of these cases, M runs for at most $t(2^{|x|} - 1)$ steps. ∎

Corollary 1 NP *contains an* n^2*-generic language if and only if* NP *contains a set that is almost-everywhere unpredictable in time* 2^{2n}.

By Theorem 8, Hypothesis H' holds if NP$\cap$co-NP contains a set that, for some $\varepsilon > 0$, is $2^{n^{\varepsilon}}$-bi-immune. So, Hypothesis H' requires bi-immunity, which is weaker than almost-everywhere unpredictability, and the time-bound is reduced from 2^{2n} to $2^{n^{\varepsilon}}$. On the other hand, we require the language to belong to NP$\cap$co-NP instead of NP. Similarly, when we consider Hypothesis H, we require the language to be P-bi-immune and not in DTIME$(2^{n^{\varepsilon}})$, whereas now we require the language to be in UP$\cap$co-UP. Moreover, the conclusion of Theorem 1 is not known to follow from the genericity hypothesis. At the same time, we note that the genericity hypothesis separates several bounded-truth-table completeness notions in NP that do not seem obtainable from our hypotheses.

4.3 Relativization

Before stating our next result, we should observe that the discrete logarithm problem is an interesting possible witness for Hypothesis H: The discrete logarithm problem belongs to UP$\cap$*co*-UP. Researchers believe that it is hard on most instances, and the best known deterministic algorithm requires time greater than $2^{n^{\frac{1}{3}}}$ [Gor93]. Thus, for any $0 < \varepsilon \leq \frac{1}{3}$, the discrete logarithm problem is a candidate witness for the noninclusion UP$\cap$*co*-UP $\not\subseteq$ DTIME$(2^{n^{\varepsilon}})$.

Theorem 11 *There exists an oracle relative to which the polynomial hierarchy is infinite and Hypotheses H and* H' *both hold.*

Proof. Define Kolmogorov random strings $r_0, r_1, \ldots$ as follows: r_n is the first string of length n such that

$$K^{2^n}(r_n \mid r_0, r_1, \ldots, r_{n-1}) > n/2.$$

Then, define the oracle $A = \{r_n \mid n \geq 0\}$. The full proof shows that the hypotheses hold relative to A. Then, because A is a sparse set, using results of Balcazar *et al.* [BBS86] we can show that there is an oracle relative to which the hypotheses holds and the polynomial hierarchy is infinite. ∎

Hypothesis H fails relative to any oracle for which P $=$ NP$\cap$co-NP [BGS75]. Fortnow and Rogers [FR94] obtained an oracle relative to which NP $\neq$ co-NP and Hypothesis H' fails. We know of no oracle relative to which P $\neq$ NP and every $\leq^{\mathrm{P}}_{\mathrm{T}}$-complete set is $\leq^{\mathrm{P}}_{m}$-complete.

References

[AR88] E. Allender and R. Rubinstein. P-printable sets. *SIAM Journal on Computing*, 17(6):1193–1202, 1998.

[ASB97] K. Ambos-Spies and L. Bentzien. Separating NP-completeness under strong hypotheses. In *Proceedings of the 12th annual IEEE Conference on Computational Complexity*, pages 121–127, 1997.

[ASFH87] K. Ambos-Spies, H. Fleischhack, and H. Huwig. Diagonalizations over polynomial time computable sets. *Theoretical Computer Science*, 51:177–204, 1987.

[ASNT96] K. Ambos-Spies, H. Neis, and A. Terwijn. Genericity and measure for exponential time. *Theoretical Computer Science*, 168(1):3–19, 1996.

[ASTZ97] K. Ambos-Spies, A. Terwijn, and X. Zheng. Resource bounded randomness and weakly complete problems. *Theoretical Computer Science*, 172(1):195–207, 1997.

[BGS75] T. Baker, J. Gill, and R. Solovay. Relativizations of the P =? NP Question. *SIAM Journal on Computing*, 4(4):431–441, 1975.

