Hyper-Polynomial Hierarchies and the NP-Jump

Stephen Fenner *
University of Southern Maine

Steven Homer †
Boston University

Randall Pruim ‡
Boston University

Marcus Schaefer §
University of Chicago

September 20, 1997

Abstract

Assuming that the polynomial hierarchy (PH) does not collapse, we show the existence of ascending sequences of ptime Turing degrees of length ω_1^{CK} all of which are in PSPACE and uniformly hard for PH, such that successors are NP-jumps of their predecessors. This is analgous to the hyperarithmetic hierarchy, which is defined similarly but with the (recursive) Turing degrees. The lack of uniform least upper bounds for ascending sequences of ptime degrees causes (the limit levels of) our hyper-polynomial hierarchy to be inherently non-canonical. This problem is investigated in depth, and various possible structures for hyper-polynomial hierarchies are explicated, as are properties of the NP-jump operator on the languages which are in PSPACE but not in PH.

*Computer Science Department, University of Southern Maine, Portland, ME 04104. E-mail: fenner@cs.usm.maine.edu.

Supported in part by the NSF under grants CCR 92-09833 and CCR 95-01794.

†Computer Science Department, Boston University, Boston, MA 02215. E-mail: homer@cs.bu.edu.

Supported in part by the NSF under grant NSF-CCR-9400229 and by the Mathematical Institute, Oxford University.

‡Computer Science Department, Boston University, Boston, MA 02215. E-mail: rpruim@calvin.edu. On leave from Department of Mathematics and Computer Science, Calvin College, Grand Rapids, MI 49546.

§Computer Science Department, University of Southern Maine, Portland, ME 04104. E-mail: schaefer@cs.uchicago.edu.

Supported in part by the NSF under grant CCR 95-01794.

On leave from Department of Computer Science, University of Chicago, Chicago, IL 60637.

1 Introduction

Since its definition in 1976, [Sto77] the polynomial hierarchy has been used to classify and measure the complexity of infeasible combinatorial problems. It has been hugely successful in this capacity, providing the main framework for complexity classes above polynomial time within which most subsequent complexity theory has taken place. The classes in this hierarchy, particularly in the first few levels of the hierarchy, have been studied extensively and their structure carefully examined. In this paper we consider extensions of the polynomial hierarchy into extended hierarchies, all lying within PSPACE. Our aim is to provide tools for further understanding of many complex and interesting PSPACE problems which lie just outside PH as well as to gain further understanding of the intricacies of ptime reductions, degrees and the NP-jump operator. The NP-jump has proven to be a fundamental and useful tool in complexity theory. It is the central concept in the definition of the high and low hierarchies. Its properties have recently been explored by Fenner [Fen95].

The polynomial time hierarchy was defined and motivated in analogy with the arithmetic hierarchy first studied by Stephen Kleene. The structure and many key properties of the classes in the polynomial hierarchy are similar to those in the arithmetic hierarchy. Furthermore various concepts and definitions originating in the arithmetic hierarchy have been important in illuminating interesting aspects of the complexity theory of problems in the resource bounded setting. For example, the alternating quantifier characterizations of the levels of both the arithmetic and polynomial hierarchies provides a simple and useful method for placing problems within levels of these hierarchies.[1] One of the deepest and most elegant developments in this area of mathematical logic was the extension of the arithmetic hierarchy to the hyperarithmetic hierarchy by transfinite iteration of the Turing jump operator and the subsequent development by Kleene, Spector and others of the properties of this hierarchy. Our work here intends to develop an analogous resource bounded framework for problems lying within PSPACE and above PH. In this work we define and (under reasonable assumptions) prove the existence of hyper-polynomial hierarchies formed by transfinite iteration of the NP-jump operator and study their properties and the properties of the NP-jump operator in the realm between PH and PSPACE.

Assuming the polynomial hierarchy is infinite, Ambos-Spies [AS89] has shown the existence of a rich, infinite partial order of degrees in PSPACE above PH. In this paper we extend his techniques to define infinite, *NP-jump-respecting* hierar-

[1] *Is there something we can say about alternating quantifiers and hardness in the* p-HYP*hierarchy ? There is such in the* HYP*hierarchy, though its not as neat as for finite levels.*

chies of length ω_1^{CK} in PSPACE above PH, which naturally extend the polynomial hierarchy. This shows that if PH does not collapse then not only is there a rich and complex structure to the degrees in $\mathrm{PSPACE} - \mathrm{PH}$, but that PSPACE is in some sense "very far" from PH, since not even ω_1^{CK} many NP-jumps suffice to get from PH to PSPACE. We are hopeful that the classes of problems hard for levels of these hierarchies will also provide a new classification scheme for interesting hard combinatorial problems, such as the PP-complete languages, which lie in PSPACE but above PH.[2]

The major technical obstacle encountered in proving the existence of an extended polynomial hierarchy is the lack of uniform least upper bounds for ascending sequences of ptime degrees. This fact was first noted by Ambos-Spies [AS89], and makes the definition of our hierarchies non-canonical at limit levels, giving rise to several possibilities for the properties of the extended hierarchy. This situation is explored in depth here and various possible structures for the hyper-polynomial hierarchy are explicated. For example, under reasonable assumptions about the structure of uniformly hard sets for PH, we prove that there is a problem which is a uniform upper bound for PH but is not such a bound for any ptime non-constant alternation class. Such a problem would lie "just above" PH, and a careful examination of the proof of Toda's Theorem [Tod91] indicates that the PP-complete languages may fit this description.

Outline. After providing the necessary background on recursive ordinals and uniform upper bounds in Section 2, we constuct in Section 3 an infinite hierarchy of problems of length ω_1^{CK} in PSPACE above PH. This hierarchy is proper provided that PH doesn't collapse. In Sections 4 and 5 we investigate the extent to which such a hierarchy is or is not canonical by asking where within $\mathrm{PSPACE} - \mathrm{PH}$ such a hierarchy can be placed. Finally, in Section 6 we present some directions for further investigation.

2 Preliminaries

We identify ω, the natural numbers, with Σ^*, the set of all binary strings, via the usual diadic representation. We let ϵ be the empty string and denote by $\langle\cdot,\cdot\rangle : \Sigma^* \times \Sigma^* \to \Sigma^*$ a standard ptime-computable, ptime-invertible bijection such that $\langle\epsilon,\epsilon\rangle = \epsilon$, and $\langle x, y\rangle > y$ for all $x \neq \epsilon$.

[2] *Here is a problem we need to address some how. We don't actually classify any problems here, yet this is a primary motivation. Are there any (even trivial) consequences of* PP*-complete sets lying somewhere in* p-HYP*? Should we say anything about an extended Toda's theorem here? Are there any (even very artificial) problems in* PSPACE *which we can say are hard for our extensions, or prove are not hard? Can we say anything precise about* PP*-complete sets here?*

We fix a standard, acceptable enumeration $N_0, N_1, N_2, \ldots$ of nondeterministic oracle TMs, and a standard enumeration $D_0, D_1, D_2, \ldots$ of deterministic ptime oracle TMs, where for each i, $\{D_{\langle i,j\rangle}\}_{j\in\omega}$ enumerates the set of all oracle computations running in time $n^i + i$ for all oracles and inputs of length n. We write $D_e : A \leq_r B$ if $D_e^A(x) = B(x) = \chi_B(x)$ and D_e accesses the oracle A only in a manner allowed by the reduction type $\leq_r$. For the most part we are interested in $\leq_{\mathrm{T}}^p$- and $\leq_m^p$-reductions. As usual, $\varphi_0, \varphi_1, \varphi_2, \ldots$ is a standard acceptable enumeration of the computable partial functions.

