Bounding the Complexity of Advice Functions

Ricard Gavaldà

Department of Software (L.S.I.)

Universitat Politècnica de Catalunya

08028 Barcelona, Spain

gavalda@lsi.upc.es

March 1993

Abstract. It was known that every set A in P/poly has an advice function in $\mathrm{PF}(\Sigma_2^{\mathrm{p}}(A))$. This paper shows that A also has an advice function in $\mathrm{PF}(\mathrm{NP}(A) \oplus \Sigma_3^{\mathrm{p}})$. From this new bound, it is shown that separating Δ_2^{p} and Σ_2^{p} relative to a set in P/poly is as hard as obtaining the same separation, unrelativized.

Suggested running title: Complexity of advice functions

This research was supported by the National Science Foundation under grant CCR89-13584 while the author was visiting the University of California at Santa Barbara, by the ESPRIT Basic Research Actions Program of the EC under contract No. 7141 (project ALCOM II), and by the DAAD and MEC through Acciones Integradas 1992, 131-B, 313-AI-e-es/zk.

1. Introduction

1.1. Statement of the problem

Complexity theory studies the resources needed to solve problems. These resources can be measured uniformly, that is, on a single device that solves every instance of the problem, or nonuniformly, that is, allowing the computing device to change with the size of the instance.

This paper deals with sets whose nonuniform complexity is low. Karp and Lipton [13] proposed a framework for the study of these sets in terms of machines that take advices. Roughly speaking, an advice for a set is a string encoding some information that allows to decide membership to the set quickly, for all strings of some fixed length. The class P/poly is the class of all sets with advices of polynomial length that allow to decide the set in polynomial time.

An *advice function* for a set A is a function $f : 0^\star \to \Sigma^\star$ such that, for every n, $f(0^n)$ is a correct advice for A on length n. Hence, every set in P/poly has some advice function whose length increases only polynomially.

(Formal definitions for all these concepts will be given in Section 2.2. Since P/poly equals the class of sets accepted by families of polynomial-size boolean circuits [20], we may assume during this discussion that advices are always encodings of circuits. That is, if f is an advice function for A, $f(0^n)$ encodes a circuit accepting exactly the strings of length n in A.)

We study the complexity of computing advice functions for sets in P/poly. However, P/poly contains sets of arbitrarily high complexity, even nonrecursive ones, and therefore their advice functions must also have arbitrarily high complexity. For this reason, we do not study the absolute complexity of these functions but that of computing advice functions for a set A in P/poly, relative to A.

We can show that this is a nontrivial problem, in the sense that even polynomial time does not always suffice. Let $\mathrm{PF}(X)$ denote the class of functions that can be computed in polynomial time relative to oracle X.

Proposition 1. There is a set A in P/poly (in fact, sparse) with no advice function in $\mathrm{PF}(A)$.

The proposition can be proved by diagonalization over all polynomial-time oracle transducers, and it also follows indirectly from the work of several authors on self-printable sets and lowness properties of sparse sets [1,8,16]. With a more complex construction, this lower bound can be improved to NPSV(A) [11], where NPSV is the class of functions computed by single-valued NP transducers.

Using the techniques of Ko and Schöning [14] and Schöning [21] one can obtain a general upper bound for the complexity of this problem. Roughly, it is always possible to compute advice functions within the functional version of relativized Δ_3^{p}.

Proposition 2 [14,21]. Every set A in P/poly has an advice function in PF$(\Sigma_2^{\mathrm{p}}(A))$.

To prove it, observe that the set of correct advices for A is in coNP(A). Close this set under prefixes, obtaining a set in $\Sigma_2^{\mathrm{p}}(A)$. Finally, use prefix search over this oracle to obtain a correct advice for a given length.

Thus, the complexity of computing advices for $A \in$ P/poly ranges between PF(A) and PF$(\Sigma_2^{\mathrm{p}}(A))$. Here we are interested in narrowing this gap. Can we always compute some advice function for A in PF(NP(A))?

1.2. Motivation

This question is interesting for two main reasons. The first one concerns the power of P/poly sets when used as oracles. The second one comes from computational learning theory and, more precisely, from the study of query learning.

The power of sets with low nonuniform complexity as oracles has been intensely studied in the last decade. In particular, whether sparse or tally sets are powerful enough to separate relativized complexity classes has been an important research theme. See [12] for a recent survey.

Two important results in this line are due to Balcázar, Book, and Schöning [9] and Long and Selman [17]. They showed that the polynomial-time hierarchy (PH) is infinite if and only if it is infinite relative to some sparse set, and if and only if it is infinite relative to every sparse set. In fact, this holds not only for sparse sets but also for the class of sets $\leq_{\mathrm{T}}^{\mathrm{P}}$-reducible to sparse sets, P/poly.

A more precise statement about each level of PH is shown in [14,21].

Theorem 3 [14,21].

(i) For every $k \geq 2$, PH $= \Sigma_k^{\mathrm{p}}$ if and only if PH$(A) = \Sigma_k^{\mathrm{p}}(A)$ for every $A \in$ P/poly.

(ii) For every $k \geq 3$, PH $= \Delta_k^{\mathrm{p}}$ if and only if PH$(A) = \Delta_k^{\mathrm{p}}(A)$ for every $A \in$ P/poly.