[BBS86] J. Balcázar, R. Book, and U. Schöning. The polynomial-time hierarchy and sparse oracles. *Journal of the ACM*, 33(3):603–617, 1986.

[BS85] J. Balcázar and U. Schöning. Bi-immune sets for complexity classes. *Mathematical Systems Theory*, 18(1):1–18, June 1985.

[BHT91] H. Buhrman, S. Homer, and L. Torenvliet. Completeness notions for nondeterministic complexity classes. *Mathematical Systems Theory*, 24:179–200, 1991.

[FFNR96] S. Fenner, L. Fortnow, A. Naik, and J. Rogers. On inverting onto functions. In *Proceedings of the 11th annual IEEE Conference on Computational Complexity*, pages 213–223, 1996.

[FPS99] L. Fortnow, A. Pavan, and A. Selman. Distributionally-hard languages. In *COCOON 99, Lecture Notes in Computer Science v. 1627*, pages 184–193. Springer-Verlag, 1999.

[FR94] L Fortnow and J. Rogers. Separability and one-way functions. In D.Z. Du and X.S. Zhang, editors, *Proceedings of the Fifth International Symposium on Algorithms and Computation*, Lecture Notes in Computer Science, pages 396–404. Springer-Verlag, 1994.

[Gor93] D. Gordon. Discrete logarithms in GF(p) using the number field sieve. *SIAM Journal on Discrete Mathematics*, 6:124–138, 1993.

[GS88] J. Grollmann and A. Selman. Complexity measures for public-key cryptosystems. *SIAM Journal on Computing*, 17(2):309–335, 1988.

[HNOS96] E. Hemaspaandra, A. Naik, M. Ogiwara, and A. Selman. P-selective sets and reducing search to decision vs. self-reducibility. *J. of Computer and System Sciences*, 53(2):194–209, 1996. Special Issue of papers selected from the Eighth Annual IEEE Conference on Structure in Complexity Theory. In press.

[HRW97] L. Hemaspaandra, J. Rothe, and G. Wechsung. Easy sets and hard certificate schemes. *Acta Informatica*, 34(11):859–879, 1997.

[HY84] J. Hartmanis and Y. Yesha. Computation times of NP sets of different densities. *Theoretical Computer Science*, 34:17–32, 1984.

[KR93] S. Homer, S. Kurtz, and J. Royer. On 1-truth-table-hard languages. *Theoretical Computer Science*, 115(2):383–389, 1993.

[HL94] S. Homer and L. Longpré. On reductions of NP sets to sparse sets. *Journal of Computer and Systems Sciences*, 48(2):324–336, 1994.

[KM81] K. Ko and D. Moore. Completeness, approximation and density. *SIAM Journal on Computing*, 10(4):787–796, Nov. 1981.

[LLS75] R. Ladner, N. Lynch, and A. Selman. A comparison of polynomial time reducibilities. *Theoretical Computer Science*, 1:103–123, 1975.

[LV97] M. Li and P. Vitányi. *An Introduction to Kolmogorov Complexity and Its Applications*. Graduate Texts in Computer Science. Springer, New York, second edition, 1997.

[LY90] L. Longpré and P. Young. Cook reducibility is faster than Karp reducibility. *Journal of Computer and System Sciences*, 41:389–401, 1990.

[LM96] J. Lutz and E. Mayordomo. Cook versus karp-levin: Separating completeness notions if NP is not small. *Theoretical Computer Science*, 164:141–163, 1996.

[OW91] M. Ogiwara and O. Watanabe. On polynomial time bounded truth-table reducibility of nP sets to sparse sets. *SIAM Journal on Computing*, 20(3):471–483, 1991.

[Sel79] A. Selman. P-selective sets, tally languages, and the behavior of polynomial time reducibilities on NP. *Mathematical Systems Theory*, 13:55–65, 1979.

[Sel82] A. Selman. Reductions on NP and P-selective sets. *Theoretical Computer Science*, 19:287–304, 1982.