Definition 2.1 *For any set $A \subseteq \Sigma^*$, we define, in the spirit of Balcázar,* et al. [BDG88],

$$\mathrm{K}(A) = \{\langle e, x, 0^t\rangle \mid N_e^A(x) \textit{ has an accepting path of length } \leq t\}.$$

We call $\mathrm{K}(A)$ the NP-jump *of A. It is complete for NP^A under (unrelativized) $\leq_m^p$-reductions.*

It is easy to check that $\mathrm{K}(\cdot)$ lifts to a well-defined operator on the $\leq_{\mathrm{T}}^p$ degrees. We denote by $\mathrm{K}^k(\cdot)$ the k-fold iteration of $\mathrm{K}(\cdot)$. Let Q be a ptime alternating oracle Turing machine such that for all A and k, $\lambda x.Q^A(0^k, x) = K^k(A)$, the canonical $(\Sigma_k^{\mathrm{P}})^A$-complete set, and write $Q_k^A(x)$ for $Q^A(0^k, x)$. We assume without loss of generality that Q has been chosen so that for any oracle A, $Q^A(0^k, x)$ only makes oracle queries of length $\leq |x|$.

2.1 Kleene's $\mathcal{O}$

Here we give a brief definition of Kleene's partial order, $\langle \mathcal{O}, <_{\mathcal{O}}\rangle$, of all notations for constructive (recursive) ordinals. Here $\mathcal{O} \subseteq \Sigma^*$ and $<_{\mathcal{O}}$ is a binary relation on $\mathcal{O}$. The information in this section comes chiefly from Sacks [Sac90], but see also [Rog67]. Our development is slightly different from, but entirely isomorphic to, Kleene's original definition. Define $\mathrm{succ}(x) = 0x$ and $\lim(x) = 1x$. We define $\langle \mathcal{O}, <_{\mathcal{O}}\rangle$ by transfinite induction. It is the least partial order such that the following hold for all $a, e \in \Sigma^*$:

1. $\epsilon \in \mathcal{O}$.
2. If $a \in \mathcal{O}$, then $\mathrm{succ}(a) \in \mathcal{O}$ and $a <_{\mathcal{O}} \mathrm{succ}(a)$.
3. If φ_e is total, $\mathrm{range}(\varphi_e) \subseteq \mathcal{O}$, and $\varphi_e(0) <_{\mathcal{O}} \varphi_e(1) <_{\mathcal{O}} \varphi_e(2) <_{\mathcal{O}} \cdots$, then $\lim(e) \in \mathcal{O}$ and $\varphi_e(n) <_{\mathcal{O}} \lim(e)$ for all $n \in \Sigma^*$.
4. $<_{\mathcal{O}}$ is transitive.

It can be shown that $\langle \mathcal{O}, <_{\mathcal{O}} \rangle$ is well-founded, and hence functions with domain $\mathcal{O}$ can be defined by transfinite recursion. For all $a \in \mathcal{O}$ we define $\|a\|$, the unique ordinal for which a is a notation, in this way:

1. $\|\epsilon\| = 0$.
2. If $a \in \mathcal{O}$, then $\|\mathrm{succ}(a)\| = \|a\| + 1$.
3. If $\lim(e) \in \mathcal{O}$, then $\|\lim(e)\| = \sup_n \|\varphi_e(n)\|$.

Each element of $\mathcal{O}$ is the notation for a constructive ordinal, and each constructive ordinal has at least one (but usually more than one) notation. Also, if $a <_{\mathcal{O}} b$, then $\|a\| < \|b\|$, but not conversely. The set of all constructive ordinals is ω_1^{CK}, which is the least non-constructive ordinal, and is countable.

The structure of $\langle \mathcal{O}, <_{\mathcal{O}} \rangle$ is a tree, where (infinite) branching occurs at every limit level. Some branches (maximal linearly ordered subsets) peter out well before reaching height ω_1^{CK} (in fact, there are branches of height only ω^2), but some branches do reach height ω_1^{CK}. The most important fact about $\mathcal{O}$ is that one can construct objects via "effective transfinite recursion" up to ω_1^{CK} by using notations from $\mathcal{O}$. We will do just that in Section 3, where we define sets H_a in PSPACE for all $a \in \mathcal{O}$ such that $a <_{\mathcal{O}} b$ implies $H_a \leq_{\mathrm{T}}^p H_b$, and $H_{\mathrm{succ}(a)} = \mathrm{K}(H_a) \not\leq_{\mathrm{T}}^p H_a$. (This last inequality assumes that PH is infinite.) This mirrors the classical construction of the hyperarithmetic hierarchy.

2.2 Uniform Upper Bounds and Padding Arrays

In computability theory, it is a simple matter to define a canonical join of a countable sequence of sets which is the least uniform $\leq_{\mathrm{T}}$-upper bound (in fact, the least uniform $\leq_m$-upper bound) for the sequence. In complexity theory this is not possible, since there is no least uniform $\leq_{\mathrm{T}}^p$-upper bound [AS89], [Lad75].[3] Furthermore, the most natural join operator has the unfortunate (for our purposes) property that the join of a collection consisting of a complete language for each level of PH is PSPACE-complete. In our case we are interested in understanding the problems which lie between PH and PSPACE and we would like the join to be as close to PH as possible. Therefore, we must work instead with uniform upper bounds, defined below, which correspond to possible choices for a nicely behaved join operator.

Definition 2.2 *Given a countable collection $\mathcal{C} = \{L_i \mid i \in \omega\}$ of languages, a uniform $\leq_r$-upper bound for $\mathcal{C}$ is a language H such that there is a computable function $f : \omega \to \omega$ with the property that for all i, $D_{f(i)} : L_i \leq_r H$.*

[3] *Is this now the correct statement of the Ambos-Spies result?*

We are primarily interested in uniform upper bounds for PH and similar classes. A uniform upper bound for PH is a uniform upper bound for $\{\mathrm{K}^i(\emptyset) \mid i \in \omega\}$.

Definition 2.3 *For any function $f : \omega \times \omega \to \omega$ and any countable collection of languages $\mathcal{C} = \{L_i \mid i \in \omega\}$, the* padding array *for C via f is the language defined by*

$$A = \{\langle k, 0^n 1x\rangle \mid x \in L_k \ \&\ n = f(k, |x|)\}.$$

Two types of padding arrays are of special interest.

1. *If for every k, $f(k, *)$ is monotone non-decreasing and $0^n \mapsto 0^{f(k,n)}$ is ptime computable, and for every n $f(*, n)$ is monotone non-decreasing, then we say that A is a* ptime padding array *via f.*

2. *If in addition, there is a constant d such that for all k, $f(k, *) = \mathrm{O}(n^d)$ then we say that A is a* ptime padding array of degree d *via f.*

A ptime padding array for PH is a ptime padding array for $\{\mathrm{K}^i(\emptyset) \mid i \in \omega\}$.

As the following lemma shows, padding arrays are particularly nice uniform upper bounds.