Hence, we cannot currently prove that, relative to some set in P/poly, PH is proper up to Σ_2^{p} or higher.

On the other hand, Baker, Gill, and Solovay [7] describe a sparse oracle S that separates NP from coNP. Thus, relative to S, PH is proper up to at least Δ_2^{p}.

This leaves a gap open at the second level of the polynomial-time hierarchy. We would like to show some set in P/poly that separates PH from Δ_2^{p}. Or, alternatively, show that the existence of such set implies $\mathrm{PH} \neq \Delta_2^{\mathrm{p}}$ (unrelativized); in other words, prove Theorem 3, part (ii), for the case $k = 2$.

For proving that the relativized separation of PH from Δ_2^{p} implies the same unrelativized separation, we need an improvement on the techniques in [14,21]. To discuss why, let us sketch the proof of Theorem 3, part (ii). Direction from right to left is trivial. To prove the implication from left to right, solve any $\mathrm{PH}(A)$ predicate as follows: compute first a correct advice for a sufficiently large initial segment of A, and give it to a non-oracle PH machine that simulates the $\mathrm{PH}(A)$ computation using the advice to decide queries to A. The first part of the computation is in $\mathrm{PF}(\Sigma_2^{\mathrm{p}}(A))$ by Proposition 2, and the second one can be solved in Δ_k^{p} assuming that $\mathrm{PH} = \Delta_k^{\mathrm{p}}$. If $k \geq 3$, this is a $\Delta_k^{\mathrm{p}}(A)$ computation.

To extend this proof to the case $k = 2$ we must be able to compute advice functions with an oracle in $\mathrm{NP}(A)$ (or at least in $\mathrm{NP}(A) \oplus \mathrm{PH}$) instead of $\Sigma_2^{\mathrm{p}}(A)$, precisely the problem we are interested in.

A more important reason for the study of this problem is given by computational learning theory. *Query learning* or *learning via queries* is currently one of the most important formal models of learning. It was introduced by Angluin [3,4] to study the process by which some learning agent infers a concept by actively asking questions about it.

In this paper we do not explain Angluin's model. The reader is referred to [3] for the basic paradigm and some examples, and to [4] for a discussion of the types of queries that may be reasonable in the design of learning algorithms. A more formal approach and some extensions to Angluin's model can be found in the recent work of Watanabe [24,25].

In order to study the complexity of learning problems, Watanabe and Gavaldà [26] have recently formulated query learning as some type of oracle computation relative to unknown oracles. In particular, they consider the notion of "bounded learning in polynomial time", as defined and discussed in [24,25]. It turns out that, for some important classes, learnability under this notion can be related to the complexity of computing advice functions for sets in P/poly.

For example, for the class of boolean circuits, learnability depends on whether such functions can be computed in a relativized version of PF(NP).

Theorem 4 [26]. If boolean circuits are learnable using the query types discussed in [4] then every set A in P/poly has an advice function in PF(NP($A[1]$)).

Here, NP($A[1]$) is a restriction of NP(A). It is defined by allowing machines of type NP to query oracle A at most once in each nondeterministic computation path.

In some cases even tighter relations are provable. By considering only circuits with a particular structure, one can define a subclass of the boolean circuits, named R_{cir}^{rep} in [26], that has the following property.

Theorem 5 [26]. R_{cir}^{rep} is learnable using the query types in [4] if and only if every set A in P/poly has an advice function in PF(NP($A[1]$)).

From these and other examples in [26] one can argue that determining the exact complexity of computing advice functions is of relevance to learning theory.

1.3. Results of this paper

The idea used to prove Proposition 2 seems to require a $\Sigma_2^{\mathrm{p}}(A)$ oracle to compute advices. We show, using a very different approach, that access of type NP to A is enough, if the oracle provides some additional but unrelativized computational power.

Theorem 6. Every set A in P/poly has an advice function in PF(NP($A[1]$) $\oplus\ \Sigma_3^{\mathrm{p}}$).

Thus, proving that the class of circuits R_{cir}^{rep} mentioned above is not learnable requires proving P $\neq$ NP or more.

Note that this theorem does not subsume the PF($\Sigma_2^{\mathrm{p}}(A)$) upper bound, because of the additional Σ_3^{p} oracle. But it gives a way to compute advice functions in PF(NP(A)) from the assumption PH $= \Delta_2^{\mathrm{p}}$.

Corollary 7. If PH $= \Delta_2^{\mathrm{p}}$, then every set A in P/poly has an advice function in PF(NP($A[1]$)).

This is all we need to prove the following theorem, using the argument sketched for Theorem 3.

Theorem 8. PH $= \Delta_2^{\mathrm{p}}$ if and only if PH(A) $= \Delta_2^{\mathrm{p}}(A)$ for every $A \in$ P/poly.

Together with Theorem 3, this characterizes precisely which levels of PH can be separated with oracles in P/poly: we can separate NP from Δ_2^{p}, but any relativized separation beyond Δ_2^{p} is as hard to prove as the same separation for unrelativized PH.

The rest of the paper is organized as follows:

- Section 2 is devoted to the proof of our main result.
- In Section 3 we discuss a related but different problem: since sets in P/poly are also those that can be $\leq_{\mathrm{T}}^{\mathrm{P}}$-reduced to sparse sets, it makes sense to study the complexity of recognizing such sparse sets relative to the original sets. We apply our proof technique to improve the known bounds on this complexity.
- In Section 4 we discuss some related work and state some open questions.

