

Inverting the Turing Jump in Complexity Theory

Stephen A. Fenner*
Computer Science Department
University of Southern Maine

Abstract

This paper investigates the invertibility of certain analogs of the Turing jump operator in the polynomial-time Turing degrees. If C is some complexity class, the C -jump of a set A is the canonical C -complete set relative to A . It is shown that the PSPACE-jump and EXP-jump operators are not invertible, i.e., there is a PSPACE-hard (resp. EXP-hard) set that is not p-time Turing equivalent to the PSPACE-jump (resp. EXP-jump) of any set. It is also shown that if PH collapses to Σ_k^p , then the Σ_k^p -jump is not invertible. In particular, if $\text{NP} = \text{co-NP}$, then the NP-jump is not invertible, witnessed, in particular, by $G \oplus \text{SAT}$, where G is any 1-generic set. These results run contrary to the Friedberg Completeness Criterion [Fri57] in recursion theory, which says that every (recursive) Turing degree above $\mathbf{0}'$ is the Turing jump of another degree. The sets used in the paper to witness C -jump noninvertibility are all of the form $G \oplus C$, where G is 1-generic and C is some C -complete set. Other facts regarding 1-generics G and $G \oplus \text{SAT}$ are also explored in this paper; in particular, G always lies in $\text{NP}^A - \text{P}^A$ for some $A \leq_{\text{tt}}^p G$, but if $A \leq_{\oplus\text{-tt}}^p G \oplus \text{SAT} \leq_{\text{T}}^p \text{SAT}^A$ for $A \oplus \text{SAT}$, which in turn implies

$\leq_{\text{T}}^p A$ or $\text{P} \neq \text{NP}$.

Introduction

Insight has been gained about the structure of \leq_{T}^p -bounded reductions and degrees by comparison with their recursion theoretic analogs. Some of these structures are similar; for example, Ladner [Lad75] showed the polynomial-time (ptime) version of Post's problem: if $\text{P} \neq \text{NP}$ then there is an intermediate degree which is neither in P nor NP -complete under p-time Turing reductions. Frequently, the structures of complexity theory are opposite—Ladner's results also show that there are no minimal ptime Turing degrees, in contrast to the recursion theoretic case. The most unfortunate consequence, however, is also the most common: the

polynomial-time analogs of known recursion theoretic results remain unresolved. The halting problem is known to be undecidable, but is $\text{SAT} \notin \text{P}$? A result of Myhill [Myh55] says that all m -complete r.e. sets are recursively isomorphic, but its polynomial-time version, the Isomorphism Conjecture, is a famous unresolved conjecture in complexity theory [BH77]. The list goes on.

The Friedberg Completeness Criterion in recursion theory [Fri57] states that the range of the Turing jump operator on the Turing degrees is as large as possible: $\{\mathbf{a} \mid \mathbf{0}' \leq \mathbf{a}\}$. In other words, a degree is jump-invertible if and only if it lies above $\mathbf{0}'$. We address natural complexity theoretic analogs of this statement for complexity classes C . Namely,

Statement 1 For any set B ptime Turing hard for C , there is a set A such that $B \equiv_{\text{T}}^p C^A$, where C^A is some canonical C^A -complete set under (unrelativized) ptime Turing reductions.

If we call C^A the C -jump of A , then Statement 1 can be rephrased as, "a ptime Turing degree has a C -jump inverse if and only if it is C -hard," which is a precise ptime analog of Friedberg's result about the (unbounded) Turing degrees. We show that Statement 1 is false for $C = \text{PSPACE}, \text{EXP}$, and other larger classes, and in the event that the polynomial hierarchy collapses to Σ_k^p , the statement is false for $C = \Sigma_k^p$ as well. In particular, if $\text{NP} = \text{co-NP}$, then the NP-jump is not invertible. Our results thus run contrary to the recursion theoretic setting, at least under strong assumptions. The question of whether the NP-jump is invertible without assumptions (or with weaker assumptions) remains open at present, and we currently know of no nontrivial class C for which Statement 1 holds.

It is worth noting that the Turing jump has been successfully inverted in at least one restricted recursion theoretic setting. Mohrherr [Moh84] showed

*Partially supported by NSF Grant CCR 92-09833. Email: fenner@cs.usm.maine.edu

that every tt-degree above K contains a jump. Her proof takes the original set constructed for the Friedberg jump inversion and shows that its jump is tt-equivalent to the target set.

We use 1-generic sets (see Section 2; also see Jockusch or Soare [Joc80, Soa87]), which greatly simplify our task of addressing these questions. In Proposition 5, we will show that if *any* \mathcal{C} -hard degree fails to have a \mathcal{C} -jump inverse, then the degree of $G \oplus C$ fails to have one also, where G is any 1-generic set and C is complete for \mathcal{C} [For93]. Observing this fact allows one to concentrate exclusively on studying the behavior of $G \oplus C$ to decide Statement 1.