Lemma 2.4 *If A is a ptime padding array for $\mathcal{C} = \{L_i \mid i \in \omega\}$ via f then A is a uniform $\leq_m^p$-upper bound for $\mathcal{C}$.*

Proof. The map $r_i : x \mapsto \langle i, 0^{f(i,|x|)}1x\rangle$ is a many-one reduction from L_i to A. $\square$

As a partial converse to the result above we have the following lemma.

Definition 2.5 *A function f is* nice *if f is monotone non-decreasing, unbounded, and can be computed in $\mathrm{O}(n + f(n))$ steps.*

Lemma 2.6 *Let A be a uniform $\leq_r$-upper bound for $\mathcal{C} = \{L_k \mid k \in \omega\}$ via f. If B is the ptime padding array for $\mathcal{C}$ via g, and g is a nice function such that for all x with $|x| = m$, $D_{f(j)}(x)$ halts in fewer than $g(j, m)$ steps, then $B \leq_r A$.*

Proof. We describe a reduction from B to A. On input $x = \langle k, 0^l 1y\rangle$:

1. If $l \neq g(k, |y|)$, then $x \notin A$. This can be determined in polynomial time because g is nice.

2. If $l = g(k, |y|)$, then compute $D^A_{f(k)}(y)$. Since $l = g(k, y)$ is greater than the number of steps required to compute $D^A_{f(k)}(y)$, this can also be done in polynomial time.

$\square$

Thus, in particular, every ptime padding array is a uniform $\leq_m^p$-upper bound for PH, and every set which is a uniform $\leq_r$-upper bound for PH is $\leq_r$-above some ptime padding array for PH.

3 A Hyper-Polynomial Hierarchy

We now come to the construction of an extended polynomial hierarchy. We show how to embed $\langle \mathcal{O}, <_{\mathcal{O}} \rangle$ into $\langle \mathrm{PSPACE}, \leq_{\mathrm{T}}^p \rangle$ in such a way that successors correspond to NP-jumps and limits to uniform upper bounds. We will call such an embedding a hyper-polynomial hierarchy, or p-HYP. More formally, we construct a set H such that if H_a denotes $\{x \mid \langle a, x \rangle \in H\} = \lambda x.H(a,x)$, then H satisfies the following properties.

(P1) $H \in \mathrm{PSPACE}$.

(P2) $H_\epsilon = \emptyset$.

(P3) $H_{\mathrm{succ}(a)} = \mathrm{K}(H_a)$ for all a.

(P4) For any e with $\lim(e) \in \mathcal{O}$, $H_{\lim(e)}$ is a uniform upper bound for $\{H_a \mid a <_{\mathcal{O}} \lim(e)\}$.

(P5) If PH is infinite, then for any $a \in \mathcal{O}$, $H_{\mathrm{succ}(a)} = \mathrm{K}(H_a) \not\leq_{\mathrm{T}}^p H_a$.

Although H_a is defined here for all $a \in \Sigma^*$, we are really only interested in H_a when $a \in \mathcal{O}$. H will be constructed by transfinite induction over $\mathcal{O}$. We will say that H is *universal* for this hyper-polynomial hierarchy.

To ensure the last two properties, we must build $H_{\lim(e)}$ carefully so that $\mathrm{PH}^{H_{\lim(e)}}$ is infinite. At the same time, we must code into $H_{\lim(e)}$ all H_a for $a <_{\mathcal{O}} \lim(e)$. We do the latter by making $H_{\lim(e)}$ a uniform upper bound of $\{H_{\varphi_e(y)} : y \in \omega\}$, which we do by making $H_{\lim(e)}$ a ptime padding array for $\{H_{\varphi_e(y)} : y \in \omega\}$. Now to get PH to separate over $H_{\lim(e)}$ we delay coding each $H_{\varphi_e(y)}$ into $H_{\lim(e)}$ until we notice that some designated $\leq_{\mathrm{T}}^p$-reduction D_i fails to reduce some Σ-level of the hierarchy over $H_{\lim(e)}$ to the previous level (say, the $(k+1)$st to the kth). If we can do this for all i and k, we are done.

Assuming by transfinite induction that $\mathrm{PH}^{H_{\varphi_e(y)}}$ separates for all y, this can be accomplished by delayed diagonalization. We are guaranteed to kill off our reduction D_i just by waiting long enough before coding each column: $H_{\lim(e)}$ will "look like" $H_{\varphi_e(y)}$ and eventually fool D_i. The particular "delayed diagonalization" strategy employed here is is similar to those used by Ambos-Spies [AS89], which in turn are based on well-known techniques of Ladner [Lad75].

We now define H formally by simultaneous transfinite induction over $\mathcal{O}$ and length-decreasing recursion. In what follows, $a, e, x \in \Sigma^*$ are arbitrary. The limit case is as explained above. We need to perform the diagonalization via a look-back technique in order to keep H in PSPACE—this explains the stringent bounds on i, k, and w in 3(b).

1. $H_\epsilon(x) = 0$ (thus $H_\epsilon = \emptyset$).
2. $H_{\mathrm{succ}(a)}(x) = Q_1^{H_a}(x)$ (thus $H_{\mathrm{succ}(a)} = \mathrm{K}(H_a)$).
3. $H_{\mathrm{lim}(e)}(\langle y, z\rangle) = 0$, unless z is of the form $0^s 1 v$, where s is least (if it exists) such that

   (a) $\varphi_e(0), \varphi_e(1), \ldots, \varphi_e(y)$ all halt in a combined total of $\leq s$ steps, and

   (b) there is "sufficient evidence" that

   $$(\forall k)\ Q_{k+1}^{H_{\mathrm{lim}(e)}} \not\leq_{\mathrm{T}}^{p} Q_k^{H_{\mathrm{lim}(e)}}.$$

   We will say there is sufficient evidence if $(\forall i < \log^* y)(\forall k < |y|)(\exists w < \log^* s)$ such that

   $$Q_{k+1}^{H_{\mathrm{lim}(e)}}(w) \neq D_i^{Q_k^{H_{\mathrm{lim}(e)}}}(w).$$

   If such a least s exists and $z = 0^s 1 v$ for some $v \in \Sigma^*$, then we let

   $$H_{\mathrm{lim}(e)}(\langle y, z\rangle) = H_{\varphi_e(y)}(v).$$

   In this way, $\lambda z. H_{\mathrm{lim}(e)}(\langle y, z\rangle)$ will be a padded version of $H_{\varphi_e(y)}$.

Theorem 3.1 *The set H satisfies properties (P1–P5) listed above.*

Proof. It is not too difficult to see that $H \in \mathrm{PSPACE}$: the recursive aspects of the definition are all length-decreasing—due to the stringent bounds on i, k, and w in 3(b)—and the rest of the algorithm clearly needs no more than a polynomial amount of space. Properties (P2) and (P3) are also clearly satisfied.

We prove properties (P4) and (P5) simultaneously by induction over $\langle \mathcal{O}, <_{\mathcal{O}} \rangle$. Actually, we need to prove a stronger property than (P5), namely:

(P5a) If PH is infinite, then PH^{H_a} is infinite.

Choose an arbitrary $a \in \mathcal{O}$ and assume that properties (P4) and (P5a) hold for all H_b with $b <_{\mathcal{O}} a$. There are three cases:

Case 1: $a = \epsilon$**.** Property (P4) holds vacuously. Since $H_\epsilon = \emptyset$, property (P5a) holds.

Case 2: $a = \mathrm{succ}(b)$**.** Again, property (P4) holds vacuously for a. Since PH^{H_b} is infinite and $H_a = \mathrm{K}(H_b)$, clearly PH^{H_a} is infinite as well.