2. Proof of Theorem 6

2.1. Outline of proof

Rather than sketching the actual proof given later, it may be clearer to sketch the three ideas that lead to the proof. Consider an arbitrary set A in P/poly, and let $A^{=n}$ denote $A \cap \Sigma^n$.

First, show that A has an advice function in $\mathrm{PF}(\mathrm{NP}(A \oplus \mathrm{PP}) \oplus \Sigma_2^{\mathrm{p}}(\mathrm{PP}))$. This is done using Angluin's "majority vote strategy" developed in the context of query learning [4]: suppose we want to find an advice for $A^{=n}$. Initially, we consider all strings within a polynomial length bound as candidates to be correct advices; we use an oracle in $\mathrm{NP}(A \oplus \mathrm{PP})$ to find a string w_1 that has length n and

- either is in A but is rejected by the majority of advices,
- or is not in A but is accepted by the majority of advices.

If w_1 exists, we (implicitly) discard half of the candidate advices by keeping w_1. Repeat this procedure while possible, obtaining w_1, w_2, $\ldots$, w_m, and halving the number of remaining advices at each round. Since the initial number of candidates is exponential in n, the process must stop after polynomially many rounds. At that point, w_1, w_2, $\ldots$, w_m exactly identify $A^{=n}$: to know whether a string is in $A^{=n}$, take a majority vote among the advices that have not been discarded. Then some additional queries to an oracle in $\Sigma_2^{\mathrm{p}}(\mathrm{PP})$ provide a correct advice for $A^{=n}$.

Second, use Stockmeyer's result [23] by which we can approximate any #P function in Δ_3^{p}. With this approximate counting, each round discards a constant fraction of the advices, possibly smaller than one half, but still enough to achieve polynomial time. Thus, A has an advice function in $\mathrm{PF}(\mathrm{NP}(A \oplus \Delta_3^{\mathrm{p}}) \oplus \Sigma_2^{\mathrm{p}}(\Delta_3^{\mathrm{p}}))$, and some work shows that the $\Sigma_2^{\mathrm{p}}(\Delta_3^{\mathrm{p}}) = \Sigma_4^{\mathrm{p}}$ oracle can be replaced with a Σ_3^{p} one. Notice that with this bound we can already show that $\Delta_2^{\mathrm{p}}(A) \neq \Sigma_2^{\mathrm{p}}(A)$ implies $\mathrm{NP} \neq \mathrm{coNP}$.

Third, get rid of the Δ_3^{p} in $\mathrm{NP}(A \oplus \Delta_3^{\mathrm{p}})$, used to approximate the size of sets of advices that accept/reject a string. We observe the following: the strategy above does not really need this approximation. It is enough to know that the set of advices that accept/reject a word is "large", so that many advices are discarded at every round. As in the proof of Stockmeyer's Theorem, we can know that a set is large by checking that it cannot be hashed with a hash family of suitable size (see the definitions in Section 2.3). This turns out to be a Π_2^{p} predicate, so we must consult an oracle in $\mathrm{NP}(A \oplus \Pi_2^{\mathrm{p}})$ where Π_2^{p} is used only positively. If we manage to extract the $\forall$ quantifier implicit in Π_2^{p}, the $\exists$ one can be absorbed in $\mathrm{NP}(A)$.

We do this as follows: before each round, we precompute a hash family that hashes *all* sufficiently small sets of advices. The precomputation can be done in $\mathrm{PF}(\Sigma_3^{\mathrm{p}})$. Then, to know that the set of advices that accept/reject a guessed word is "large", it is enough to test that this particular hash family does not hash the set. And this property is in NP.

2.2. Definitions

All the languages we consider are defined over the alphabet $\Sigma = \{0, 1\}$. Let $\langle .,. \rangle$ denote any standard pairing function $\Sigma^\star \times \Sigma^\star \to \Sigma^\star$, that is naturally extended to more than two arguments.

The join of two languages A and B, denoted $A \oplus B$, is defined as $(0A) \cup (1B)$. The join of two classes $\mathcal{A}$ and $\mathcal{B}$, $\mathcal{A} \oplus \mathcal{B}$, is the class of all sets $A \oplus B$, with $A \in \mathcal{A}$ and $B \in \mathcal{B}$.

Fix two sets I and A. We say that string y is a *correct advice* for $A^{=n}$ with respect to I if, for every x of length n, $x \in A$ if and only if $\langle x, y \rangle \in I$.

A set A is in the class P/poly if there exist a set $I \in \mathrm{P}$ and a polynomial p such that for every n there is some correct advice for $A^{=n}$ with respect to I that has length at most $p(n)$. Any set $I \in \mathrm{P}$ with this property is called an *advice interpreter* for A.

An *advice function* for a set A is a function $f : 0^\star \to \Sigma^\star$ such that for some advice interpreter I for A and every n, $f(0^n)$ is a correct advice for $A^{=n}$ with respect to I.

