ELSEVIER

Theoretical Computer Science 262 (2001) 241–256

Theoretical Computer Science

www.elsevier.com/locate/tcs

Hyper-polynomial hierarchies and the polynomial jump

Stephen Fenner[a, 1], Steven Homer[b,*,2], Randall Pruim[c], Marcus Schaefer[d, 3]

[a] *Computer Science Department, University of South Carolina, Columbia, SC 29208, USA*
[b] *Computer Science Department, Boston University, 111 Cummington Street, Boston, MA 02215, USA*
[c] *Department of Mathematics and Statistics, Calvin College, Grand Rapids, MI 49546, USA*
[d] *Computer Science Department, DePaul University, Chicago, IL 60604, USA*

Received 25 December 1997; revised 17 December 1999; accepted 27 April 2000
Communicated by O. Watanabe

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} in PSPACE such that successors are polynomial jumps of their predecessors. Moreover these ptime degrees are all uniformly hard for PH. This is analogous to the hyperarithmetic hierarchy, which is defined similarly but with the (computable) 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 polynomial jump operator on the languages which are in PSPACE but not in PH.

Keywords: Polynomial hierarchy; Polynomial jump; Hyper-polynomial hierarchy; Sequences of degrees

1. Introduction

Since its definition in 1976 [13], the polynomial hierarchy has been used to classify and measure the complexity of infeasible combinatorial problems. It has been hugely

* Corresponding author.
E-mail addresses: fenner@cs.sc.edu (S. Fenner), homer@cs.bu.edu (S. Homer), rpruim@calvin.edu (R. Pruim), mschaefer@cs.depaul.edu (M. Schaefer).

[1] Supported in part by the NSF under grants CCR 92-09833 and CCR 95-01794.
[2] Supported in part by the NSF under grant NSF-CCR-9400229 and by the Mathematical Institute, Oxford University.
[3] Supported in part by the NSF under grant CCR 95-01794. This work was done while visiting Steve Fenner at the University of Southern Maine.

PII: S0304-3975(00)00193-6

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 a 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 polynomial jump operator.

The polynomial jump has proven to be a fundamental and useful tool in complexity theory. It is a complexity–theoretic version of the Turing jump in recursion theory. For a set A, $\mathrm{K}(A)$, the polynomial jump of A is defined to be the canonical NP^A complete set. This jump operator was first considered by Uwe Schöning [9, 10] where he used the polynomial jump operator to define and study (high and low) hierarchies of sets within NP. The properties of the polynomial jump were extensively studied by Mike Townsend [15, 16] who also examined the iterated polynomial jump operator $\mathrm{K}^n(A)$, for integer n. More recently, Fenner [4], motivated by jump inversion results from recursion theory (see [11, Chapter VI]), considered the problem of determining the range of the polynomial jump operator.

The polynomial 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. 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. (See, for example, [7, 8].) 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 polynomial jump operator and study their properties and the properties of the polynomial jump operator in the realm between PH and PSPACE.

Assuming the polynomial hierarchy is infinite, Ambos-Spies [1] 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, *polynomial-jump-respecting* hierarchies of length ω_1^{CK} (the first non-constructive ordinal) 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 PSPACE–PH, but that PSPACE is in some sense "very far" from PH, since not even ω_1^{CK} many polynomial 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.

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 noted by Ambos-Spies [1], 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 [14] indicates that the PP-complete languages may fit this description.

Outline. After providing the necessary background on constructive ordinals and uniform upper bounds in Section 2, we construct in Section 3 an infinite hierarchy of languages of length ω_1^{CK} in PSPACE above PH. This hierarchy is proper provided that PH doesn't collapse. In Section 4 we investigate the extent to which such a hierarchy is or is not canonical by asking where within PSPACE–PH such a hierarchy can be placed. This investigation leads to the differentiation between two types of uniform upper bounds, *slow* and *fast*. Finally, in Section 5 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 dyadic 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 $\langle\varepsilon,\varepsilon\rangle = \varepsilon$, and $\langle x, y\rangle > y$ for all $x \neq \varepsilon$.

We fix a standard, acceptable enumeration $N_0, N_1, N_2, \ldots$ of non-deterministic 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. Often we will abuse notation and associate with a set (language) its characteristic function. Thus $x \in L$ if and only if $L(x) := \chi_L(x) = 1$. We write D_e: $A \leqslant_r B$ if $D_e^B(x) = A(x)$ for all x and D_e accesses the oracle B only in a manner allowed by the reduction type $\leqslant_r$. For the most part we are interested in $\leqslant_{\mathrm{T}}^p$- and $\leqslant_m^p$-reductions. As usual, $\varphi_0, \varphi_1, \varphi_2, \ldots$ is a standard acceptable enumeration of the computable partial functions (as in [11]).