Case 3: $a = \lim(e)$**.** Note that φ_e is total. For property (P4), we will only show that $H_{\lim(e)}$ is a uniform upper bound for $\left\{H_{\varphi_e(y)}\right\}_{y\in\Sigma^*}$. This suffices, because for any $b <_{\mathcal{O}} \lim(e)$, there is a y such that $b <_{\mathcal{O}} \varphi_e(y)$ and hence

$$H_b \leq^p_{\mathrm{T}} H_{\varphi_e(y)} \leq^p_{\mathrm{T}} H_{\lim(e)},$$

and one could furthermore find such a reduction effectively in b and $\lim(e)$, using certain basic facts about $\langle \mathcal{O}, <_{\mathcal{O}} \rangle$.

We first show that for every y, the s mentioned in case (3) of the definition of H must always exist. Assuming otherwise, let y be least such that no such corresponding s exists. Then $H_{\lim(e)}(\langle y', z\rangle) = 0$ for all $y' \geq y$ and all z, so we never code $H_{\varphi_e(y')}$ into $H_{\lim(e)}$. This makes $H_{\lim(e)} \equiv^p_{\mathrm{T}} H_{\varphi_e(y-1)}$, or if $y = 0$ then $H_{\lim(e)} = \emptyset$. Now by our inductive hypothesis, $\mathrm{PH}^{H_{\lim(e)}}$ is infinite, so in particular, for all i, k, there is a w such that

$$Q_{k+1}^{H_{\lim(e)}}(w) \neq D_i^{Q_k^{H_{\lim(e)}}}(w),$$

and hence s must exist by case 3(b) in the definition of H.

The fact that s exists for all y immediately implies that

- $\mathrm{PH}^{H_{\lim(e)}}$ is infinite, via an argument similar to the one just given, and
- for all y, a padded version of $H_{\varphi_e(y)}$ is coded into $H_{\lim(e)}$, and thus $H_{\varphi_e(y)} \leq^p_m H_{\lim(e)}$ via the mapping $v \mapsto \langle y, 0^{s(y)}1v\rangle$, where $s(y)$ is the s corresponding to y.

This concludes the proof. $\square$

4 Uniform Upper Bounds for PH

In this section and the next we investigate the extent to which the construction of extended hierarchies in Section 3 is canonical. Since uniform upper bounds can be thought of as non-canonical joins, it is inevitable that, at least level by level, such a hierarchy cannot be canonically defined. As we shall see, the fact that we were able to use ptime padding arrays of bounded degree (in fact, degree 0) for all of the uniform upper bounds in our construction allows for considerable manipulation of the structure of our extended hierarchies.

4.1 Quick Uniform Upper Bounds

The observation that all of the uniform upper bounds in our construction in the previous section were actually ptime padding arrays of degree 0 prompts the following definition. We will call a uniform $\leq_r$-upper bound A for $\mathcal{C} = \{L_i \mid i \in \omega\}$ *quick* if there is a polynomial $p(n)$ such that for each L_i there is a $\leq_r$-reduction $D_j : L_i \leq_r A$ which runs in time $p(n)$. A uniform $\leq_r$-upper bound which is not quick is *slow*. All ptime padding arrays of bounded degree are quick uniform upper bounds.

Quick uniform upper bounds for PH can also be characterized in terms of alternating time, as defined in [CKS81]. For this we define the class $\mathrm{ATIME}($poly, $f(n))$ to consist of all languages accepted by an alternating Turing machine in polynomial time and at most $f(n)$ alternations. (Note that this really is $f(n)$ alternations and not $\mathrm{O}(f(n))$.)

Lemma 4.1 *A set A is a quick uniform $\leq_r -upperboundfor \mathrm{PH}\ ifandonlyifitis \leq_r$-hard for $\mathrm{ATIME}($poly, $f(n))$ for some nice f.*

Proof. If A is a quick uniform $\leq_r$-upper bound for PH, then there is a computable function g and a polynomial p such that $D_{g(j)} : \mathrm{K}^j(\emptyset) \leq_r A$ in time $p(n)$. Let $f(n)$ be the largest j such that all of $g(0), g(1), \ldots, g(j)$ can be computed in less than $p(n)$ steps. Then f fulfills the conditions.

For the other direction let A be $\leq_r$-hard for $\mathrm{ATIME}($poly, $f(n))$ for some f as above. Consider the set $B = \{\langle j, 0^l 1y\rangle : y \in \mathrm{K}^j(\emptyset) \text{ and } j < f(l)\}$. $B \in \mathrm{ATIME}($poly, $f(n))$, so $B \leq_r A$. On the other hand $\mathrm{K}^j(\emptyset) \leq^p_m B$ via a reduction that can be found effectively. □

The following theorem says that quick uniform upper bounds for PH cannot be "just above" PH.

Theorem 4.2 *If A is a quick uniform upper bound for PH, then there is a ptime padding array (hence also a uniform upper bound for PH) B such that $\mathrm{K}(B) \leq^p_{\mathrm{T}} A$.*

The proof of Theorem 4.2 makes use of the following lemma.

Lemma 4.3 *If A and B are ptime padding arrays for PH via nice functions f and g respectively and there is some polynomial p such that*

$$f(k+1, m) \leq p(g(k, 2) + g(0, m))$$

then $\mathrm{K}(B) \leq^p_m A$.

Note that for any k, $g(k,2) + g(0,m)$ is a polynomial in m of the same degree as $g(0,*)$. So to satisfy the requirements of the lemma, A must be a ptime padding array of bounded degree. Thus, we have the theorem only for *quick* uniform upper bounds for PH. It remains open if there are slow uniform upper bounds for PH which cannot compute the jump of any other (necessarily slow) uniform upper bound for PH.

Before proceeding to the proofs of Theorem 4.2 and Lemma 4.3, we give a couple of examples.

1. If A is a padding array for PH via $f(k,m) = m^e + 2^k$ or $f(k,m) = m^e + 2^{2^k}$, where e is a constant, then $\mathrm{K}(A) \equiv_m^p A$.

2. If A is a bounded degree ptime padding array for PH via f, then the ptime padding array for PH via $g(k,m) = f(k+1,m)$ satisfies $\mathrm{K}(B) \leq_{\mathrm{T}}^p A$.

Proof of Theorem 4.2. If A is a quick uniform upper bound for PH then there is a ptime padding array for PH of bounded degree such that $C \leq_{\mathrm{T}}^p A$. By the Lemma 4.3 there is another ptime padding array for PH, B, such that $\mathrm{K}(B) \leq_{\mathrm{T}}^p C \leq_{\mathrm{T}}^p A$. $\square$

Proof of Lemma 4.3. We describe an algorithm for a reduction $r : \mathrm{K}(B) \leq_m^p A$. Remember that $\mathrm{K}(B) = \{\langle e, x, 0^t\rangle : N_e^B(x)$ accepts x in $\leq t$ steps $\}$. For this proof we will use the fact that k-QBF $= \{\check{\varphi} \mid \varphi$ is a true Σ_k^{P} formula $\} \equiv_m^p \mathrm{K}^k(\emptyset)$. WLOG we can assume that $\check{\varphi}$, the encoding of the formula φ as a binary string satisfies $|\varphi| \geq 2$.