Consider a set $A \in \mathrm{P/poly}$ and fix an advice interpreter I for A. For a string C, we say that *advice C accepts string x* if $\langle x, C \rangle \in I$. A *sample* for Σ^n is (the encoding of) a set of pairs of the form $\langle x, b \rangle$, where x has length n and $b \in \{0, 1\}$. We say that *advice C is consistent with sample l* when, for each pair $\langle x, b \rangle$ in l, C accepts x if and only if $b = 1$. We will use a sample as a compact representation of the set of advices that are consistent with it.

For a natural number n, $\log n$ denotes $\max(1, \lfloor \log_2 n \rfloor)$.

2.3. Hashing and approximation

We recall from Sipser's work [22] the definitions and properties of hashing that we will need in the proof.

Definitions [22]. 1) Let M be a $m \times p$ matrix with entries in Σ. M defines a *linear transformation* $h : \Sigma^p \to \Sigma^m$ as follows: For any string s, let s_i denote the ith bit of s. For $x \in \Sigma^p$, $h(x)$ is a string $y \in \Sigma^m$ with

$$y_i = \left(\sum_{j=1}^{p} M[i,j] \wedge x_j \right) \bmod 2, \qquad \text{for } 1 \leq i \leq m.$$

2) A *hash family of size* s is a collection of s linear transformations with the same domain and range.
3) A hash family $H = h_1, \ldots, h_s : \Sigma^p \to \Sigma^m$ *hashes* a set $X \subseteq \Sigma^p$ if for every $w \in X$ there is an $i \in \{1, \ldots, s\}$ such that for every $w' \in X$ different from w, $h_i(w) \neq h_i(w')$.

It is easy to show that small hash families cannot hash large sets.

Lemma 9 [22]. If a hash family $H = h_1, \ldots, h_s : \Sigma^p \to \Sigma^m$ hashes the set X, then $\|X\| \leq s \cdot 2^m$.

Proof. Each h_i can hash at most 2^m different words without collisions, and there are s transformations h_i. ■

Sipser showed that every set is hashed, with high probability, by a hash family whose size and range are small compared to the size of the set.

Coding Lemma [22]. Let $X \subseteq \Sigma^p$ be a set, $k \geq \|X\|$, and $m = 1 + \log k$. Then the probability that H hashes X is at least $1/2$, when the hash family $H = h_1, \ldots, h_m : \Sigma^p \to \Sigma^m$ is chosen uniformly at random.

Note that m defines both the size and the range of H.

We need an extension of the Coding Lemma: not only a particular set can be hashed but, for any collection of sets, there is some hash family with small size and range that hashes all sets simultaneously.

Lemma 10. Let $X_1, \ldots, X_{2^n} \subseteq \Sigma^p$ be a collection of 2^n sets, k be such that $k \geq \|X_i\|$ for every i, and $m = 1 + \log k$. Then the probability that H hashes every X_i is at least $1/2$, when the hash family $H = h_1, \ldots, h_{(n+1)\cdot m} : \Sigma^p \to \Sigma^m$ is chosen uniformly at random.

Proof. Split H into $n+1$ families $H_1, \ldots, H_{n+1}$ of size m. For fixed i and j, the probability that H_j does not hash X_i is at most $1/2$ by the Coding Lemma. Since all H_j are independently chosen, the probability that no H_j hashes X_i is at most $2^{-(n+1)}$, for fixed i. Then,

$$\begin{aligned}
&\text{Prob}\{\ \exists i\ (\text{no } H_j \text{ hashes } X_i)\ \}\\
&\qquad \le \sum_i \text{Prob}\{\ \text{no } H_j \text{ hashes } X_i\ \}\\
&\qquad \le 2^n \cdot 2^{-(n+1)} = 1/2.
\end{aligned}$$

Finally, if each X_i is hashed by some subfamily H_j of H it is also hashed by H itself. ∎

We will also use a theorem by Stockmeyer that allows us to approximate any #P function within the polynomial-time hierarchy.

Definition [23]. For two functions $f, g : \Sigma^\star \to \mathbb{N}$, we say that g is an $r(n)$-approximation of f if for every $x \in \Sigma^\star$

$$g(x) \cdot r(|x|) \le f(x) \le g(x)/r(|x|).$$

Theorem 11 ([23], Theorem 3.1). For every function $f \in \#\text{P}$ and every $d > 0$ there is some function $g \in \text{PF}(\Sigma_2^{\text{p}})$ that is a $(1+n^{-d})$-approximation of f.

2.4. Proof of Theorem 6

Fix a set $A \in \text{P/poly}$. Let $I \in \text{P}$ be an advice interpreter for A and q a polynomial such that every $A^{=n}$ has some advice of length at most $q(n)$. Modifying I if needed, we may assume that every $A^{=n}$ has some advice of length *exactly* $q(n)$. We describe an algorithm of type $\text{PF}(\text{NP}(A[1]) \oplus \Sigma_3^{\text{p}})$ that finds one such advice given 0^n as input.

For each n, each sample l for Σ^n, and each string $w \in \Sigma^n$, define the sets

$Cons(0^n, l)$: strings of length $q(n)$ that are consistent with l,

$Acc(0^n, l, w)$: strings in $Cons(0^n, l)$ that accept w, and

$Rej(0^n, l, w)$: strings in $Cons(0^n, l)$ that reject w.