[Tod91] S. Toda. On polynomial-time truth-table reducibilities of intractable sets to P-selective sets. *Mathematical Systems Theory*, 24(2):69–82, 1991.

[Wat87] O. Watanabe. A comparison of polynomial time completeness notions. *Theoretical Computer Science*, 54:249–265, 1987.

Appendix

Here we give proofs of key results that are not provided in the main body.

Hypothesis $\mathrm{H}^{'}$**:** There is an NP-machine M that accepts 0^* such that for some $0 < \varepsilon < 1$, no 2^{n^ε} time-bounded Turing machine correctly computes infinitely-many accepting computations of M.

Theorem 2 *If Hypothesis* $\mathrm{H}^{'}$ *is true, then there exists a Turing complete language for* NP *that is not* $\leq_m^{\mathrm{P}}$*-complete for* NP.

Proof. Let M be an NP-machine that satisfies the conditions of Hypothesis $\mathrm{H}^{'}$. For each $n \geq 0$, let a_n be the lexicographically maximum accepting computation of M on 0^n, and let $|a_n| = l_n$. Define the language

$$L_1 = \{\langle x, u\rangle \mid |x| = n,\ u \text{ is an accepting computation of } M \text{ on } 0^m, n = m^\varepsilon/2, \text{ and } x \in \mathrm{SAT}\}.$$

Let $a = a_1 a_2 a_3 \cdots$, and define

$$L_2 = L(a) = \{x \mid x < a\}.$$

Define $L = L_1 \oplus L_2$.

It is easy to see, as in the previous argument, that L is $\leq_{\mathrm{T}}^{\mathrm{P}}$-complete for NP. In order to prove that L is not $\leq_m^{\mathrm{P}}$-complete, we define the set

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

which belongs to NP, and assume there is a $\leq_m^{\mathrm{P}}$-reduction f from S to L. Consider the following procedure $\mathcal{D}$:

input 0^n;
$y := \lambda$;
Repeat l_n times
    begin
    $f(\langle 0^n, y0\rangle) := x_0$;
    $f(\langle 0^n, y1\rangle) := x_1$;
    if both x_0 and x_1 are queries to L_2
        then if $x_0 \leq x_1$
            then $y := y0$
            else $y := y1$
        else {At least one of x_0 and x_1 is a query to L_1; let $b \in \{0,1\}$ be the least index
            such that x_b queries L_1, and let $x_b = \langle z, u\rangle$ }
            if u is not an accepting computation of M {thus, $x_b \notin L_1$}
                then $y = y\bar{b}$
                else {let u is an accepting computation of M on 0^t}
                    if $t \geq n^\varepsilon$
                        then output "Unsuccessful," print u, and terminate
                        else $\{t < n^\varepsilon\}$
                            if $|z| = t^\varepsilon/2$ and $z \in$ SAT ({thus, $x_b \in L_1$})
                                then $y := yb$
                                else $\{x_b \notin L_1\}$ $y := y\bar{b}$
    end;
output "Successful" and print y.

First we will analyze the running time and then we treat two cases, namely, either $\mathcal{D}(0^n)$ is successful for infinitely many n, or it is unsuccessful for all but finitely many n.

Claim 3 *The above procedure halts in* $O(l_n 2^{n^{\varepsilon^2}/2})$ *steps.*

Proof. Consider an iteration of the repeat loop. The most expensive step is the test of whether "$z \in$ SAT". This test occurs only when $|z| = t^\varepsilon/2$ and $t < n^\varepsilon$. Hence we can decide whether z belongs to SAT in $2^{n^{\varepsilon^2}/2}$ steps. All other steps take polynomial time. Hence the time taken by the procedure is $O(l_n 2^{n^{\varepsilon^2}/2})$. ∎

Since $0 < \varepsilon < 1$, the running time of procedure $\mathcal{D}$ is bounded by 2^{n^ε}.