We give preliminary definitions in Section 2 and prove our main results in Section 3. In Section 4, we explore further the behavior of 1-generic sets, determining that every 1-generic set lies strictly between A and SAT^A for some A .¹ In fact, for all 1-generic G , there is an A such that

$$A \leq_{tt}^p G \leq_m^p \text{SAT}^A, \quad (1)$$

but there are no ptime Turing reductions going in the opposite direction, so in particular, $G \in \text{NP}^A - \text{P}^A$. This result *does* bear a resemblance to its recursion theoretic analog, [Joc80, Theorem 5.1], which states that every 1-generic set G is r.e. in some $A <_T G$. We then show that the ptime tt-reduction mentioned in (1) cannot be replaced by a ptime m -reduction or even a ptime parity tt-reduction without assuming $\text{P} \neq \text{NP}$ unrelativized. The proof of the latter result relies on the *linearity* of parity tt-reductions.

Finally in Section 5, we examine the related question of NP-hardness versus relative NP-completeness. We also suggest extensions to this work and pose some open problems.

2 Preliminaries

We assume the reader is familiar with the basic concepts of computability and complexity theory, which can be found in a number of standard texts, including [Soa87, HU79]. We use ω to denote the natural numbers, and use x, y, z, \dots for variables ranging over ω . We use $\alpha, \beta, \gamma, \sigma, \tau, u, v, w$ for variables ranging over finite characteristic functions, i.e., functions $D \rightarrow \{0, 1\}$ where $D \subseteq \omega$ is finite. We call these functions (*binary*) *strings* if D is an initial segment

of ω . We identify strings with natural numbers via the usual binary representation. For $x \in \omega$, we let $|x|$ denote the length of x as a binary string. We use A, B, C, \dots for subsets of ω , and often identify them with their characteristic functions. For $A, B \subseteq \omega$, $A \oplus B \stackrel{\text{df}}{=} \{2x \mid x \in A\} \cup \{2x + 1 \mid x \in B\}$ denotes the join of A and B , $A \Delta B$ denotes the symmetric difference of A and B , and \bar{A} denotes $\omega - A$. If σ is a string, then $\sigma \oplus B$ denotes the partial characteristic function f such that for all x , $f(2x + 1) = B(x)$, $f(2x) = \sigma(x)$ if $\sigma(x)$ is defined, and $f(2x)$ is undefined otherwise. We fix a bijection $\langle \cdot, \cdot \rangle: \omega \times \omega \rightarrow \omega$ to be the standard ptime computable, ptime invertible pairing function [Rog67].

If f and g are partial characteristic functions, we say $f \preceq g$ to mean f is extended by g , and $f \prec g$ to mean f is properly extended by g . The domain of f is denoted by $\text{domain}(f)$, and $f(x) \downarrow$ means $x \in \text{domain}(f)$. If $D \subseteq \text{domain}(f)$ is finite, then $f \upharpoonright D$ denotes the unique partial characteristic function extended by f whose domain is D . Let S be a set of binary strings and $A \subseteq \omega$. A *meets* S if there is a $\sigma \prec A$ with $\sigma \in S$. A *strongly avoids* S if there is a $\sigma \prec A$ such that for all $\tau \succeq \sigma$, $\tau \notin S$. A set $G \subseteq \omega$ is *1-generic* if G either meets or strongly avoids every r.e. set of strings. For more on 1-generic and generic sets and their uses in recursion and complexity theory, see [Joc80, Soa87, BI87, FFKL93].

If M is a time-bounded accepting/rejecting oracle Turing machine (OTM) and f a partial characteristic function, then $M^f(x)$ is defined just in case all of M 's oracle queries on input x are in $\text{domain}(f)$. M^f then denotes the partial characteristic function computed by M with oracle f . The one exception to this rule is when M is an NP machine; then, $M^f(x)$ will also be regarded as defined if there is some accepting path of $M^f(x)$, all of whose oracle queries are in $\text{domain}(f)$ and answered according to f , even if queries made on other paths are not in $\text{domain}(f)$. For a set A , we use M^A to denote both the machine M using oracle A and the set computed by M with oracle A . It should be clear from the context which meaning we intend. If M 's oracle is the explicit join of two sets A and B with B recursive, then we often separate the two oracles, and view M as explicitly using the oracle A only (M^A), but having “private” access to the oracle B ; that is, M is a Turing reduction to A that uses B internally.

This paper is concerned primarily with questions about the structure of the ptime Turing degrees. Thus all our notions of reducibility, equivalence, degrees, completeness, and hardness will be with respect to

¹ SAT^A is defined by Goldsmith and Joseph [GJ86]. Any other NP^A -complete set under unrelativized ptime Turing reductions can be substituted for SAT^A in the statement above, for example, the set $K(A)$ [BDG88]. We use SAT^A here mainly to avoid confusion with the halting problem, which is denoted by K .

unrelativized ptime Turing reductions unless explicitly stated otherwise. If \mathcal{C} is some relativizable complexity class that is “reasonable,” i.e., $A \leq_T^p B$ implies $P^A \subseteq C^A \subseteq C^B$ for all A and B , then we say that \mathcal{C} admits a jump if there is some OTM M such that M^A is C^A -complete for all A (remember that completeness is with regard to *unrelativized* ptime Turing reductions). We will denote this complete set by C^A , and will write C^\emptyset simply as C . Most of the usual complexity classes containing P admit jumps: P , NP , Σ_2^p , Π_2^p , \dots , $PSPACE$, EXP , $NEXP$, \dots . If \mathcal{C} admits a jump, then the operator that takes a set A to C^A we will call the \mathcal{C} -jump. By the “reasonableness” of \mathcal{C} , the \mathcal{C} -jump clearly preserves ptime Turing equivalence, so it can be thought of as an operator on the ptime Turing degrees, independent of the machine used to compute C^A . From now on, when we use \mathcal{C} we will assume that it admits a jump.