We will be interested in finding a hash family H that hashes all sufficiently small sets of the form Acc or Rej. Define the polynomial $p(n) = n^2 \cdot q^2(n)$ and the predicate $hashes\_all(0^n, l, H)$, where l is a sample for Σ^n and H is a hash family:

```
input 0^n;
l := ∅;    (* l is a sample for Σ^n *)
repeat
(1)     find a hash family H such that hashes_all(0^n, l, H);
(2)     find any string w of length n such that
            – w ∈ A and H does not hash Rej(0^n, l, w), or
            – w ∉ A and H does not hash Acc(0^n, l, w);
        (* w is the counterexample for this iteration *)
        if such w exists then
            l := l ∪ {⟨w, χ_A(w)⟩}
        else
            exit the repeat loop
        endif
endrepeat;
(* At this point, l and H are such that for every w ∈ Σ^n, *)
(*      – if w ∈ A then H hashes Rej(0^n, l, w), and        *)
(*      – if w ∉ A then H hashes Acc(0^n, l, w)             *)
(3) find any string C in Cons(0^n, l) such that for every w ∈ Σ^n,
        – if C accepts w then H hashes Rej(0^n, l, w), and
        – if C rejects w then H hashes Acc(0^n, l, w);
output C;
```

Figure 1. An algorithm to find advices.

"for every $w \in \Sigma^n$,

– if $\|Acc(0^n, l, w)\| \leq \|Cons(0^n, l)\|/p(n)$ then H hashes $Acc(0^n, l, w)$, and

– if $\|Rej(0^n, l, w)\| \leq \|Cons(0^n, l)\|/p(n)$ then H hashes $Rej(0^n, l, w)$."

Finally, define the algorithm in Figure 1.

We argue the following in a series of claims: a) the algorithm can be implemented with an oracle in $\mathrm{NP}(A[1]) \oplus \Sigma_3^{\mathrm{p}}$; b) it runs in polynomial time; and c) on input 0^n, it prints an advice that accepts $A^{=n}$. Clearly, the theorem holds if we prove all of these claims.

Claim a. The algorithm can be implemented with an oracle in $\mathrm{NP}(A[1]) \oplus \Sigma_3^{\mathrm{p}}$.

Proof. We discuss how to implement steps marked (1), (2), and (3) in the algorithm.

Observe first that the predicate "H hashes the set $Acc(0^n, l, w)$" is in coNP, as noted in [22]: it is defined in a form $\forall\exists\forall$, but the $\exists$ quantifier has polynomial range and can be replaced by a deterministic search.

Step (1) is the most involved. Note that the function that associates $\|Cons(0^n, l)\|$ to each input $\langle 0^n, l\rangle$ is in #P. By Stockmeyer's Theorem, there is some 2-approximation $g \in \Delta_3^{\mathrm{p}}$ of this function. That is, $g(0^n, l)$ is between 1/2 and 2 times $\|Cons(0^n, l)\|$.

Fix n and l, and let G be $g(0^n, l)$. Whenever $\|Acc(0^n, l, w)\| \le \|Cons(0^n, l)\|/p(n)$ we also have $\|Acc(0^n, l, w)\| \le 2G/p(n)$. By the Coding Lemma, every set of size $2G/p(n)$ is hashed by some hash family of size $1 + \log(2G/p(n))$. Then, any H that satisfies the following predicate (*) also satisfies $hashes\_all(0^n, l, H)$:

(*) "for every $w \in \Sigma^n$, and for $t = 1 + \log(2G/p(n))$
   - if some H' of size t hashes $Acc(0^n, l, w)$, then H hashes $Acc(0^n, l, w)$, and
   - if some H' of size t hashes $Rej(0^n, l, w)$, then H hashes $Rej(0^n, l, w)$,

  where the linear transformations in H' have domain $\Sigma^{q(n)}$ and range Σ^t."

Furthermore, we show that some H exists that satisfies (*) and has polynomial size. Let w have length n. If, for example, $Acc(0^n, l, w)$ is hashed by some H' of size t, then it is not too large: by Lemma 9, $\|Acc(0^n, l, w)\| \le t \cdot 2^t \le q(n) \cdot 4G/p(n) = 4G/(n^2 \cdot q(n))$. We now apply Lemma 10: consider the collection of sets of the form $Acc(0^n, l, w)$ and $Rej(0^n, l, w)$ that are hashed by some hash family of size t. There are at most 2^{n+1} of these, and we have just shown that each of them has size at most $4G/(n^2 \cdot q(n))$. Define $m = 1 + \log(4G/(n^2 \cdot q(n)))$. Then this collection of sets is hashed by some hash family H of size $(n+2) \cdot m$, consisting of transformations $\Sigma^{q(n)} \to \Sigma^m$. In other words, this H satisfies (*).

Now, step (1) is implemented as follows: given n and l, compute $G = g(0^n, l)$; compute t and m as defined above; then find H of size $(n+2) \cdot m$ satisfying (*). We have shown that this H exists and satisfies $hashes\_all(0^n, l, H)$. Furthermore, predicate (*) has the form "$\forall\ ((\exists\forall) \Longrightarrow \forall)$", that can be rewritten in Π_2^{p} form. Hence, H can be found by prefix search over an oracle in $\mathrm{NP}(\Pi_2^{\mathrm{p}}) \subseteq \Sigma_3^{\mathrm{p}}$.