Algorithm for $r(y)$:

1. On input y, first determine e, x, and t such that $y = \langle e, x, 0^t\rangle$. If none such exist, then $y \notin \mathrm{K}(B)$, so $r(x)$ can be chosen to be some fixed element of $\bar{B}$.

2. We need to find a QBF ψ, such that $N_e^B(x)$ accepts if and only if ψ is true. Furthermore, we must in time polynomial in $n = |y|$ be able to produce a string z where ψ is coded into B. For this we need the following observations about the queries made during the computation $N_e^B(x)$ (along any path).

   - $n = |y| = |\langle e, x, 0^t\rangle| \geq t, |x|$.
   - Therefore, if $N_e^B(x)$ makes queries to any strings q, then $|q| < t \leq |y| = n$.
   - If q does not have the form $0^{g(k,m)}1\check{\varphi}$, where φ is a Σ_k^{P} formula and $|\check{\varphi}| = m$, then we know that $q \notin B$, so we could modify N_e so that it answers such queries "internally" (via a syntax check) without actually querying the oracle.

- If $q = 0^{g(k,m)}1\check{\varphi}$, then we will say that q *is a query of type* (k, m) *about* φ. Note that $g(k, m) < n$, since $g(k, m) < |q| \leq |y| = n$.
- Let q be the query of type (k, m) such that k is maximal among the queries. Then all of the queries in the simulation of $N_e^B(x)$ not handled by the syntax check above are about Σ_k^{P} formulas with codes of length $\leq n$.

3. Using the observations above, we see that determining whether $N_e^B(x)$ accepts is equivalent to a formula of the form

$$\begin{aligned}\psi \quad \equiv \quad & \exists\, \vec{p}\, \exists\, \vec{q}\, \exists\, \vec{a} \\ & [[\![q_i \in B]\!] = a_i \;\&\; \Phi(e, x, t, p, q, a)]\end{aligned}$$

where $\vec{p}$ codes a path in the non-deterministic computation tree, $\vec{q}$ codes a sequence of queries to B, $\vec{a}$ codes answer bits to those queries, and Φ is a polynomial time predicate which checks that along path $\vec{p}$, $N_e^B(x)$ makes queries to $\vec{q}$ if the answers (to previous queries) are $\vec{a}$ and halts in an accepting configuration.

By the comments above, each predicate $[\![q_i \in B]\!] = a_i$ is $\Sigma_k^{\mathrm{P}} \cup \Pi_k^{\mathrm{P}}$, so that $\check{\psi} \in (k+1)$-QBF.

4. Finally, $r(y) = 0^{f(k+1,|\check{\psi}|)}1\check{\psi}$. Notice that $|\check{\psi}| \leq \mathrm{O}(n^3)$ and that $g(k, 2) \leq g(k, m) \leq n$, since there was a query of type (k, m). So $f(k+1, m) \leq f(k+1, \mathrm{O}(n^3)) \leq \mathrm{O}(f(k+1, n)^3) < \mathrm{O}((p(g(k, 2)+g(0, n)))^3) \leq \mathrm{O}(p(n+n^d)^3)$, where d is the degree of $g(0, *)$. Therefore, $r(y)$ can be computed in time polynomial in n.

$\square$

4.2 Slow Uniform Upper Bounds

We will now construct a set A which is a slow uniform upper bound for PH, i.e. it is an upper bound for PH, but there is no uniform time bound on the reductions from each level of the hierarchy to A. For this proof we need to assume more than that PH separates. We present one hypothesis which is sufficient; modifications are possible.

It uses a notion of subexponential advice and says roughly that no level of PH can be computed from a previous level, even with the additional aid of polynomial time reductions with subexponential advice. By increasing the padding, we can modify our construction of a p-HYP to include slow uniform upper bounds, if they exist.

We begin by defining what we mean by subexponential advice. For this, we use definitions of advice classes and reductions which are based on oracle access to the advice string. This definition is equivalent to the usual one for defining common classes such as $\mathrm{P}/poly$, P/log, etc. The motivation for our definition comes from the observation that, in defining $\mathrm{P}/\mathcal{F}$ for some function class $\mathcal{F}$ such that $\mathcal{F} \not\subseteq \mathrm{FP}$, the usual definition in terms of languages (see [BDG88]) allows the accepting machine not only to get advice from the advice string, but may also provide it with greater resource bounds, since the length of the advice string counts toward the length of the input. Thus, for example, an advice string consisting solely of a super-polynomially long sequence of 0's becomes potentially useful, not because it contains any information, but simply because of its length. Our definition avoids this problem.

Definition 4.4 (Superpolynomial advice) *Let $\mathcal{F}$ be a class of functions. For any string x, let $\hat{\chi}(x)$ be the smallest finite set for which x is an initial segment of the characteristic sequence.*

We write $A \leq_{\mathrm{T}}^{\mathrm{P}/\mathcal{F}} B$ if there is a an oracle Turing machine M which runs in polynomial time regardless of oracle and a function $h : \omega \to \{0,1\}^*$ in $\mathcal{F}$ such that

$$x \in A \iff M^{B \oplus \hat{\chi}(h(|x|))}(x) \text{ accepts.}$$

and write $A \in \mathrm{P}/\mathcal{F}$ if $A \leq_{\mathrm{T}}^{\mathrm{P}/\mathcal{F}} \emptyset$.

This is equivalent to saying that access to the advice comes by querying bits of the advice string, $h(n)$.

Similar notions could be defined for other classes of Turing machines as well.

For $f : \omega \to \omega$, let

$$\mathrm{P}/f = \mathrm{P}/\{h \ : \ (\forall n)\ |h(n)| \leq f(n)\}.$$

The classes

$$\begin{aligned} \mathrm{P}/poly &= \textstyle\bigcup_k \mathrm{P}/n^k, \text{ and} \\ \mathrm{P}/log &= \textstyle\bigcup_k \mathrm{P}/k\log(n) \end{aligned}$$

have their usual meanings, since using the advice oracle, one can calculate the advice string as a preprocess, and using the advice string one can simulate the queries to the advice oracle by checking bits of the advice string. Similar statements can be made whenever the machines are capable of querying the entire advice oracle. Also note that $\mathrm{P}/2^n$ is the class of all languages, since we can use 2^n bits to code the membership of all the strings of length n.

We can now state a hypothesis sufficient to demonstrate the existence of slow uniform upper bounds for PH.

Hypothesis H. The polynomial hierarchy does not collapse under polynomial time reductions with subexponential advice in the following sense:

$$(\exists \varepsilon > 0)(\forall m)(\forall i)(\exists j)\ \mathrm{K}^j(\emptyset) \not\leq_T^{\mathrm{DTIME}(n^i)/2^{n^\varepsilon}} \mathrm{K}^m(\emptyset).$$

Notice that Hypothesis H is (*a priori*) slightly stronger than the statement that

$$(\forall k)\ \mathrm{PH} \not\subseteq P_{\leq_{\mathrm{T}}^{\mathrm{P}/subexp}}(\mathrm{K}^k(\emptyset)),$$

where

$$subexp = \{h : (\forall \varepsilon > 0)\ |h(n)| = O(2^{n^\varepsilon})\}$$

and

$$P_{\leq_{\mathrm{T}}^{\mathrm{P}/subexp}}(\mathrm{K}^k(\emptyset)) = \{L \mid L \leq_{\mathrm{T}}^{\mathrm{P}/subexp} \mathrm{K}^k(\emptyset)\}.$$

Theorem 4.5 *If f is nice, then* ATIME(*poly*, $f(n)$) *contains a slow uniform upper bound for* PH*, unless H fails.*

Corollary 4.6 *If f is nice, then* ATIME(*poly*, $f(n)$) *contains a uniform upper bound for* PH *that is not hard for any* ATIME(*poly*, $g(n)$) *(with g nice), unless H fails.*

Proof. We define A and verify the properties.

$$A = \{0^{(|y|+2)^{2^k}} 1y : y \in \mathrm{K}^k(\emptyset)\}.$$

From the definition of A it is clear that A is a uniform upper bound for PH. If A were quick, then we would have

$$(\exists i)(\forall j)\ \mathrm{K}^j(\emptyset) \in \mathrm{DTIME}^A(n^i).$$

We show that this contradicts Hypothesis H.