Claim 4 *If $\mathcal{D}(0^n)$ is successful for infinitely many n, then there is a 2^{n^ε}-time-bounded Turing machine that correctly computes infinitely many accepting computations of M.*

Proof. We demonstrate that if $\mathcal{D}$ is successful on an input 0^n, then the string that is printed is an accepting computation of M on 0^n. In order to accomplish this, we prove by induction that y is a prefix of an accepting computation of M on 0^n during every iteration of the repeat loop (i.e., a loop invariant). Initially when $y = \lambda$ this is true. Assume that y is a prefix of an accepting computation of M at the beginning of an iteration. Then, at least one of $f(\langle 0^n, y0\rangle) = x_0$, $f(\langle 0^n, y1\rangle) = x_1$ must belong to L. If both x_0 and x_1 are queries to L_2, then the smaller of x_0 and x_1 belongs to L_2 because L_2 is p-selective. Thus, in this case, the procedure extends y correctly. If at least one of x_0 and x_1 is a query to L_1, then the procedure determines whether $x_b \in L_1$, where x_b is the query to L_1 with least index. If x_b belongs to L, then $\langle 0^n, yb\rangle \in S$. Hence, yb is a prefix of an accepting computation. If $x_b \notin L$, then $x_{\bar{b}}$ belongs to L, because at least one of x_b or $x_{\bar{b}}$ belongs to L. Thus, in this case, $y\bar{b}$ is a prefix of an accepting computation. This completes the induction argument.

The loop repeats l_n times. Therefore, the final value of y, which is the string that $\mathcal{D}$ prints, is an accepting computation. ∎

Claim 5 *If $\mathcal{D}(0^n)$ is unsuccessful for all but finitely many n, then there is a 2^{n^ε}-time-bounded Turing machine that correctly computes infinitely many accepting computations of M.*

Proof. The proof is similar to the proof of Claim 1. The following procedure computes infinitely many accepting computations of M.

input 0^n;
for $j = 1$ to $j = n^{1/\varepsilon}$ **do**
    if $\mathcal{D}(0^j)$ outputs u and u is an accepting computation of M on 0^n
        then print u and terminate.

The running time of this algorithm can be bounded as follows: The procedure $\mathcal{D}(0^j)$ runs in time $l_j 2^{j^{\varepsilon^2}/2}$ steps. So the total running time is $\sum_1^{n^{1/\varepsilon}} l_j 2^{j^{\varepsilon^2}/2} = O(2^{n^\varepsilon})$. ∎

Since the cases treated both by Claims 4 and 5 demonstrate Turing machines that correctly compute infinitely many accepting computations of M in 2^{n^ε} time, we have a contradiction to Hypothesis H'. Thus L is not $\leq_m^{\mathrm{P}}$-complete for NP. ∎

Theorem 5 *The following statements are equivalent:*

1. *There is a language L in* P *that contains exactly one string of every length such that L is* P*-printable-immune and, for some $\varepsilon > 0$, L is not 2^{n^ε}-printable.*
2. *There exists a polynomial-bounded, one-one, function $f : 0^* \to \Sigma^*$, such that f is almost-everywhere not computable in polynomial time, for some $\varepsilon > 0$, f is not computable in time 2^{n^ε}, and the graph of f belongs to* P.
3. *Hypothesis H is true*

Proof. Let L satisfy item one. Define $f(0^n) =$ the unique string of length n that belongs to L. Clearly, f us polynomial-bounded and one-one. The graph of f belongs to P, because L belongs to P. Suppose that M is a Turing machine that computes f and that runs in polynomial time on infinitely many inputs. Then, on these inputs, M prints $L \cap \Sigma^n$. Similarly, f is not computable in time 2^{n^ε}.

Let f satisfy item two. Define a UP-machine M to accept 0^* as follows: On input 0^n, M guesses a string y of length within the polynomial-bound of f, and accepts if and only if $\langle 0^n, y\rangle \in graph(f)$. The rest of the proof is clear.