Remark. The size of the H computed will be important again in Claim c.

Recalling that the predicate "H does not hash a fixed Acc (or Rej) set" is in NP, an oracle in $\mathrm{NP}(A[1])$ is enough to implement step (2). By a similar argument, step (3) can be implemented by prefix search over an oracle in Σ_2^{p}. Therefore, the whole algorithm can be implemented with an oracle in $\mathrm{NP}(A) \oplus \Sigma_3^{\mathrm{p}}$. ■ **Claim a**

Claim b. The algorithm halts in polynomial time, if steps (1), (2), and (3) are solved as explained in Claim a.

Proof. It is enough to prove that the number of iterations in the main loop is polynomial. Let us say that a certain iteration "kills" an advice C if the counterexample w for that iteration proves that C is a wrong advice; that is, C and A disagree on w. We will show that each counterexample kills a substantial part of the advices not killed before.

Let w be the counterexample for an iteration, and assume $w \in A$. At this iteration, all advices that reject w are killed. Also, by the way w is chosen in step (1), H does not hash $Rej(0^n, l, w)$. Since $hashes\_all(0^n, l, H)$, $\|Rej(0^n, l, w)\| > \|Cons(0^n, l)\|/p(n)$, so a fraction $1/p(n)$ of advices in $Cons(0^n, l)$ are killed. The symmetrical argument works when $w \notin A$.

The initial number of advices of length $q(n)$ is $2^{q(n)}$, so after each iteration,

$$\begin{aligned} \|Cons(0^n, l)\| &\leq 2^{q(n)} \cdot \left(1 - 1/p(n)\right)^{\|l\|} \\ &\leq 2^{q(n)} \cdot 2^{-\|l\|/p(n)}. \end{aligned}$$

On the other hand, we are assuming that $A^{=n}$ has at least one correct advice of length $q(n)$. This advice is never killed because it agrees with every possible counterexample, so $\|Cons(0^n, l)\|$ is always positive. This means that $\|l\|$ can never exceed $p(n) \cdot q(n)$, and the maximum number of iterations of the main loop is bounded by a polynomial in n. ■ **Claim b**

Claim c. For all but finitely many n, the algorithm on input 0^n outputs an advice that accepts $A^{=n}$.

Proof. We must show that the algorithm always produces an output and that this output is correct. For the first part, consider any advice of length $q(n)$ accepting $A^{=n}$. If this correct advice accepts w, then w must be in A. By the comment included in the algorithm, H must hash $Rej(0^n, l, w)$. Similarly, if the advice rejects w then H hashes $Acc(0^n, l, w)$. Thus, at least one C satisfying the conditions can be found in step (3).

To prove that the output is correct, we show first that at the end of the loop,

$$H \text{ hashes } Acc(0^n, l, w) \implies H \text{ does not hash } Rej(0^n, l, w)$$

for every $w \in \Sigma^n$.

As H has size $(n+2)\cdot m$, by Lemma 9, H hashes no set of cardinality greater than $(n+2)\cdot m\cdot 2^m$. Some routine calculations show that this is less than $9G/n$ for all sufficiently large n, recalling that $m = 1+\log(4G/(n^2\cdot q(n)))$ and observing that $G \leq 2\cdot 2^{q(n)}$.

Suppose that H hashes $Acc(0^n, l, w)$. Because G is a 2-approximation of $\|Cons(0^n, l)\|$,

$$\|Acc(0^n, l, w)\| \leq 9G/n \leq 18\,\|Cons(0^n, l)\|/n.$$

Therefore,

$$\|Rej(0^n, l, w)\| \geq \|Cons(0^n, l)\|\cdot(1-18/n) \geq (G/2)\cdot(1-18/n).$$

This is greater than $9G/n$ for all sufficiently large n. Therefore H cannot hash $Rej(0^n, l, w)$.

Finally, let C be the output of the algorithm, and $|w| = n$. If C accepts w then H hashes $Rej(0^n, l, w)$ (by definition of C), H does not hash $Acc(0^n, l, w)$ (by the fact above), so $w \in A$ (by the comment in the algorithm). Similarly, if C rejects w, then $w \notin A$. ■ **Claim c**

This concludes the proof of the theorem.

3. Complexity of sparse set descriptions

We move now to a slightly different type of problems. We have studied the complexity of advice functions for sets in P/poly. Recall that P/poly is also the class of sets $\leq_{\mathrm{T}}^{\mathrm{P}}$-reducible to sparse sets (this observation is attributed to Meyer in [10]). Thus, a parallel question that has received some attention is the complexity of sparse sets to which sets in P/poly can be $\leq_{\mathrm{T}}^{\mathrm{P}}$-reduced. Note that this problem is naturally formulated as a set recognition problem, instead of a functional problem.

As in the case of advice functions, upper and lower bounds were known for this problem. The upper bound follows directly from Proposition 2.

Proposition 12 [14,21]. Every set $A \in \mathrm{P/poly}$ is $\leq_{\mathrm{T}}^{\mathrm{P}}$-reducible to some sparse set in $\Delta_3^{\mathrm{p}}(A)$.

More tightly, such a sparse set can be expressed as the difference of a $\Sigma_2^{\mathrm{p}}(A)$ set and a $\Pi_2^{\mathrm{p}}(A)$ set (O. Watanabe pointed out this fact to the author).

