Cryptographic Limitations on Parallelizing Membership and Equivalence Queries with Applications to Random-Self-Reductions

Marc Fischlin

Fachbereich Mathematik (AG 7.2)
Johann Wolfgang Goethe-Universität Frankfurt am Main
Postfach 111932
60054 Frankfurt/Main, Germany
marc@mi.informatik.uni-frankfurt.de
http://www.mi.informatik.uni-frankfurt.de/

Abstract

We assume wlog. that every learning algorithm with membership and equivalence queries proceeds in rounds. In each round it puts in parallel a polynomial number of queries and after receiving the answers, it performs internal computations before starting the next round. The query depth is defined by the number of rounds. In this paper we show that, assuming the existence of cryptographic one-way functions, for any fixed polynomial $d(n)$ there exists a concept class that is efficiently and exactly learnable with membership queries in query depth $d(n) + 1$, but cannot be weakly predicted with membership and equivalence queries in depth $d(n)$. Hence, concerning the query depth, efficient learning algorithms for this concept class cannot be parallelized. We also discuss applications to random-self-reductions and coherent sets.

1 Introduction

A fundamental problem in computer science is the question if and how sequential algorithms can be parallelized. This is an intrinsic problem in computational learning theory, too. Parallelizing PAC algorithms [30] is only a matter of parallelizing the internal computations if the complexity of the target concept is known in advance, because a sufficient number of random examples can be generated in a single concurrent step [12,31,11]. For learning algorithms with membership and equivalence queries [1,2] this problem also depends on the grade of adaptiveness of the queries. A quantitative formalization of the adaptiveness is via the query depth of a learning algorithm: We assume

wlog. that the learning algorithm proceeds in rounds. In each round it is allowed to put in parallel a polynomial number of membership and equivalence queries. After receiving the answers, it performs internal computation and then starts the next round. The query depth (as a function of complexity parameter n) is the maximal number of rounds, where the maximum is taken over all target concepts of complexity n. Bshouty and Cleve [10,9] prove that exact learning with membership and equivalence queries e.g. of read-once Boolean functions and monotone DNF formulas in n variables requires a query depth of $\Omega(n/\log n)$. Balcázar et al. [5] show that DFA with n states can be learned exactly with membership and equivalence queries in depth $O(n/\log n)$. Moreover, they prove that this bound is optimal as there cannot exist a learning algorithm that learns DFA exactly in query depth $o(n/\log n)$. These negative results are not tight in the sense that it remains open if there is a concept class where allowing a single additional round helps. Also, these lower bounds are sublinear and hold for exact learning exclusively. In this paper, we show that for any given polynomial $d(n)$ there is a concept class such that the class cannot be weakly predicted with membership and equivalence queries in query depth $d(n)$, though there exists a polynomial-time algorithm that learns every target concept in query depth $d(n)+1$ exactly with membership queries. We emphasize that, adding a single level of adaptiveness, we can learn this class exactly, while any learning algorithm with depth $d(n)$ miserably fails, i.e., cannot satisfy a potentially weaker requirement than PAC-learnability (with queries). While our impossibility result as well as the lower bound of [5] only hold for polynomial-time algorithms, the result of Bshouty and Cleve is also valid for computationally unbounded parallel learners — as long as the number of queries is polynomially bounded. In contrast to [10,5,9], who deal with "natural" concept classes, our class is somewhat artificial and tailor made to prove the desired result.

The intractability of our concept class is based on a cryptographic assumption, namely the existence of one-way functions. These are functions that are easy to evaluate but hard to invert on a random value. Despite complexity based impossibility results (see for example [26]) several negative results for learning algorithms have been based on cryptographic primitives. Angluin and Kharitonov [3] use one-way functions to show that membership queries do not add any power to PAC-algorithms when learning DNF formulas. Similarly, Kearns and Valiant [23] and Kharitonov [24] show that polynomial-size Boolean formulas are not efficiently PAC-learnable with membership queries if one-way functions exist. Rivest and Yin [28] present a concept class based on the existence of one-way functions where self-directed learning is inferior to teacher-directed learning. We exploit their idea to define our concept class using so-called collections of pseudorandom functions. Loosely speaking, a collection of pseudorandom functions is a sequence $(F_n)_{n\in\mathbb{N}}$ of function sets $F_n \subset \{g : \{0,1\}^n \to \{0,1\}^n\}$ with the following two properties: Each set F_n contains 2^n (not necessarily distinct) functions, where every function $f \in F_n$ is

described by a key $k \in \{0,1\}^n$ such that one can efficiently compute $f(x)$ given k and x; second, F_n preserves the randomness property, i.e., if we uniformly choose a key $k \in \{0,1\}^n$ then the function described by this key "looks" like a uniformly chosen function from the set $\{g : \{0,1\}^n \to \{0,1\}^n\}$. Note that most of the functions in the set of all 2^{n2^n} functions $g : \{0,1\}^n \to \{0,1\}^n$ must have exponential description size. This means that there cannot exist an algorithm that evaluates each function in polynomial time in n given the identifier of the function and the value as input. In contrast, pseudorandom functions have this property but still look sufficiently random. It is well-known that collections of pseudorandom functions exist if and only if one-way functions exist [20,19]. Furthermore, the existence of cryptographic one-way functions implies P $\neq$ NP though it is not known if the converse holds (see [18] for a discussion). Given any collection of pseudorandom functions we define the concepts of complexity n by the keys of F_n and such that a particular query sequence of depth $d(n)+1$ yields the key of the function resp. the name of the target concept. Hence, we can easily learn the target concept in depth $d(n)+1$. On the other hand, there cannot exist any probabilistic polynomial-time algorithm that, after experimenting using membership and equivalence queries in query depth $d(n)$, classifies a random example correctly with probability at least $1/2 + 1/p(n)$ for an arbitrary positive polynomial $p(n)$ and all but finite $n \in \mathbb{N}$. Otherwise we derive a contradiction to the pseudorandomness of the underlying collection.