Definition 2.1. For any set $A \subseteq \Sigma^*$, we define, in the spirit of Balcázar *et al.* [2],

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

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

Many fundamental properties of the polynomial jump can be found in the work of Mike Townsend [15, 16]. It is easy to check that $\mathrm{K}(\cdot)$ lifts to a well-defined operator on the $\leqslant_{\mathrm{T}}^p$ degrees. We denote by $\mathrm{K}^k(\cdot)$ the k-fold iteration of $\mathrm{K}(\cdot)$. Let Q be an alternating oracle Turing machine such that for all A and k, $\lambda x.Q^A(0^k,x)=\mathrm{K}^k(A)$, 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 k, Q_R runs in polynominal time for all oracles, and for any oracle A, $Q^A(0^k,x)$ only makes oracle queries of length $\leqslant|x|$.

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 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 [8], but see also [7]. 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. $\varepsilon\in\mathcal{O}$.
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. $\|\varepsilon\|=0$.
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 $H_a\leqslant_{\mathrm{T}}^p H_b$, and $H_{\mathrm{succ}(a)}=\mathrm{K}(H_a)\not\leqslant_{\mathrm{T}}^p H_a$. (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 uniformly enumerable sequence of sets which is the least uniform $\leqslant_{\mathrm{T}}$-upper bound (in fact, the least uniform $\leqslant_m$-upper bound) for the sequence. In complexity theory this is not possible, since there is no least uniform $\leqslant_{\mathrm{T}}^p$-upper bound [1, 5]. 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 $\mathscr{C} = \{L_i \mid i \in \omega\}$ of languages, a *uniform* $\leqslant_r$*-upper bound* for $\mathscr{C}$ is a language H such that there is a computable function $f : \omega \to \omega$ with the property that for all i, $D_{f(i)}$: $L_i \leqslant_r H$.

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 $\{\mathrm{K}^i(\emptyset) \mid i \in \omega\}$. Since for any $a \in \mathcal{O}$, $\{b \in \mathcal{O} \mid b <_{\mathcal{O}} a\}$ is computably enumerable in a, it also makes sense to talk about uniform upper bounds for collections indexed by $\{b \in \mathcal{O} \mid b <_{\mathcal{O}} a\}$.

Definition 2.3. For any computable function $f : \omega \times \omega \to \omega$ and any countable collection of languages $\mathscr{C} = \{L_i \mid i \in \omega\}$, 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 are constants d and C such that for all k, $f(k,x) < C|x|^d$ 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 $\mathscr{C} = \{L_i \mid i \in \omega\}$ via f then A is a uniform $\leqslant_m^p$-upper bound for $\mathscr{C}$.*

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. □

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 $O(n + f(n))$ steps.

Lemma 2.6. *Let A be a uniform $\leqslant_r$-upper bound for $\mathscr{C} = \{L_k \mid k \in \omega\}$ via f. If B is the ptime padding array for $\mathscr{C}$ via g, and g is a nice function such that for all x, both $f(j)$ and $D^A_{f(j)}(x)$ halt in fewer than $g(j, |x|)$ steps, then $B \leqslant_r A$.*

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

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. □

Thus, in particular, every ptime padding array is a uniform $\leqslant^p_m$-upper bound for PH, and every set which is a uniform $\leqslant_r$-upper bound for PH is above some ptime padding array for PH with respect to $\leqslant_r$.

3. A hyper-polynomial hierarchy

We now come to the construction of an extended polynomial hierarchy. We show how to "embed"[4] $\langle \mathcal{O}, <_{\mathcal{O}}\rangle$ into $\langle PSPACE, \leqslant^p_T\rangle$ in such a way that successors correspond to polynomial 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) $H_\varepsilon = \emptyset$.
(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 $a \in \mathcal{O}$, $H_{\mathrm{succ}(a)} = \mathrm{K}(H_a) \not\leqslant^p_T H_a$.

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 build $H_{\lim(e)}$ 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 $\{H_{\varphi_e(y)}\colon\ y \in \omega\}$, which we do by making $H_{\lim(e)}$ a ptime padding array for $\{H_{\varphi_e(y)}\colon\ y \in \omega\}$. 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

[4] Strictly speaking, this may not be an embedding, since we will preserve comparability but not necessarily incomparability.