The best lower bound up to now is due to Gavaldà and Watanabe.

Theorem 13 [11]. There is a set B in P/poly that is reducible to no sparse set in $\mathrm{NP}(B) \cap \mathrm{coNP}(B)$.

Again, we would like to shrink the gap between the $\mathrm{NP} \cap \mathrm{coNP}$ lower bound and the Δ_3^{p} upper bound. For example, is it the case that every $A \in \mathrm{P/poly}$ is $\leq_{\mathrm{T}}^{\mathrm{P}}$-reducible to some sparse set in $\mathrm{NP}(A)$?

Mahaney [19] showed that this is the case when A is $\leq_{\mathrm{m}}^{\mathrm{P}}$-reducible to some sparse set, and his proof is easily extended to 1-tt and conjunctive reducibilities. For 2-tt, 3-tt, and disjunctive reducibilities some weaker results are known [2,5,6]. The question for general $\leq_{\mathrm{T}}^{\mathrm{P}}$-reducibility remains open, but with a further refinement of our proof technique we can prove the following theorem.

Theorem 14. Every set A in P/poly is $\leq_{\mathrm{T}}^{\mathrm{P}}$-reducible to a tally set in $\mathrm{NP}(A \oplus \Sigma_2^{\mathrm{p}})$.

Corollary 15. If $\mathrm{NP} = \mathrm{coNP}$ then every set A in P/poly is $\leq_{\mathrm{T}}^{\mathrm{P}}$-reducible to a tally set in $\mathrm{NP}(A)$.

Therefore, improving the $\mathrm{NP} \cap \mathrm{coNP}$ lower bound in [11] to NP is at least as hard as showing $\mathrm{NP} \neq \mathrm{coNP}$.

Proof of Theorem 14. Again, let A be an arbitrary set in P/poly and $q(n)$ the length of the advices for $A^{=n}$. Modify the algorithm in Figure 1 into a nondeterministic transducer M as follows: M expects inputs of the form $\langle 0^n, j\rangle$, and makes at most j iterations of the loop. At each iteration, M guesses the hash family H and the counterexample w, instead of computing them deterministically. Then it verifies using its oracle that they satisfy the conditions in steps (1) and (2) respectively. If any of these checks fails during the process, M halts without output. Otherwise, after at most j iterations, M executes step (3) as in the original algorithm, finding an advice C. Finally, M computes the lexicographically smallest C' accepting exactly the same set as C and outputs C'.

Now, H and w can be verified with oracles in Π_2^{p} and $A \oplus \mathrm{NP}$ respectively. Recall that step (3) was implemented with an oracle in Σ_2^{p}, and note that the computation of C' from C can also be made with a Σ_2^{p} oracle. Hence we can assume that to do this task M queries oracle $A \oplus K$, for some $K \in \Sigma_2^{\mathrm{p}}$.

Define the tally set T_A as:

$$T_A = \{\ 0^{\langle n,j,i,b\rangle}\ \mid \text{some computation of } M^{A\oplus K}(0^n, j) \text{ prints an advice whose } i\text{th bit is } b\ \}.$$

Since M is of type NP, T_A can be decided in $\mathrm{NP}(A \oplus K) \subseteq \mathrm{NP}(A \oplus \Sigma_2^{\mathrm{p}})$. We also show that oracle T_A allows to find correct advices for A in polynomial time, so $A \in \mathrm{P}(T_A)$:

(i) Given 0^n, obtain the maximum j such that $0^{\langle n,j,1,b\rangle} \in T_A$ for some $b \in \{0,1\}$.

(ii) Then obtain each bit of an advice for $A^{=n}$, by querying to T_A all words of the form $0^{\langle n,j,i,b\rangle}$ for all $i \leq q(n)$ and $b \in \{0,1\}$.

To see why this works, note that the j obtained in (i) is the maximum number of elements that the sample l can have, taken over all sequences of counterexamples. We have shown when proving Theorem 6 that this maximum is polynomial in n. It is easy to see that every nondeterministic computation of M that guesses a maximal sequence of counterexamples must print a correct advice, that is, an advice accepting $A^{=n}$. On the other hand, recall that when M outputs an advice C, no lexicographically smaller advice accepts the same language as C. Therefore, all computations of M that guess maximal sequences of counterexamples output *the same* advice, namely, the smallest advice accepting $A^{=n}$. Then, for every i, exactly one of $0^{\langle n,j,i,0\rangle}$ and $0^{\langle n,j,i,1\rangle}$ is in T_A, and step (ii) reconstructs a correct advice for $A^{=n}$.

Therefore, $A \in \mathrm{P}(T_A)$ and $T_A \in \mathrm{NP}(A \oplus \Sigma_2^{\mathrm{p}})$. ■

4. Final remarks and open questions

Recently, Köbler [15] has refined our proof technique and shown that in Theorem 6 one can replace Σ_3^{p} with Σ_2^{p}. His result is the first real improvement of Proposition 2, and places P/poly in the Δ_3^{p} level of the extended low hierarchy. Furthermore, by a very careful census argument, he has shown that P/poly is Θ_3^{p}-extended low, which is optimal by the results in [1,18].