We say that the \mathcal{C} -jump is *invertible* if every \mathcal{C} -hard degree is the \mathcal{C} -jump of some degree. In other words, for every \mathcal{C} -hard set B , there is an A such that $P^B = P^{C^A}$.

If $\mathcal{C} = NP$ for example, we will take C^A to be SAT^A , the relativized version of SAT defined by Goldsmith and Joseph [GJ86]. We will identify SAT^A with some NP oracle machine that computes it. For every A , SAT^A is complete for NP^A under unrelativized ptime m -reductions. Moreover, the queries made by $SAT^A(x)$ are strings of length at most $|x|$.

Finally, for $S \subseteq \omega$ and $n \in \omega$ we use $S^{<n}$ to denote the set $\{x \in S \mid |x| < n\}$. If $\sigma(x)$ is defined for all x with $|x| < n$, then $\sigma^{<n}$ denotes the set $\{x \mid |x| < n \ \& \ \sigma(x) = 1\}$.

3 Main Results

We first show the following (rather easy) fact:

Theorem 2 *The PSPACE-jump is not invertible.*

Proof: Let B be any set. If there is a set A such that $PSPACE^A = P^B$, then we have

$$\begin{aligned} PSPACE^B &= PSPACE^{PSPACE^A} \\ &= PSPACE^A = P^B. \end{aligned}$$

It is routine to construct a PSPACE-hard set B such that $P^B \neq PSPACE^B$, which proves the theorem. \square

Stephan [Ste95] has recently extended this result to show that the EXP-jump is also not invertible, and likewise for any reasonable deterministic class above EXP.

Theorem 3 (Stephan) *If \mathcal{C} is a class and C^A is closed under the NP-jump for all A , then the \mathcal{C} -jump is not invertible.*

Proof: A \mathcal{C} -hard set B can always be constructed such that $P^B \neq NP^B$ (in particular, the set $G \oplus C$ described below suffices, where G is 1-generic and C is \mathcal{C} -complete). Suppose $B \equiv_T^p C^A$ for some A . Then, since $SAT^{C^A} \in C^A$, and C^A is C^A -complete, we have

$$SAT^B \equiv_T^p SAT^{C^A} \leq_T^p C^A \equiv_T^p B,$$

and thus $P^B = NP^B$, which contradicts our choice of B . \square

Corollary 4 (Stephan) *The EXP-jump is not invertible.*

We turn our attention in the rest of this section to the question of jump invertibility for classes, some of whose relativizations are *not* closed under the NP-jump, i.e., classes that fail to satisfy the hypothesis of Theorem 3 above. Here, the situation is much more subtle; the proofs are more difficult and are based upon unproven and unlikely assumptions. Still, the results below provide strong evidence for the noninvertibility of these jumps, and the techniques used in the proof of Theorem 7 have more general applicability, which we explore in the next section.

In the proof of Theorem 3, the set witnessing \mathcal{C} -jump noninvertibility was taken to be $G \oplus C$, where G is any 1-generic set and C is \mathcal{C} -complete. Actually, this choice works in general. When considering the question of \mathcal{C} -jump inversion for an arbitrary class \mathcal{C} , we need *only* study the set $G \oplus C$. This is by virtue of the following proposition [For93], whose proof may be skipped:

Proposition 5 (Fenner, Fortnow) *Let \mathcal{C} be a complexity class that admits a jump, and let G be any 1-generic set. If the \mathcal{C} -jump is not invertible (i.e., there exists a \mathcal{C} -hard set B that is not equivalent to the \mathcal{C} -jump of any set), then $G \oplus C$ is not equivalent to the \mathcal{C} -jump of any set.*

Proof: The idea is that we can diagonalize against any equivalence between $G \oplus C$ and C^A by finite extension. Let B be as in the hypothesis. Since $B \oplus C \equiv_T^p B$, $B \oplus C$ also satisfies the hypothesis, and so does $D \oplus C$ for any D such that $D \triangle B$ is finite. Let M_1 , M_2 , and M_3 be any three ptime deterministic OTMs, and for all sets D let $L_1(D)$, $L_2(D)$, and $L_3(D)$ denote M_1^D ,

M_2^D , and M_3^D , respectively. By hypothesis, for any set D which is a finite variant of B , we have

$$D \oplus C \not\equiv_T^{p} C^{L_1(D \oplus C)},$$

and thus

$$\begin{aligned} C^{L_1(D \oplus C)} &\neq L_2(D \oplus C) \\ \text{or} \\ D \oplus C &\neq L_3(C^{L_1(D \oplus C)}). \end{aligned} \quad (2)$$

This inequality is forced by only a finite amount of D ; that is, there exists a string $\tau \prec D$ such that then

$$(\exists x) \left[\begin{array}{c} C^{L_1(\tau \oplus C)}(x) \downarrow \neq L_2(\tau \oplus C)(x) \downarrow \\ \text{or} \\ (\tau \oplus C)(x) \downarrow \neq L_3(C^{L_1(\tau \oplus C)})(x) \downarrow \end{array} \right].$$