$\leqslant^p_{\mathrm{T}}$-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 level: $H_{\lim(e)}$ will "look like" $H_{\varphi_e(y)}$ and since $K^{k+1}(H_{\varphi_e(y)}) \not\leqslant^p_{\mathrm{T}} H_{\varphi_e(y)}$, D_i will eventually make a mistake. The particular "delayed diagonalization" strategy employed here is similar to those used by Ambos-Spies [1], which in turn are based on well-known techniques of Ladner [5].

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 and Q is an alternating oracle Turing machine such that for all A and k, $\lambda x.Q^A(0^k,x) = \mathrm{K}^k(A)$, the canonical $(\Sigma^{\mathrm{p}}_k)^A$-complete set, and write $Q^A_k(x)$ for $Q^A(0^k,x)$. We assume without loss of generality that Q has been chosen so that for any k, Q_R runs in polynominal time for all oracles, and for any oracle A, $Q^A(0^k,x)$ only makes oracle queries of length $\leqslant|x|$. 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. $H_\varepsilon(x) = 0$ (thus $H_\varepsilon = \emptyset$).
2. $H_{\mathrm{succ}(a)}(x) = Q^{H_a}_1(x)$ (thus $H_{\mathrm{succ}(a)} = \mathrm{K}(H_a)$).
3. $H_{\lim(e)}(\langle y,z\rangle) = 0$, unless z is of the form 0^s1v, 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 $\leqslant s$ steps, and
   (b) there is "sufficient evidence" that

$$(\forall k)\ Q^{H_{\lim(e)}}_{k+1} \not\leqslant^p_{\mathrm{T}} Q^{H_{\lim(e)}}_k.$$

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

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

   If such a least s exists and $z = 0^s1v$ for some $v \in \Sigma^*$, then we let

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

In this way, $\lambda z.H_{\lim(e)}(\langle y,z\rangle)$ will be a padded version of $H_{\varphi_e(y)}$.
It is important to observe that the value of s is 3(b) above only depends on y and not on z.

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 and $a \in \theta$, 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: $a = \varepsilon$. Property (P4) holds vacuously. Since $H_\varepsilon = \emptyset$, 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 $\{H_{\varphi_e(y)}\}_{y \in \Sigma^*}$. This suffices, because for any $b <_{\mathcal{O}} \lim(e)$, there is a y such that $b <_{\mathcal{O}} \varphi_e(y)$ and hence

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

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 $y' \geqslant y$ and all z, so we never code $H_{\varphi_e(y')}$ into $H_{\lim(e)}$. This makes $H_{\lim(e)} \equiv_{\mathrm{T}}^{p} H_{\varphi_e(y-1)}$, 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 $H_{\varphi_e(y)} \leqslant_m^{p} H_{\lim(e)}$ 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. □

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 $\leqslant_r$-upper bound A for $\mathscr{C} = \{L_i \mid i \in \omega\}$ *quick* if there is a polynomial $p(n)$ such that for each L_i there is a $\leqslant_r$-reduction $D_j : L_i \leqslant_r A$ that runs in time $p(n)$, where n is the length of the input to the reductions D_j. A uniform $\leqslant_r$-upper bound that 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 [3]. For this we define the class $\Sigma^p_{f(n)}$ to consist of all languages accepted by an alternating Turing machine in polynomial time and at most $f(n)-1$ alternations ($f(n)$ blocks), beginning in an existential state. (Note that this really is $f(n)-1$ alternations and not $O(f(n))$.)

Lemma 4.1. *A set A is a quick uniform $\leqslant_r$-upper bound for* PH *if and only if it is $\leqslant_r$-hard for $\Sigma^p_{f(n)}$ for some nice f.*

Proof. If A is a quick uniform $\leqslant_r$-upper bound for PH, then there is a computable function g and a polynomial p such that $D_{g(j)}$: $\mathrm{K}^j(\emptyset) \leqslant_r A$ 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 $\leqslant_r$-hard for $\Sigma^p_{f(n)}$ for some f as above. Consider the set $B = \{\langle j, 0^l, y\rangle : y \in \mathrm{K}^j(\emptyset) \text{ for some } j < f(l)\}$. $B \in \Sigma^p_{f(n)}$, so $B \leqslant_r A$. On the other hand, $\mathrm{K}^j(\emptyset) \leqslant^p_m B$. We just have to make sure that there is a fixed time bound on the running time of the reductions independent of j. Let the function g on input j first compute the smallest l such that $j < f(l)$. It then computes the index of a reduction $R_{g(j)}$ that for inputs x with $|x| < l$ in one step returns the right answer (we hardwire the answer into the reduction). If $|x| > l$ the reduction writes $\langle j, 0^l, x\rangle$ on the query-tape and queries B. The reduction $R_{g(j)}$ runs in time $c(|x|+1)$ for some constant c not depending on j. Composing the reduction with the Turing-reduction from B to A we get a family $(R_{g(j)})_{j\in\omega}$ of reductions all of which run in time $p(n)$ for some polynomial $p(n)$. □