So suppose A is quick, and fix $\varepsilon > 0$. Let $j > m = \log \frac{2i}{\varepsilon}$. Since A is quick, there is an oracle Turing machine D_e such that $D_e : \mathrm{K}^j(\emptyset) \leq_{\mathrm{T}}^p A$ in time n^i. We will show that D_e can be used to get $\mathrm{K}^j(\emptyset) \leq_T^{\mathrm{P}/2^{n^\varepsilon}} \mathrm{K}^m(\emptyset)$.

Consider an input x to D_e. On input x the reduction D_e can only make queries q of length less than n^i where $n = |x|$. So if $q = 0^{(|y|+2)^{2^k}} 1y$ (and only queries of this form are interesting) we know:

$$k \leq \log\log n - \log\log(|y|+2) + \log i, \tag{1}$$

in particular, $k < \log\log n + \log i$. This leads us to define the sequence of sets S_n as follows:

$$S_n = \{\langle y, k\rangle \ : \ |y| < n^{\varepsilon/2} \ \&\ y \in \mathrm{K}^k(\emptyset)\}$$

Simulate $D_e^{(\cdot)}(x)$ and assume access to $S_{|x|}$ and to $\mathrm{K}^m(\emptyset)$. Whenever the computation tries to make a query q to the oracle, do the following:

1. If q does not have the form $0^{(|y|+2)^{2^k}}1y$ for some y and k, answer the query negatively, else fix y and k such that $q = 0^{(|y|+2)^{2^k}}1y$.
2. If $|y| < n^{\varepsilon/2}$ use S to decide whether $y \in \mathrm{K}^k(\emptyset)$ and answer the query to q accordingly (note that $k < \log\log n + \log i$ since $|q| < n^i$).
3. If $|y| \geq n^{\varepsilon/2}$, then using inequality (1),

$$\begin{aligned} k &\leq \log\log n - \log\log(|y|+2) + \log i \\ &\leq \log\frac{2i}{\varepsilon} = m. \end{aligned}$$

   So we can use the $\mathrm{K}^m(\emptyset)$ oracle to decide $y \in \mathrm{K}^k(\emptyset)$.

Thus, D_e can be simulated making use only of S_n and $\mathrm{K}^m(\emptyset)$. Furthermore, since $\langle y, k\rangle \in S_n$ implies that $|y| < n^\varepsilon$ and $k < \log\log(n) + i$, we can code the information about S_n into a string of length 2^{n^ε}. So from our assumption that A is quick it follows that

$$(\forall\varepsilon > 0)(\exists m)(\forall j > m)\ \mathrm{K}^j(\emptyset) \leq_T^{\mathrm{DTIME}(n^{i+1})/2^{n^\varepsilon}} \mathrm{K}^m(\emptyset)$$

and therefore that H fails.

To this point we have not concerned ourselves with the complexity of A. The proof just given can, however, be improved to get the upper bound required by the statement of the theorem. Given any nice f, we can obtain a slow uniform upper bound for PH which is in $\mathrm{ATIME}(\text{poly}, f(n))$ as follows. Let $h(k)$ be the smallest j for which $f(j) > k$. We will use h for additional padding:

$$A = \{0^{(|x|+2)^{2^k}+h(k)}1x : x \in \mathrm{K}^k(\emptyset)\}.$$

Then if y is a string of length n in A, we know that $h(k) < n$, and hence $f(n) \geq f(h(k)) > k$. Hence A lies in $\mathrm{ATIME}(\text{poly}, f(n))$. The rest of the proof needs only minor adjustments. This yields the result as stated. □

4.3 Fixed Points of the NP-jump

In constructing a p-HYP we must avoid fixed points of the NP-jump. We show that every fixed point of the NP-jump is a uniform upper bound for PH and that fixed points exist which are very unlikely (even more unlikely than the examples after Lemma 4.3) to be PSPACE-complete. Thus it really is necessary to actively avoid them in our construction.

Lemma 4.7 *If* $\mathrm{K}(A) \leq_{\mathrm{T}}^{p} A$ *then* A *is a uniform upper bound for* PH *(in fact for* PH^{A}*).* □

Proof. Fix the reduction D_e from $\mathrm{K}(A)$ to A i.e. $\mathrm{K}(A)(x) = D_e^A(x)$. We assume that our enumeration of nondeterministic OTMs is nice in that the jump and composition of machines is effective, i.e., that there are two computable functions jump and comp such that

- if $B \leq_{\mathrm{T}}^{p} A$ via D_i, then $\mathrm{K}(B) \leq_{\mathrm{T}}^{p} \mathrm{K}(A)$ via $D_{\mathrm{jump}(i)}$, and
- if $B \leq_{\mathrm{T}}^{p} C$ via D_i and $C \leq_{\mathrm{T}}^{p} D$ via D_j, then $B \leq_{\mathrm{T}}^{p} D$ via $D_{\mathrm{comp}(i,j)}$.

Now by definition $\mathrm{K}^1(A) \leq_{\mathrm{T}}^{p} A$ say via D_e. Then we can prove by induction that $\mathrm{K}^{k+1}(A) \leq_{\mathrm{T}}^{p} \mathrm{K}^k(A)$ via $D_{\mathrm{jump}^{(k)}(e)}$. Using comp we can define a computable function f such that $\mathrm{K}^k(A) \leq_{\mathrm{T}}^{p} A$ via $D_{f(k)}$ which proves that A is a uniform upper bound of PH^A. Starting with $\mathrm{K}^1(\emptyset)$ instead of $\mathrm{K}^1(A)$ will yield the same result for PH (without making any additional assumptions). □

Remarks.

1. If we start with the stronger assumption that $\mathrm{K}(A) \leq_{m}^{p} A$, then A will be a uniform upper bound for PH with regard to m-reductions. The necessary adjustments in the proof are straightforward.
2. If $\mathrm{K}(A) \leq_{\mathrm{T}}^{p} A$ is witnessed by a reduction which runs in linear time, then the uniformity in the above proof, together with the fact that jump and comp are also computable in linear time, yields that A is hard for $\mathrm{ATIME}(\text{poly}, \sqrt{\log n})$. This means (Lemma 4.1) that A is quick, and not slow.