Let σ be an arbitrary string. In (2), if we let $D \stackrel{\text{df}}{=} \sigma$ except that $D(x) = \sigma(x)$ for all $x \in \text{domain}(\sigma)$, then there is a $\tau \succeq \sigma$ satisfying (3). Thus the set

$$S \stackrel{\text{df}}{=} \{\tau \mid \tau \text{ satisfies (3)}\}$$

is dense. S is also r.e., so G meets S . Therefore

$$\begin{aligned} C^{L_1(G \oplus C)} &\neq L_2(G \oplus C) \\ \text{or} \\ G \oplus C &\neq L_3(C^{L_1(G \oplus C)}). \end{aligned}$$

Since the ptime OTMs M_1 , M_2 , and M_3 were chosen arbitrarily, we have

$$G \oplus C \not\equiv_T^{p} C^{L_1(G \oplus C)}$$

for any M_1 . For any A , if $C^A \equiv_T^{p} G \oplus C$, then we must have $A \leq_T^{p} G \oplus C$, and this is explicitly ruled out by (5). Thus there is no A such that $C^A \equiv_T^{p} G \oplus C$. \square

The next lemma is crucial to most of our later results, especially Theorems 7 and 12. It says essentially that any self-reduction of a 1-generic set must also self-reduce all sets extending some finite string.

Lemma 6 *Suppose G is a 1-generic set and R is some time-bounded OTM. If $G = R^G$, then there is a $\sigma \prec G$ such that $B = R^B$ for all $B \succ \sigma$.*

Proof: The set of strings

$$S \stackrel{\text{df}}{=} \{\tau \mid \exists x(\tau(x) \downarrow \neq R^\tau(x) \downarrow)\}$$

is r.e. and hence is either met or strongly avoided by G . By hypothesis, G does not meet S , so the conclusion follows. \square

We now prove our main results, which are all corollaries of the next theorem. We say that a class \mathcal{C} is *polynomial-use bounded* if there is a polynomial p such that for all oracles A and inputs x , $C^A(x)$ uses A only on queries of length at most $p(|x|)$. PSPACE and Σ_k^p for $k \geq 0$ are all polynomial-use bounded.

Theorem 7 *Suppose \mathcal{C} is polynomial-use bounded and G is 1-generic. For any set A , if*

$$A \not\equiv_T^{p} G \oplus C$$

$$\text{SAT}^{A \oplus C} \not\leq_T^{p} G \oplus C.$$

then the next corollary leads directly to our results about specific jumps. It pertains to classes that display a particular kind of weak NP-robustness.

Corollary 8 *If \mathcal{C} is polynomial-use bounded and $\text{SAT}^{A \oplus C} \leq_T^{p} C^A$ for all A , then the \mathcal{C} -jump is not invertible.*

Proof: Suppose the \mathcal{C} -jump is invertible. For any 1-generic G , let A be such that $G \oplus C \equiv_T^{p} C^A$. The hypotheses of Theorem 7 hold, so we get $\text{SAT}^{A \oplus C} \not\leq_T^{p} G \oplus C \equiv_T^{p} C^A$ as a contradiction. The \mathcal{C} -hard set $G \oplus C$ is therefore not equivalent to the \mathcal{C} -jump of any set. \square

Note that PSPACE clearly satisfies the hypotheses of Corollary 8, so we obtain an alternate (albeit much more complicated) proof of Theorem 2 this way.

Corollary 9 *For any $k > 0$, if $\Sigma_k^p = \Pi_k^p$, then the Σ_k^p -jump is not invertible. In particular, if $\text{NP} = \text{co-NP}$, then the NP-jump is not invertible.*

Proof: Assume the hypothesis and let C be a Σ_k^p -complete set. Both C and \overline{C} are recognized respectively by NP oracle machines M and N with an oracle

$D \in \Sigma_{k-1}^p$. For any A , we can compute $\text{SAT}^{A \oplus C}$ on input x by an $\text{NP}^{A \oplus D}$ algorithm as follows: we guess a path p of $\text{SAT}^{A \oplus C}(x)$ by nondeterministically guessing answers to queries to C . If p is rejecting, we reject. If p is accepting, we can then verify that our answers were correct by guessing accepting paths y_1, \dots, y_r of M^D on those C -queries answered "yes" and accepting paths n_1, \dots, n_s of N^D on those C -queries answered "no." We accept iff we find such paths. The algorithm shows us that

$$\text{SAT}^{A \oplus C} \in \text{NP}^{A \oplus D} \subseteq \text{NP}^{(\Sigma_{k-1}^p)^A} \subseteq \Sigma_k^{p,A},$$

and thus Σ_k^p satisfies the conditions of Corollary 8. \square

Proof of Theorem 7: This proof combines Lemma 6 with a certain indirect NP hiding technique. The trick is to define a language in $\text{NP}^{A \oplus C}$ which depends on too much of G to be reducible to $G \oplus C$ in polynomial time.

Let $A = M^G$ for a ptime DOTM (deterministic OTM) M with private oracle access to C , and let $G = N^A$ for some DOTM N whose use of A is restricted to queries of polynomial length (N^A computes G from A via C^A). The machine N exists because C is polynomial-use bounded, although N will not necessarily run in polynomial time. We have

$$G = N^{M^G},$$

so by Lemma 6 there is a string $\sigma_0 \prec G$ such that for all $B \succ \sigma_0$, $B = N^{M^B}$. By making a few minor adjustments to M and N , we can assume without loss of generality that

$$(\forall B) B = N^{M^B}, \quad (6)$$

as well as $A = M^G$ and $G = N^A$.