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 $\mathrm{K}(B) \leqslant^p_{\mathrm{T}} A$.*

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 either*
1. *$f(k+1, m) \leqslant p(g(k, 2) + g(0, m))$, or*

2. *$f(k+1,m) \leqslant p(g(k,m))$ and $g(k,m) < \mathrm{O}(m^d)$ for some constant d, then $\mathrm{K}(B) \leqslant_m^p A$.*

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 that 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 some other examples of applications of Lemma 4.3.

Example 4.4.

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 $\mathrm{K}(A) \equiv_m^p A$, so A is a fixed point of the* polynomial jump.
2. *If A is a bounded degree ptime padding array for* PH*via f, and B is the ptime padding array for* PH*via $g(k,m) = f(k+1,m)$, then $\mathrm{K}(B) \leqslant_{\mathrm{T}}^p A$.*

Proof of Example 4.4.

1. We must find polynomials p_1 and p_2 such that

$$m^e + 2^{k+1} \leqslant p_1(2^e + 2^k + m^e + 2^0))$$

and

$$m^e + 2^{2^{k+1}} \leqslant p_2(2^e + 2^{2^k} + m^e + 2^{2^0})).$$

Choosing $p_1(n) = 2n$ and $p_2(n) = n^2$ is sufficient. (This could also be demonstrated using case 2 of the lemma, since each of these examples has bounded degree.)

2. If f yields a bounded degree padding array then so does g, so this is immediate from case 2. □

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

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

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 $\bar{B} = \Sigma^* - B$.

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).
   - $n=|y|=|\langle e,x,0^t\rangle|\geqslant t,|x|$.
   - Since the length of a query is bounded by running time, if $N_e^B(x)$ makes queries to any strings q, then $|q|<t\leqslant|y|=n$.
   - 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 $g(k,m)<|q|\leqslant|y|=n$.
   - Let k be maximal such that there is a query q of type (k,m). 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 $\leqslant n$.
3. Using the observations above, we see that determining whether $N_e^B(x)$ accepts is equivalent to a formula of the form

$$\psi\equiv\exists\,\vec{p}\exists\vec{q}\,\exists\,\vec{a}$$
$$\forall i[[\![q_i\in B]\!]=a_i\,\&\,\Phi(e,x,t,p,q,a)]$$

   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 that 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 after at most t steps.
   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 $\Leftrightarrow N_e^B(x)$ accepts.
4. Let $r(y)=0^{f(k+1,|\check{\psi}|)}1\check{\psi}$.
   Notice that $m=|\check{\psi}|\leqslant O(n^3)$ and that $g(k,2)\leqslant g(k,m)\leqslant n$, since there was a query of type (k,m). Also $f(k+1,m)\leqslant f(k+1,O(n^3))\leqslant O(f(k+1,n)^3)$. So in case 1 we have $f(k+1,m)\leqslant O((p(g(k,2)+g(0,n)))^3)\leqslant O(p(n+n^d)^3)$, where d is the degree of $g(0,*)$, and in case 2 we have $f(k+1,m)\leqslant O((p(g(k,n)))^3)\leqslant O(p(n^d)^3)$, where d is an upper bound on the degree of $g(k,*)$ for every k. Therefore, $r(y)$ can be computed in time polynomial in n. □

4.2. Slow uniform upper bounds

We will now construct a set A that 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.

Hypothesis H. The polynomial hierarchy does not collapse even with access to a set computable in subexponential space in the following sense:

$$(\exists \varepsilon > 0)\ (\forall S \in \mathrm{DSPACE}(n^{\varepsilon}))\ (\forall m)(\exists j)\mathrm{K}^j(\emptyset) \not\leqslant_{\mathrm{T}}^{p} \mathrm{K}^m(\emptyset) \oplus S.$$

The hypothesis says that for some $\varepsilon > 0$ sets in $\mathrm{DSPACE}(n^{\varepsilon})$ will not collapse the hierarchy, namely for any given level of the hierarchy and a set S computable in space n^{ε} there will be a higher level that will not reduce to the lower level even with access to S.