Lemma 4.8 $\mathrm{ATIME}(\textit{poly}, \log\log n + O(1))$ *is closed under Turing reductions and the* NP-jump *and has a complete set in* $\mathrm{ATIME}(\textit{poly}, \log\log n)$. □

Proof. For every i there is a constant c such that $\log\log n^i \leq \log\log n + c$. Hence $\mathrm{ATIME}(\text{poly}, \log\log n + O(1))$ is closed under Turing reductions, and the set

$$B = \{\langle e, x, 0^t\rangle : \text{ the } e^{th} \text{ alternating Turing machine halts in } t \text{ steps on input } x \text{ with at most } \log\log t \text{ alternations}\}$$

is hard for this class and lies in $\mathrm{ATIME}(\text{poly}, \log\log n)$. Since $\mathrm{ATIME}(\text{poly}, \log\log n + O(1))$ is closed under adding a constant number of alternations it is certainly closed under the NP-jump. □

In particular we note:

Corollary 4.9 *There is a fixed point of the* NP-jump *in* $\mathrm{ATIME}(\textit{poly}, \log\log n)$, *which is not* PSPACE*-complete, unless* $\mathrm{PSPACE} = \mathrm{ATIME}(\textit{poly}, \log\log n + O(1))$.
□

5 The Structure of Hyper-Polynomial Hierarchies

In Section 3 we gave some indication of how far apart PSPACE is from PH by building an image H of $\mathcal{O}$ in the PSPACE degrees that respects the NP-jump operator and upper bounds, and (assuming PH is infinite) has no NP-jump fixed points. Furthermore, it is evident from our construction of H that H_a is a quick uniform upper bound for every $a \in \mathcal{O}$ with $\|a\| \geq \omega$. Section 4 showed how every quick uniform upper bound can compute the jump of another quick uniform upper bound. Combining these results it is possible to give further evidence of the richness of the quick uniform upper bounds with respect to the NP-jump. We iterate our construction in Section 4 to construct a proper, "upside-down" image of $\mathcal{O}$ in the quick uniform upper bounds below any given quick uniform upper bound. That is,

Theorem 5.1 *Given any quick* $A \in \mathrm{PSPACE}$ *which is a uniform upper bound for* PH*, there is a* $J \in \mathrm{PSPACE}$ *such that (letting* $J_a(x) = J(a, x)$*)*

1. *for all* $a \in \mathcal{O}$*,* J_a *is a quick uniform upper bound for* PH *and if* PH *separates, then* PH^{J_a} *separates,*

2. $J_\epsilon \leq^p_{\mathrm{T}} A$*,*

3. *for all* $a \in \mathcal{O}$*,* $\mathrm{K}(J_{\mathrm{succ}(a)}) \leq^p_{\mathrm{T}} J_a$ *via a reduction found effectively in* a*,*

4. *for all* e *such that* $\lim(e) \in \mathcal{O}$, $J_{\lim(e)}$ *is a uniform* lower *bound for* $\{J_a \mid a <_{\mathcal{O}} \lim(e)\}$, *and*

5. *if* PH *is infinite, then for all* $a \in \mathcal{O}$, $\mathrm{K}(J_a) \not\leq^p_{\mathrm{T}} J_a$, *so the embedding is proper in the ptime degrees.*

Within this upside-down p-HYP, it is also possible to place a (right-side up) p-HYP starting with any $J_{\lim(e)}$ and completely below all the J_a for $a <_{\mathcal{O}} \lim(e)$.
Proof Sketch. [4]We define a hierarchy of padding functions so that successor functions satisfy the conditions in Lemma 4.3 with respect to their predecessors, and limit functions dominate all previous padding functions. This by itself is straightforward, but there are a few extra wrinkles that we must smooth out. All our padding functions must be nice, to satisfy the conditions in Lemma 4.3, and must give rise to *good* sets (i.e., sets over which PH separates) to ensure a properly descending hierarchy of degrees.

Let $A \in \mathrm{PSPACE}$ be a quick uniform upper bound for PH and fix an $i \in \Sigma^*$ and a computable f such that for all k, $K^k(\emptyset) = D^A_{\langle i, f(k)\rangle}$. By taking a padded version of A and choosing a new f, we can assume that $i = 1$. Hence, by Lemma 2.6 there is a computable g such that $\mathrm{PAD}(g(k)) \leq^p_m A$, and this fact remains true if we increase the padding. Now let e_0 be least such that $\varphi_{e_0}(k) = 0^k = \mathrm{succ}^k(\epsilon)$ for all $k \in \omega$. This implies that $\lim(e_0) \in \mathcal{O}$, $\|\lim(e_0)\| = \omega$, and $H_{\lim(e_0)} = \mathrm{PAD}(s(k))$, where $s(y)$ is the function corresponding to $H_{\lim(e_0)}$, c.f. Section 3. So we know that $\mathrm{PH}^{\mathrm{PAD}(s(k))}$ is still infinite, and $\mathrm{PAD}(s(k)) \leq^p_m A$ and is a good quick uniform upper bound. Thus, $\mathrm{PAD}(s(k))$ is a good candidate for J_ϵ, but we must be careful to construct things uniformly.

In the proof of Theorem 3.1, we actually have much leeway in defining the function $s = s(k)$ for $H_{\lim(e_0)}$: we can pick s to be fully time-constructible and monotone (i.e., nice), and in addition, s can dominate any given computable function *uniformly*. Our construction of J depends on this observation, which we state without proof. (The extra parameter a in the lemma will be used later on.)

Lemma 5.2 *There is a deterministic Turing machine* T *that on input* e, a, x *runs in time* $S(e, a, x) = S_e(a, x)$*, where* S *is a computable partial function satisfying the following properties for every* $e, a \in \Sigma^*$ *such that* φ_e *is total:*

[4]*Randy/Marcus/Steve: it may be that we can get an equally rich ordering of the quick uniform upper bounds just by considering various ATIME*$(poly, f(n))$ *classes. Questions to be answer here: If* A *is in (complete for) ATIME*$(poly, f(n))$*, does this mean that* $\mathrm{K}(A)$ *is in (complete for) ATIME*$(poly, f(n)+1)$*? If* PH *separates, is ATIME*$(poly,$
$\log\log n + O(1))$ *the least class closed under the* NP*-jump and m-reductions and possessing an m-complete set?*

1. *$\lambda k.S_e(a,k)$ is total and monotone nondecreasing, and $\varphi_e(a,k) \leq S_e(a,k)$ for all k.*

2. *$\lambda k.S_e(a,k)$ satisfies the conditions for s in the construction of H_{e_0}, that is, $\mathrm{PAD}(s(k) = S_e(a,k))$ is good.*

By the existence of the machine T, $S(e,a,x)$ is fully time-constructible. $\square$

To define J, we first define a transfinite hierarchy (using $\mathcal{O}$) of nice, quickly growing padding functions. (Recall the definition of $g(k)$, above.)