We define a language $L^A \in \text{NP}^{A \oplus C}$ and show that $L^A \not\leq_{\text{T}}^p G \oplus C$, which proves the theorem. If $t(n)$ is a monotone, recursive bound on the running time of $N^B(x)$ for all B and for all x of length n , then there is a monotone, unbounded, ptime computable function $b(n)$ such that $\max\{b(n), 2^{b(n)} \cdot t(b(n))\} \leq n$ for all n . (For example, if $t(n)$ is exponential, then $b(n) = \log \log n$ will certainly do.) Suppose p is a monotone polynomial bounding both the running time of M and the length of oracle queries of N . Let

$$L^A \stackrel{\text{df}}{=} \left\{ 0^n : \exists x_{|x| \leq p(n)} [A(x) \neq M^{(N^A)^{<b(n)}}(x)] \right\}.$$

To see that $L^A \in \text{NP}^{A \oplus C}$, we note that for $|x| \leq p(n)$, $M^{(N^A)^{<b(n)}}(x)$ can be computed from $A \oplus C$ in deterministic time polynomial in n , owing to our choice of $b(n)$. Both A and C are needed for the computation; recall that M has private oracle access to C . Now let Q be any ptime DOTM with private oracle access to C , and let q be a polynomial bound on Q 's running time. If we show that $L^A \neq Q^G$, we are done; since Q was chosen arbitrarily, we have $L^A \not\leq_{\text{T}}^p G \oplus C$. Since G is 1-generic, we need only show that for every string $\sigma \prec G$ there is a string $\tau \succeq \sigma$ and x such that

$$L^{M^\tau}(x) \downarrow \neq Q^\tau(x) \downarrow. \quad (7)$$

For if this is the case, then G must meet the set of all such τ , and so $Q^G \neq L^{M^G} = L^A$.

Given any σ , let m_0 be the least m such that $\sigma(z)$ is undefined for all z of length m , and let n_0 be least such

that $b(n_0) \geq m_0$ and $q(n_0) < 2^{n_0}$. We define $\hat{\tau}$ to be σ extended with zeros just far enough to be defined on all y of length at most $\max\{q(n_0), p(p(n_0))\}$. Note that for any sets $\hat{G} \succ \hat{\tau}$ and $\hat{A} \stackrel{\text{df}}{=} M^{\hat{G}}$, we have $\hat{G} = N^{\hat{A}}$ by Equation (6). Thus for all x of length at most $p(n_0)$, we have

$$\begin{aligned} \hat{A}(x) &= M^{\hat{G}}(x) = M^{\hat{G}^{<m_0}}(x) \\ &= M^{\hat{G}^{<b(n_0)}}(x) = M^{(N^{\hat{A}})^{<b(n_0)}}(x). \end{aligned}$$

Therefore, $0^{n_0} \notin L^{\hat{A}}$. If $Q^{\hat{\tau}}(0^{n_0})$ accepts, then letting $\tau \stackrel{\text{df}}{=} \hat{\tau}$ and $x \stackrel{\text{df}}{=} 0^{n_0}$ satisfies Equation (7).

If $Q^{\hat{\tau}}(0^{n_0})$ rejects, then let y be least such that $|y| = n_0$ and y is not queried by $Q^{\hat{\tau}}(0^{n_0})$. Define τ to be the same as $\hat{\tau}$ except that $\tau(y) = 1$. $Q^\tau(0^{n_0})$ still rejects. Now consider any $\tilde{G} \succ \tau$ and $\tilde{A} \stackrel{\text{df}}{=} M^{\tilde{G}}$; further, let $D \stackrel{\text{df}}{=} M^{\tilde{G}^{<b(n_0)}}$. Again by Equation (6) we have

$$\tilde{G} = N^{\tilde{A}} \text{ and } \tilde{G}^{<b(n_0)} = N^D.$$

But $\tilde{G}(y) \neq \tilde{G}^{<b(n_0)}(y)$, and so there must be some x queried by $N^{\tilde{A}}(y)$ such that $\tilde{A}(x) \neq D(x)$. That is, there is an x with $|x| \leq p(n_0)$ such that

$$\begin{aligned} \tilde{A}(x) &\neq D(x) \\ &= M^{\tilde{G}^{<b(n_0)}}(x) = M^{(N^{\tilde{A}})^{<b(n_0)}}(x). \end{aligned}$$

Thus $0^{n_0} \in L^{\tilde{A}}$, and Equation (7) is satisfied. \square

Remark: The machines M and N in the previous proof can be viewed as total recursive operators (e.g., mapping a set B to M^B). Equation (6) then says that their composition, $N \circ M$, is the identity. This immediately implies that M is one-to-one and N is onto. The injectivity of M is evident, for example, when we argue that changing a single bit of G must change A somewhere.

Stephan [Ste95] has an alternate proof of Theorem 7 which does not use 1-generic sets. With the hypothesis that $\text{NP} = \text{co-NP}$, he constructs an NP-hard degree in EXP with no NP-jump inverse.