Let M be a UP-machine that satisfies item three, i.e., that satisfies the conditions of Hypothesis H. Let a_n be the unique accepting computation of M on 0^n and let $|a_n| = n^l$. Let r_n be the rank of a_n among all strings of length n^l. Now, we define L as follows: Given a string x, if $|x| = n^l$ for some n, then x belongs to L if and only if $x = a_n$. If $(n-1)^l < |x| < n^l$, then x belongs to L if and only if the rank of x (among all the string of length $|x|$) is r_{n-1}. It is clear that $L \in$ P and has exactly one string per each length. Again, any machine that prints infinitely many strings of L can be used to print infinitely many accepting computations of M. ∎

Theorem 11 *There exists an oracle relative to which the polynomial hierarchy is infinite and Hypotheses H and* $\mathrm{H}^{'}$ *both hold.*

Proof. Define Kolmogorov random strings $r_0, r_1, \ldots$ as follows: r_n is the first string of length n such that
$$K^{2^n}(r_n \mid r_0, r_1, \ldots, r_{n-1}) > n/2.$$
Then, define the oracle $A = \{r_n \mid n \geq 0\}$.

Define M to be an oracle Turing machine that accept 0^* with oracle A as follows: On input 0^n, guess a string y of length n. If $y \in A$, then accept. M is a UP^A-machine that accepts 0^* because A contains exactly one string of every length.

Now we show that no 2^{n^ε} oracle Turing machine with oracle A, for any $0 < \varepsilon < 1$, correctly computes infinitely many accepting computations of M. Observe that relative to A, this implies both Hypotheses H and $\mathrm{H}^{'}$. Suppose otherwise, and let T be such an oracle Turing machine. The gist of the remainder of the proof is that we will show how to simulate T without using the oracle, and that will contradict the randomness or r_n.

Suppose that $T^A(0^n) = r_n$. Let $l = 3n^\varepsilon$. Then we simulate this computation without using an oracle as follows:

1. Compute $r_0, r_1, \ldots, r_{l-1}$. Do this iteratively: Compute r_i by running every program (with input strings $r_0, r_1, \ldots, r_{i-1}$) of length $\leq i/2$ for 2^i steps. Then r_i is the first string of length i that is not output by any of these programs. Note that the total time for executing this step is
$$l 2^{l/2} 2^l \leq l 2^{3l/2} \leq 2^{5n^\varepsilon}.$$

2. Simulate T on input 0^n, except replace all oracle queries q by the following rules: If $|q| < l$, answer using the previous computations. Otherwise, just answer "no."

If the simulation is correct, then this procedure outputs r_n without using the oracle. The running time of this procedure on input 0^n is $2^{5n^\varepsilon} + 2^{n^\varepsilon}$, which is less than 2^n. So, we can describe r_n by a string of length $O(\log n)$, to wit, a description of T and 0^n. This contradicts the definition of r_n.

We need to show that the simulation is correct. The simulation can only be incorrect if $|q| \geq l$ and $q = r_m$, for some $m > l$. Let r_m be the first such query. This yields a short description of r_m, given $r_0, r_1, \ldots, r_{l-1}$. Namely, the description consists of the description of T (a constant), the description of 0^n ($\log n$ bits), and the description of the number j such that $q = r_m$ is the j-th query (at most n^ε). Thus, the length of the description is $O(n^\varepsilon)$. Since $l = 3n^\varepsilon$, it follows that the length of the description of r_m is less than $m/2$. The running time of T, given $r_0, r_1, \ldots, r_{l-1}$, is 2^{n^ε}, which is less than 2^m. (The reason is that the first step in the simulation of T is not needed.) Therefore, the simulation is correct.

Finally, because A is a sparse set, using results of Balcazar *et al.* [BBS86], there is an oracle relative to which the hypotheses holds and the polynomial hierarchy is infinite. ∎