Lemma 5.3 *There is computable partial function $\psi(a,k)$ such that, for all $a \in \mathcal{O}$ and $k \in \Sigma^*$,*

1. *$\lambda k.\psi(a,k)$ is total and nice, and $\mathrm{PAD}(\psi(a,k))$ is a good quick uniform upper bound,*

2. *$g(k) \leq \psi(\epsilon,k)$,*

3. *$\psi(\mathrm{succ}(a),k) \geq \psi(a,k+1)$, and*

4. *if $a = \lim(e)$, then*
$$\psi(a,k) \geq \max_{i \leq k} \psi(\varphi_e(i),k).$$

Proof of Lemma 5.3. Using the s-m-n theorem, define

$$\varphi_{f(e)}(a,k) = \begin{cases} g(k) & \text{if } a = \epsilon, \\ S_e(b,k+1) & \text{if } a = \mathrm{succ}(b), \\ \max_{i \leq k} S_e(\varphi_d(i),k) & \text{if } a = \lim(d). \end{cases}$$

By the recursion theorem, there is a c with $\varphi_c = \varphi_{f(c)}$. We set $\psi = S_c$. $\psi(a,k)\!\downarrow$ for all $a \in \mathcal{O}$ and all k by transfinite induction. The properties of ψ are ensured by Lemma 5.2 and transfinite induction on the definition of ψ. Furthermore, notice that ψ itself is fully time-constructible. $\square$

We now define J by the equation $J_a = \mathrm{PAD}(\psi(a,k))$. Since ψ was constructed in a way that allows us to use Lemma 4.3 uniformly, J will satisfy all its advertised properties. To see this we observe some critical facts about the proof of Lemma 4.3. Everything about the proof is effectively uniform. For example, there is a computable function u such that, if $\mathrm{PAD}(\varphi_e(k))$ is a degree 0 ptime padding array, then it is a quick uniform upper bound via $\varphi_{u(e)}$.[5] Likewise, there is a computable j such that, if $\mathrm{PAD}(\varphi_{e_1}(k))$ is a degree 0 ptime padding array

[5] *Are we sure this is correct?*

and $\varphi_{e_2}(k) \geq \varphi_{e_1}(k+1)$ for all k, then $\mathrm{K}(\mathrm{PAD}(\varphi_{e_2}(k))) \leq^p_{\mathrm{T}} \mathrm{PAD}(\varphi_{e_1}(k))$ via $D_{j(e_1,e_2)}$. Finally, there is a computable ℓ such that, if $\mathrm{PAD}(\varphi_{e_1}(k))$ is a degree 0 ptime padding array and $\varphi_{e_2}(k) \geq \varphi_{e_1}(k)$ for all $k \geq N$, then $\mathrm{PAD}(\varphi_{e_2}(k)) \leq^p_{\mathrm{T}} \mathrm{PAD}(\varphi_{e_1}(k))$ via $D_{\ell(e_1,e_2,N)}$. $\square$

6 Open Questions

In continuing to investigate the world between PH and PSPACE, there remain many unanswered questions, especially about those languages which are "just above" PH or "just below" PSPACE, and the location of well-known, natural languages in this spectrum.

1. Under what plausible assumptions (if any) is there a uniform upper bound for PH that cannot compute the jump of any other uniform upper bound? (It must be slow.)

   These are languages which are outside PH, have enough resources to provide easy computation of PH, but not much more (not a jump more). Such languages, if part of a p-HYP would have to sit at level ω.

   More generally, one can ask: Are there any ordinals α and languages L such that L is at level α in some p-HYP, but is not at level $\beta > \alpha$ for any p-HYP?

2. Which languages are hyper-polynomial?

   Since there is no canonical notion of *the* hyper-polynomial hierarchy, it does not make sense to define the hyper-polynomial languages to be those languages which are at or below some level of it (as one defines the hyperarithmetic sets). However, we could call a language A hyper-polynomial if there is some p-HYP, H, and some $a \in \mathcal{O}$ such that $A \leq^p_{\mathrm{T}} H_a$. This says that A is far (more than ω_1^{CK} NP-jumps) from being PSPACE-complete. The PSPACE-complete languages are not hyper-polynomial under this definition. Are there any other languages in PSPACE which are not hyper-polynomial? These languages would in some sense be much harder than PH yet still not PSPACE-complete. En route to answering this question, the following pair of questions arises:

   (a) Can a p-HYP be placed above any PSPACE set which is not within a finite number of jumps from PSPACE-complete?

   (b) Is there an A with $\mathrm{P}^A \neq \mathrm{NP}^A = \mathrm{PSPACE}$?

3. Where are the PP-complete languages in this scheme?

A careful look at the exponents on the polynomials in the proof of Toda's Theorem suggests that PP-complete languages might not be uniformly hard for any ptime unbounded alternation class. This would make them slow uniform upper bounds for PH, and indicate that PP is only very slightly larger than PH. Can this be made precise using hyper-polynomial hierarchies?

4. One of our primary motivations for this work is the logical theory of the hyperarithmetic sets which provides a well-developed link between the arithmetic sets and the analytic sets of integers [Sac90]. In this generalization of classical recursion theory, admissible recursion theory, the hyperarithmetic sets play the role of the computable sets and Σ_1^1 corresponds to computably enumerable. We have only begun to explore the resource bounded theory in this work and there remains much to be done. We mention here a few key aspects of this research.

   The well-known theorem of Spector and Markwald [Spe55], [Mar54] shows that ω_1^{CK} is the supremum of the lengths of all recursive well-orderings of the integers. A careful proof of this result can be made to yield this same fact for ptime well-orderings of the integers (that is $\omega_1^{\mathrm{CK}} = \omega_1^p$). There is a quite natural (though not fully invariant) alternating quantifier characterization of levels of the hyperarithmetic hierarchy. What is the corresponding characterization of alternations of polynomial-bounded quantifiers? The crowning result of basic hyperarithmetic theory is the theorem of Kleene that $\Delta_1^1 = \mathrm{HYP}$ (the sets Turing reducible to some level of the hyperarithmetic hierarchy). What corresponds to Δ_1^1 in this setting? PSPACE (less the complete languages) seems the obvious and most reasonable candidate, but see 2(b) above.

We believe continued work in this area will result in further insights into the complexity of hard combinatorial problems in PSPACE.

From rpruim@cs.bu.edu Thu Sep 18 13:32:23 1997 Date: Thu, 18 Sep 1997 13:24:46 -0400 (EDT) From: Randall Pruim ¡rpruim@cs.bu.edu¿ To: Steve Homer ¡homer@cs.bu.edu¿ Subject: hyp.bbl

References

[AS89] K. Ambos-Spies. On the relative complexity of hard problems for complexity classes without complete problems. *Theoretical Computer Science*, 63:43–61, 1989.

[BDG88] J. L. Balcázar, J. Díaz, and J. Gabarró. *Structural Complexity I*, volume 11 of *EATCS Monographs on Theoretical Computer Science*. Springer-Verlag, 1988.

[CKS81] A. Chandra, D. Kozen, and L. Stockmeyer. Alternation. *Journal of the ACM*, 28(1):114–133, 1981.

[Fen95] S. A. Fenner. Inverting the Turing jump in complexity theory. In *Proceedings of the 10th IEEE Structure in Complexity Theory Conference*, pages 102–110, 1995.

[Lad75] R. Ladner. On the structure of polynomial-time reducibility. *Journal of the ACM*, 22:155–171, 1975.

[Mar54] Markwald. Zur theorie der konstruktiven wohlordnungen. *Mathematiche Annalen*, 127:135–149, 1954.

[Rog67] H. Rogers. *Theory of Recursive Functions and Effective Computability*. McGraw-Hill, 1967. Reprinted. MIT Press. 1987.

[Sac90] G. E. Sacks. *Higher Recursion Theory*. Springer-Verlag, 1990.

[Spe55] C. Spector. Recursive well-orderings. *Journal of Symbolic Logic*, 20:551–563, 1955.

[Sto77] L. Stockmeyer. The polynomial-time hierarchy. *Theoretical Computer Science*, 3:1–22, 1977.

[Tod91] S. Toda. PP is as hard as the polynomial-time hierarchy. *SIAM Journal on Computing*, 20(5):865–877, 1991.

– Randall J. Pruim Computer Science Dept, Boston University ——————
——————- Quidquid latine dictum sit, altum videtur.