4 NP-Intermediacy of 1-Generic Sets

Although we suspect that $G \oplus \text{SAT}$ cannot be in the degree of SAT^A for any A , a natural question to ask is whether G or $G \oplus \text{SAT}$ can lie strictly between A and SAT^A for some A . The next theorem shows this to be the case for both G and $G \oplus \text{SAT}$. In fact, we get

$G \in \text{NP}^A - \text{P}^A$. Unlike Theorem 7, Theorem 10 does correspond positively with a recursion theoretic result regarding 1-generic sets, namely, every 1-generic set G is r.e. in some set strictly below G [Joc80].

Theorem 10 For any 1-generic set G there is a set A such that for all recursive R ,

1. $A \leq_{tt}^p G \leq_m^p \text{SAT}^A$,
2. $\text{SAT}^A \not\leq_T^p G \oplus R$, and
3. $G \not\leq^p A \oplus R$.

Interestingly, replacing the tt-reducibility in Theorem 10 by a more restrictive reducibility, such as \leq_m^p or even just $\leq_{\oplus-tt}^p$, implies $\text{P} \neq \text{NP}$. This is also true for $G \oplus \text{SAT}$ as well as G . Thus we cannot get such a stronger result without either using unproven complexity theoretic assumptions or proving $\text{P} \neq \text{NP}$.

Theorem 12 Let D be an arbitrary recursive set. If G is 1-generic and there is an A such that

$$A \leq_{\oplus-tt}^p \text{SAT}^A \leq^p$$

then $G \leq^p \text{SAT}^D$.

Proof Sketch: For any oracle D we define a length preserving function

$$\xi^D(x) \stackrel{\text{df}}{=} D(x1)D(x10)D(x100) \cdots D(x10^{x-1})$$

and two languages

$$L_1^D \stackrel{\text{df}}{=} \{\langle x, \xi^D(x) \rangle \mid x \in D\}, \quad L_2^D \stackrel{\text{df}}{=} \{1^n \mid (\exists \langle x, y \rangle \in D) |x| = |y|\}$$

Corollary 13 If there is a 1-generic set G , a set $D \in \text{PH}$, and a set A such that

Given a 1-generic G , let $A \stackrel{\text{df}}{=} L_1^G$. Clearly, $A \leq_{\oplus-tt}^p G \oplus D \leq_T^p \text{SAT}^A$ and $G \not\leq_T^p A$, and for all x ,

$$x \in G \iff (\exists y)[|x| = |y| \ \& \ \langle x, y \rangle \in A],$$

then $\text{P} \neq \text{NP}$.

Proof: If $\text{P} = \text{NP}$, then $\text{SAT}^D \in \text{P}$ and so $G \leq^p$

so $G \in \text{NP}^A$, i.e., $G \leq_m^p \text{SAT}^A$. Theorem 12. \square

Let R be a recursive set and let M be any DOTM running in polynomial time with the private oracle access to R . Since $L_2^D \in \text{NP}^D$ for all D , it suffices to show that $L_2^A \neq M^G$ and $G \neq M^A$ with finite extensions arguments starting with a prefix space over $\Sigma \subseteq \omega$ is any set, then we let $L_2^A \neq M^G$, start with a string denoting the set of all characteristic functions with enough. If $M^{\sigma 0^{\infty}}(1^n)$ accepts, then we can naturally view 2^S as a vector space over the field \mathbb{Z}_2 , with vector addition being componentwise addition modulo 2. For $u, v \in 2^S$, we let $u+v$ denote the vector sum of u and v .² Suppose M computes a parity-tt reduction by making nonadaptive queries. To see how M induces linear maps between these vector spaces, one can readily verify that for any σ and τ with the same domain, $M^{\sigma+\tau} = M^\sigma + M^\tau$. (M^σ , M^τ , and $M^{\sigma+\tau}$ all have the same domain.)

Showing $G \neq M^A$ is a similar routine application of NP hiding, and the details are left to the reader. \square

Proof of Theorem 12: Suppose $A \leq_{\oplus-tt}^p G$ via a ptime DOTM M which can privately make unrestricted adaptive queries to D but only parity-tt queries to G . Suppose further that N is a ptime

Corollary 11 For every 1-generic set G there is an A such that both G and $G \oplus \text{SAT}$ are in $\text{NP}^A - \text{P}^A$, and not in the complete degree of NP^A .

²It is customary to use \oplus to denote this sum, but \oplus is overloaded enough in this paper as it is.

DOTM and $G = N^{\text{SAT}^A}$. As in the proof of Theorem 7, we then have

$$G = N^{\text{SAT}^{M^G}} = R^G,$$

where R is a DOTM simulating the computation of $N^{\text{SAT}^{M^G}}$. By Lemma 6, there is a string $\sigma_0 \prec G$ such that $(\forall B \succ \sigma_0) B = R^B = N^{\text{SAT}^{M^B}}$. Again by tweaking M and N slightly, we may also assume that

$$(\forall B) B = R^B = N^{\text{SAT}^{M^B}}. \quad (8)$$

Given input x , we describe a procedure to compute $G(x)$ from $A \oplus \text{SAT}^D$. First we find finite functions α, γ with size polynomially bounded in $|x|$ such that $\alpha \preceq M^\gamma$ and α determines the value of $N^{\text{SAT}^\alpha}(x)$ (i.e., makes the computation defined). We can do this readily using the SAT^D portion of the oracle:

We start with $\alpha := \gamma := \emptyset$, and we simulate the computation of $N(x)$. Each time N makes an oracle query q , we determine if there are polynomial sized $\sigma \succeq \alpha$ and $\tau \succeq \gamma$ such that $\sigma \preceq M^\tau$ and $\text{SAT}^\sigma(q) = 1$. If there are *any* such σ and τ , then there are ones of polynomial size: it suffices that M^τ yield a σ that gives answers to queries made along some accepting path of $\text{SAT}^\sigma(q)$. Moreover, such σ and τ can be found easily via prefix search using SAT^D . (The prefix search asks existential questions about the computation of M , which itself uses D implicitly as an oracle.) If there are such σ and τ , we set $\alpha := \sigma$ and $\gamma := \tau$, answer “yes” to N ’s query, and continue. If not, we answer “no” and continue.