Theorem 4.5. *If f is nice, then $\Sigma_{f(n)}^{p}$ contains a slow uniform $\leqslant_{\mathrm{T}}^{p}$-upper bound for* PH, *unless H fails.*

Lemma 4.1 allows us to give a more striking version of the theorem which does not refer to slowness at all.

Corollary 4.6. *If f is nice, then $\Sigma_{f(n)}^{p}$ contains a uniform $\leqslant_{\mathrm{T}}^{p}$-upper bound for* PH *that is not hard for any $\Sigma_{g(n)}^{p}$ (with g nice), unless H fails.*

Proof of Theorem 4.5. Let f be a nice function. Define $h(k)$ as the smallest j for which $f(j) > k$. We define A and verify the properties we claimed.

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

From the definition of A it is clear that A is a uniform $\leqslant_{\mathrm{T}}^{p}$-upper bound for PH. If $x = 0^{(|y|+2)^{2^k}+h(k)}1y$ is a string of length n in A, we know that $h(k) < n$, and hence $f(n) \geqslant f(h(k)) > k$. Therefore, A lies in $\Sigma_{f(n)}^{p}$.

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 $j > m = \log(2i/\varepsilon)$. Since A is quick, there is an oracle Turing machine D_e such that $D_e : \mathrm{K}^j(\emptyset) \leqslant_{\mathrm{T}}^{p} A$ in time n^i. We will show that D_e can be used to get $\mathrm{K}^j(\emptyset) \leqslant_{\mathrm{T}}^{p} \mathrm{K}^m(\emptyset) \oplus S$ for some $S \in \mathrm{DSPACE}(n^{c\varepsilon})$ where c is a constant independent of j, m and ε.

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 $q = 0^{(|y|+2)^{2^k}+h(k)}1y$ (and only queries of this form are interesting) we know:

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

Define a set S as follows:

$$S = \{\langle 0^n, y, k\rangle : |y| < n^{\varepsilon/2}\ \&\ y \in \mathrm{K}^k(\emptyset)\}.$$

Then $S \in \mathrm{DSPACE}(n^{c\varepsilon})$ for some constant c only depending on what complete problem we use to define the polynomial-jump.

Run $D_e^{(\cdot)}(x)$ and assume access to S 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 $0^{(|y|+2)^{2^k}+h(k)}1y$ for some y and k, answer the query negatively, else fix y and k such that $q = 0^{(|y|+2)^{2^k}+h(k)}1y$.
2. If $|y| < n^{\varepsilon/2}$ use S to decide whether $y \in \mathrm{K}^k(\emptyset)$ and answer the query to q accordingly.
3. If $|y| \geqslant n^{\varepsilon/2}$, then using inequality (1),

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

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

Thus, D_e can be simulated making use only of S and $\mathrm{K}^m(\emptyset)$. So from our assumption that A is quick it follows that

$$(\forall \varepsilon > 0)(\exists S \in \mathrm{DSPACE}(n^{c\varepsilon}))\ (\exists m)\ (\forall j > m)\mathrm{K}^j(\emptyset) \leqslant_{\mathrm{T}}^p \mathrm{K}^m(\emptyset) \oplus S,$$

which implies the failure of H. □

4.3. Fixed points of the polynomial-jump

In constructing a p-HYP we must avoid fixed points of the polynomial jump. We show that every fixed point of the polynomial jump is a uniform upper bound for PH and that fixed points exist that 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* $\mathrm{K}(A) \leqslant_{\mathrm{T}}^p A$ *then* A *is a uniform* $\leqslant_{\mathrm{T}}^p$*-upper bound for* PH (*in fact for* PH^A).

Proof. We assume that our enumeration of nondeterministic OTMs has the property that there are two computable functions jump and comp such that

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

By the hypothesis of the lemma, $\mathrm{K}^1(A) \leqslant_{\mathrm{T}}^p A$. Fix a reduction D_e witnessing this fact, i.e., such that $\mathrm{K}(A)(x) = D_e^A(x)$. By induction $\mathrm{K}^{k+1}(A) \leqslant_{\mathrm{T}}^p \mathrm{K}^k(A)$ via $D_{\mathrm{jump}^{(k)}(e)}$. Using comp we can define a computable function f such that $\mathrm{K}^k(A) \leqslant_{\mathrm{T}}^p A$ 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). □

Remark.