An obvious open question is whether the Σ_2^{p} oracle in Köbler's bound can be eliminated, that is, whether every set A in P/poly has some advice function in $\mathrm{PF}(\mathrm{NP}(A))$. Or, one might try to show that this improvement is not possible from some reasonable assumption, such as the infiniteness of PH.

Acknowledgements

I have had recurrent discussions with José L. Balcázar on whether Proposition 2 could be improved. I am grateful to Ronald Book for his hospitality in UCSB. I thank them as well as Elvira Mayordomo for detailed comments on drafts of the paper. I also thank Johannes Köbler for liking the paper and deciding to improve on it, and Osamu Watanabe for teaching me the connections between these problems and learning theory.

References

[1] E. Allender and L. Hemachandra: "Lower bounds for the low hierarchy". *Journal of the ACM* **39** (1992), 234–251.

[2] E. Allender, L. Hemachandra, M. Ogiwara, and O. Watanabe: "Relating equivalence and reducibility to sparse sets". *SIAM Journal on Computing* **21** (1992), 521–539.

[3] D. Angluin: "Learning regular sets from queries and counterexamples". *Information and Computation* **75** (1987), 87–106.

[4] D. Angluin: "Queries and concept learning". *Machine Learning* **2** (1988), 319–342.

[5] V. Arvind, Y. Han, L. Hemachandra, J. Köbler, A. Lozano, M. Mundhenk, M. Ogiwara, U. Schöning, R. Silvestri, and T. Thierauf: "Reductions to sets of low information content". In *Proc. 19th International Colloquium on Automata, Languages, and Programming,* 162–173. Lecture Notes in Computer Science 623. Springer-Verlag, 1992.

[6] V. Arvind, J. Köbler, and M. Mundhenk: "Lowness and the complexity of sparse and tally descriptions". In *Proc. Third International Symposium on Algorithms and Computation,* 249–258. Lecture Notes in Computer Science 650. Springer-Verlag, 1992.

[7] T. Baker, J. Gill, and R. Solovay: "Relativizations of the P=?NP question". *SIAM Journal on Computing* **4** (1975), 431–442.

[8] J. Balcázar and R. Book: "Sets with small generalized Kolmogorov complexity". *Acta Informatica* **23** (1986), 679–688.

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

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

[11] R. Gavaldà and O. Watanabe: "On the computational complexity of small descriptions". In *Proc. 6th Annual Conference on Structure in Complexity Theory,* 89–101. IEEE Computer Society Press, 1991.

[12] L. Hemachandra, M. Ogiwara, and O. Watanabe: "How hard are sparse sets?". In *Proc. 7th Annual Conference on Structure in Complexity Theory,* 222–238. IEEE Computer Society Press, 1992.

[13] R. Karp and R. Lipton: "Some connections between nonuniform and uniform complexity classes". In *Proc. 12th Annual ACM Symposium on Theory of Computing,*

302–309. ACM Press, 1980.

[14] K. Ko and U. Schöning: "On circuit-size complexity and the low hierarchy in NP". *SIAM Journal on Computing* **14** (1985), 41–51.

[15] J. Köbler: "Locating P/poly optimally in the extended low hierarchy". In *Proc. 10th Symposium on Theoretical Aspects of Computer Science,* 28–37. Lecture Notes in Computer Science 665. Springer-Verlag, 1993.

[16] T. Long: "On restricting the size of oracles compared with restricting access to oracles". *SIAM Journal on Computing* **14** (1985), 585–597.

[17] T. Long and A. Selman: "Relativizing complexity classes with sparse oracles". *Journal of the ACM* **33** (1986), 618–627.

[18] T. Long and M. Sheu: "A refinement of the low and high hierarchies". Technical Report OSU-CISRC-2/91-TR6, The Ohio State University, 1991. To appear in *Mathematical Computing Theory.*

[19] S. Mahaney: "Sparse complete sets for NP: solution of a conjecture by Berman and Hartmanis". *Journal of Computer and System Sciences* **25** (1982), 130–143.

[20] N. Pippenger: "On simultaneous resource bounds". In *Proc. 20th Annual Symposium on Foundations of Computer Science,* 307–311. IEEE Computer Society Press, 1979.

[21] U. Schöning: *Complexity and Structure.* Lecture Notes in Computer Science 211. Springer-Verlag, 1986.

[22] M. Sipser: "A complexity theoretic approach to randomness". In *Proc. 15th Annual ACM Symposium on Theory of Computing,* 330–335. ACM Press, 1983.

[23] L. Stockmeyer: "On approximation algorithms for #P". *SIAM Journal on Computing* **14** (1985), 849–861.

[24] O. Watanabe: "A formal study of learning via queries". In *Proc. 17th International Colloquium on Automata, Languages, and Programming,* 139–152. Lecture Notes in Computer Science 443. Springer-Verlag, 1990.

[25] O. Watanabe: "A framework for polynomial time query learnability". Technical Report 92TR–0003, Department of Computer Science, Tokyo Institute of Technology, april 1992. To appear in *Mathematical Computing Theory.*

[26] O. Watanabe and R. Gavaldà: "Structural analysis of polynomial time query learnability". Technical Report 92TR–0004, Department of Computer Science, Tokyo Institute of Technology, april 1992. To appear in *Mathematical Computing Theory.*