[Note that in general $\alpha \not\prec A$ and $\gamma \not\prec G$, so we cannot simply output $\gamma(x)$ at this point.] We let $S \stackrel{\text{df}}{=} \text{domain}(\alpha)$ and $T \stackrel{\text{df}}{=} \text{domain}(\gamma)$, and compute $\beta \stackrel{\text{df}}{=} A \setminus S$, making queries to A . Finally, if $x \notin T$, output 0. Otherwise, find a $\zeta: T \rightarrow \{0, 1\}$ such that $M^\zeta(y) = A(y)$ for all $y \in S$ (prefix search using SAT ; note that $|S|$ and $\max(S)$ are both polynomially bounded in $|x|$), and output $\zeta(x)$. This ends the algorithm.

The procedure above clearly runs in polynomial time relative to $A \oplus \text{SAT}^D$. To see that it outputs $G(x)$, we first observe that $x \in T$; otherwise, for any set $B \succ \gamma$, the computation of $R^B(x)$ does not depend on $B(x)$. That is, we have $R^{B \cup \{x\}}(x) = R^{B - \{x\}}(x)$,

which violates Equation (8) above. Next, we observe that ζ exists; this follows from the fact that $M^\gamma(y)$ is defined for all $y \in S$ and M is a tt-reduction from A to G .

We now consider the map $f: 2^T \rightarrow 2^S$ induced by the computation of M on the members of S :

$$(\forall u \in 2^T) f(u) \stackrel{\text{df}}{=} M^u \setminus S.$$

This is a linear map (M is a parity tt-reduction), and $\alpha, \beta \in \text{range}(f)$. This means that $f^{-1}(\alpha)$ and $f^{-1}(\beta)$ are nonempty affine subspaces of 2^T . Furthermore, it is easy to verify that $f^{-1}(\beta)$ is an affine translate of $f^{-1}(\alpha)$, that is, there is a $u_0 \in 2^T$ such that $f^{-1}(\beta) = f^{-1}(\alpha) + u_0$. For any $v, w \in f^{-1}(\alpha)$, we have $R^v(x) \downarrow = R^w(x) \downarrow = N^{\text{SAT}^\alpha}(x) \downarrow$, hence $v(x) = w(x)$ by Equation (8). Since ζ and $G \setminus T$ are both elements of $f^{-1}(\beta)$, there are vectors $v, w \in f^{-1}(\alpha)$ such that

$$\zeta = v + u_0$$

and

$$G \setminus T = w + u_0.$$

But then,

$$\begin{aligned} \zeta(x) &= [v + u_0](x) \\ &= [w + u_0](x) = [G \setminus T](x) = G(x), \end{aligned}$$

and the algorithm is correct. \square

5 Related Questions and Further Work

We conjecture that the NP-jump is not invertible. By Corollary 9, we know that proving the opposite would be as hard as proving $\text{NP} \neq \text{co-NP}$. It would be interesting if one could prove that the NP-jump *is* invertible based on some reasonable complexity theoretic assumptions (ones that hold in a relativized world, say). We present the following proposition as evidence that the NP-jump is not invertible, and we hope it will be helpful in obtaining a proof of noninvertibility.

Proposition 14 *Suppose G is 1-generic and M is a ptime DOTM such that*

$$G \oplus \text{SAT} \equiv_{\text{T}}^{\text{P}} \text{SAT}^A,$$

where $A = M^{G \oplus \text{SAT}}$. Then the following hold:

1. $\text{NP} \neq \text{co-NP}$.
2. $\text{NP}^{A \oplus \text{SAT}} \neq \text{NP}^A$.
3. A is not NP-hard.
4. $A \not\leq_{\text{T}}^{\text{P}} G$.
5. $M^{\emptyset \oplus \text{SAT}}$ is low for P^{NP} .

Proof Sketch:

1. This follows immediately from Corollary 9 and Proposition 5.
2. In the proof of Theorem 7 with $\mathcal{C} = \text{NP}$, the set L^A is in $\text{NP}^{A \oplus \text{SAT}}$, and $L^A \not\leq_T^p G \oplus \text{SAT}$. If $L^A \in \text{NP}^A$, then $\text{SAT}^A \leq_T^p G \oplus \text{SAT}$, contradiction.
3. This follows immediately from the item 2.
4. If $A \leq_T^p G$ we can remove the SAT portion of M 's oracle, which puts L^A into NP^A . We proceed as in the proof of item 2.
5. Let Q be a ptime DOTM such that $\text{SAT}^A = \text{SAT}^{M^{G \oplus \text{SAT}}} = Q^{G \oplus \text{SAT}}$. By adapting the proof of Lemma 6, we may assume that for all B ,

$$\text{SAT}^{M^{B \oplus \text{SAT}}} = Q^{B \oplus \text{SAT}}.$$

Setting $B = \emptyset$ yields

$$\text{SAT}^{M^{\emptyset \oplus \text{SAT}}} = Q^{\emptyset \oplus \text{SAT}}.$$

The right hand side is in P^{NP} .