1. If we start with the stronger assumption that $K(A) \leqslant_m^p A$, 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 $K(A) \leqslant_T^p A$ is witnessed by a reduction that 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 $\Sigma^p_{(\log n)^{1/2}}$. This means (Lemma 4.1) that A is quick, and not slow.

Lemma 4.8. *$\Sigma^p_{\log\log n+O(1)}$ is closed under Turing reductions and the* polynomial jump *and has a complete set in $\Sigma^p_{\log\log n}$.*

Proof. For every i there is a constant c such that $\log\log n^i \leqslant \log\log n + c$. Hence $\Sigma^p_{\log\log n+O(1)}$ 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 $\Sigma^p_{\log\log n}$. Since $\Sigma^p_{\log\log n+O(1)}$ is closed under adding a constant number of alternations it is certainly closed under the polynomial jump. □

In particular, we note:

Corollary 4.9. *There is a fixed point of the* polynomial jump *in $\Sigma^p_{\log\log n}$, which is not* PSPACE*-complete, unless* PSPACE $= \Sigma^p_{\log\log n+O(1)}$.

5. 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. What are the possible structures 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 polynomial

jump operator and upper bounds, and (assuming PH is infinite) has no polynomial 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 $\|a\| \geqslant \omega$. 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 polynomial jump by iterating the construction of Section 4. The iterated constructions illuminate a very complex structure of constructible hyper-polynomial hierarchies. It would be interesting to understand the limits of such constructions and to try to characterize the hyper-polynomial hierarchies.

3. 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 hyper-arithmetic 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 $A \leqslant_{\mathrm{T}}^{p} H_a$. This says that A is far (more than ω_1^{CK} polynomial jumps) from being PSPACE-complete. Assuming $\mathrm{P} \neq \mathrm{PSPACE}$, 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 $\mathrm{P}^A \neq \mathrm{NP}^A = \mathrm{PSPACE}$? (Such an A would ot be PSPACE-complete, but would be "only one jump away".)
4. 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?
5. 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 [8]. In this generalization of classical computability theory, admissible recursion theory, the hyperarithmetic sets play the role of the computable sets and Σ_1^1 corresponds to the computably enumerable sets. 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 [12, 6] shows that ω_1^{CK} is the supremum of the lengths of all computable 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.

References

[1] K. Ambos-Spies, On the relative complexity of hard problems for complexity classes without complete problems, Theoret. Comput. Sci. 63 (1989) 43–61.
[2] J.L. Balcázar, J.Díaz, J. Gabarró, Structural Complexity I, EATCS Monographs on Theoretical Computer Science, vol. 11, Springer, Berlin, 1988.
[3] A. Chandra, D. Kozen, L. Stockmeyer, Alternation, J. ACM 28 (1) (1981) 114–133.
[4] S.A. Fenner, Inverting the Turing jump in complexity theory, in: Proc. 10th IEEE Structure in Complexity Theory Conference, 1995, pp. 102–110.
[5] R. Ladner, On the structure of polynomial-time reducibility, J. ACM 22 (1975) 155–171.
[6] Markwald, Zur theorie der konstruktiven wohlordnungen, Math. Ann. 127 (1954) 135–149.
[7] H. Rogers, Theory of Recursive Functions and Effective Computability, McGraw-Hill, New York, 1967 (Reprinted. MIT Press, Cambridge, MA, 1987).
[8] G.E. Sacks, Higher Recursion Theory, Springer, Berlin, 1990.
[9] U. Schöning, A note on complete sets for the polynomial-time hierarchy, SIGACT News, June, 1980.
[10] U. Schöning, A low and high hierarchy within NP, J. Comput. Systems Sci. 27 (1993) 14–28.
[11] R.I. Soare, Recursively Enumerable Sets and Degrees, Springer, Berlin, 1987.
[12] C. Spector, Recursive well-orderings, J. Symbolic Logic 20 (1955) 551–563.
[13] L. Stockmeyer, The polynomial-time hierarchy, Theoret. Comput. Sci. 3 (1977) 1–22.
[14] S. Toda, PP is as hard as the polynomial-time hierarchy, SIAM J. Comput. 20 (5) (1991) 865–877.
[15] M. Townsend, A polynomial jump operator and complexity for type two relations, Ph.D. Thesis, University of Michigan, 1982.
[16] M. Townsend, A polynomial jump operator, Inform. and Control 68 (1986) 146–169.