□

Is every NP-hard set relatively NP-complete? That is, for every set B with $\text{SAT} \leq_T^p B$, is there an A such that B is NP^A -complete? This question is different from the one addressed in Section 3, because (ptime Turing) completeness for NP relative to an oracle A customarily means that the witnessing ptime Turing reductions also have access to A . We are not particularly interested in this question here because (1) it does not directly relate to the structure of the ptime Turing degrees, and (2) it can be affirmed trivially: given an NP-hard set B , choose any A such that $B \leq_T^p A$ and $\text{P}^A = \text{NP}^A$.

The set $G \oplus C$ of Theorem 7 is not recursive, and it begs the question of whether the C -jump is not invertible on the ptime degrees of recursive sets. A more careful analysis of the requirements of Theorem 7 should yield a recursive witness to noninvertibility. It is indeed likely that some resource-bounded notion of genericity—of the type studied by Ambos-Spies *et al.* [ASNT94, AS95]—will suffice.

Is the NEXP-jump invertible? What about Σ -levels of the exponential hierarchy? Neither Theorem 3 nor Theorem 7 immediately applies to these classes.

Returning to the question of NP-jump invertibility, is there a relativized world where the NP-jump *is* invertible? In phrasing this question, one must make

sure that all computations have free access to the oracle, including the reductions. It is easily checked that this question is equivalent to the existence of an oracle O such that

$$(\forall B)(\exists A)[B \leq_T^p \text{SAT}^{A \oplus O} \leq_T^p B \oplus \text{SAT}^O],$$

or equivalently,

$$(\exists B \text{ 1-generic in } O)(\exists A) [B \leq_T^p \text{SAT}^{A \oplus O} \leq_T^p B \oplus \text{SAT}^O].$$

Acknowledgments

I wish to thank Lance Fortnow, Steve Homer, and Stuart Kurtz for several helpful discussions about jump inversion in complexity theory, especially Lance for trying his hand at an oracle that makes the NP-jump invertible. Special thanks to Luc Longpré for pointing out a serious error in a previous draft, and to Peter Fejer for showing me Mohrherr's result [Moh84]. Frank Stephan deserves my gratitude for sharing his results about the EXP- and other jumps. Finally, I would like to thank Harry Buhrman for suggesting this problem to me in the first place.

References

- [AS95] K. Ambos-Spies, 1995. Private communication.
- [ASNT94] K. Ambos-Spies, H.-C. Neis, and S. A. Terwijn. Genericity and measure for exponential time. In *Proceedings of MFCS '94*, Lecture Notes in Computer Science, pages 221–232. Springer-Verlag, 1994. Extended abstract.
- [BDG88] J. L. Balcázar, J. Díaz, and J. Gabarró. *Structural Complexity I*, volume 11 of *EATCS Monographs on Theoretical Computer Science*. Springer-Verlag, 1988.
- [BH77] L. Berman and J. Hartmanis. On isomorphism and density of NP and other complete sets. *SIAM Journal on Computing*, 1:305–322, 1977.
- [BI87] M. Blum and R. Impagliazzo. Generic oracles and oracle classes. In *Proceedings of the 28th IEEE Symposium on Foundations of Computer Science*, pages 118–126, New York, 1987. IEEE.
- [FFKL93] S. Fenner, L. Fortnow, S. Kurtz, and L. Li. An oracle builder's toolkit. In *Proceedings of the 8th IEEE Structure in Complexity Theory Conference*, pages 120–131, 1993.
- [For93] L. Fortnow, 1993. Private communication.
- [Fri57] R. M. Friedberg. A criterion for completeness of degrees of unsolvability. *Journal of Symbolic Logic*, 22:159–160, 1957.

- [GJ86] J. Goldsmith and D. Joseph. Three results on the polynomial isomorphism of complete sets. In *Proceedings of the 27th Annual IEEE Symposium on Foundations of Computer Science*, pages 390–397, 1986.
- [HU79] J. E. Hopcroft and J. D. Ullman. *Introduction to Automata Theory, Languages, and Computation*. Addison-Wesley, Reading, Mass., 1979.
- [Joc80] C. G. Jockusch. Degrees of generic sets. In F. R. Drake and S. S. Wainer, editors, *Recursion Theory: Its Generalizations and Applications*, pages 110–139. Cambridge University Press, 1980.
- [Lad75] R. Ladner. On the structure of polynomial-time reducibility. *Journal of the ACM*, 22:155–171, 1975.
- [Moh84] J. Mohrherr. Density of a final segment of the truth-table degrees. *Pacific J. Math.*, 15(2):409–419, 1984.
- [Myh55] J. Myhill. Creative sets. *Z. Math. Logik Grundlagen Math*, 1:97–108, 1955.
- [Rog67] H. Rogers. *Theory of Recursive Functions and Effective Computability*. McGraw-Hill, 1967. Reprinted. MIT Press. 1987.
- [Soa87] R. I. Soare. *Recursively Enumerable Sets and Degrees*. Springer-Verlag, Berlin, 1987.
- [Ste95] F. Stephan, 1995. Private communication.