We stress that the impossibility of learning our concept class in depth $d(n)$ does not rely on any universal bound on the width of the queries. Though, the number of queries in each round is polynomially bounded since the impossibility result holds for polynomial-time learning algorithms only. But this specific bound depends on the running time of the algorithm in question. If we also bound the width by some fixed polynomial $w(n)$ for any learning algorithm, then we restrict the total number of queries by $t(n) = d(n)w(n)$. In this case, using polynomials over finite fields, we can easily construct a concept class that cannot be weakly predicted with queries in depth $d(n)$ and width $w(n)$, though it can be learned exactly with $n \cdot (t(n) + 1)$ nonadaptive membership queries (which, of course, can be arbitrarily distributed on any number of rounds). The negative result for depth- and width-bounded algorithms does not involve any unproven assumption and even holds for computationally unbounded learning algorithms. Details are given in Appendix A.

We apply our result on the non-parallelizability of the queries to so-called random-self-reductions [8]. Informally, a language $\mathcal{L}$ is self-reducible [29] if, for any x, we can compute the characteristic function $\chi_{\mathcal{L}}$ of $\mathcal{L}$ at x from values $\chi_{\mathcal{L}}(y_1), \ldots, \chi_{\mathcal{L}}(y_m)$, where $|y_1|, \ldots, |y_m| < |x|$. In other words, $\mathcal{L}$ is self-reducible if membership can be decided by querying the oracle $\chi_{\mathcal{L}}$ for smaller values. A classic example of a self-reducible language is SAT. An interesting special case of self-reductions are random-self-reductions, where each

query y_i is a random value distributed independently of x (but not necessarily independently of the other queries). For an overview about applications of random-self-reductions we refer to [14,17]. Unlike self-reductions, random-self-reductions do not require that the queries are smaller; if the length of the queries equals the length of x then the reduction is called length-preserving. The query depth of a random-self-reduction is defined analogously to the query depth of a learning algorithm. Feigenbaum et al. [16] show that adaptive (more specifically, query depth $|x|$) random-self-reductions are more powerful than nonadaptive ones. Combining our result with [16] we establish the following hierarchy: Let $\beta(n)$ be an unbounded, nondecreasing function $\beta(n)$ such that $n^{\beta(n)}$ is time-constructible (e.g., $\beta(n) = \log^* n$) and let $d(n)$ be a fixed polynomial. If one-way functions exist, there is a language in DSPACE($n^{\beta(n)}$) such that there is a random-self-reduction with query depth $d(n) + 1$, while every length-preserving random-self-reduction of depth $d(n)$ fails. We also show slight extensions of this result, for example to coherent sets, i.e., sets $\mathcal{L}$ where membership of any x can be decided efficiently with help of the oracle $\chi_{\mathcal{L}-\{x\}}$.

The paper is organized as follows. In Section 2 we introduce notations and definitions of learning theory, cryptography and random-self-reductions and coherence. In Section 3 we define our concept class and prove the positive resp. negative result about learnability. The issue of depth- and width-bounded algorithms appears in Appendix A. In Section 4, we apply the results of Section 3 to random-self-reductions as well as coherent sets.

2 Preliminaries

We introduce some basic notations. For a finite set S let $y \in_R S$ denote a uniformly chosen element y from S. We write $\pi_j(y) \in \{0,1\}$ for the projection onto the j-th bit of $y \in \{0,1\}^n$, where n is understood from the context and $j \in \{1,\ldots,n\}$. For notational convenience, we sometimes switch between natural numbers and their binary representations.

2.1 Computational Learning Theory

We briefly recall notations and definitions of learning theory. Let $X = (X_n)_{n\in\mathbb{N}}$ denote the *domain*, where $X_n \subseteq \{0,1\}^{p(n)}$ for some polynomial $p(n)$. For $k \in \{0,1\}^n$, a *concept* c_k is a subset of X_n. We call k the *name* of c_k. Let $\mathcal{C}_n = \{c_k \mid k \in \{0,1\}^n\}$ and define the *concept class* by $\mathcal{C} = (\mathcal{C}_n)_{n\in\mathbb{N}}$. We usually view c_k as a Boolean function; that is, $c_k(x) = 1$ if $x \in c_k$ and $c_k(x) = 0$ otherwise. Let $\mathcal{D} = (\mathcal{D}_n)_{n\in\mathbb{N}}$ be a sequence of distributions $\mathcal{D}_n$ on X_n. We say that $\mathcal{D}$ is *efficiently sampleable* if there is a probabilistic polynomial-time

algorithm such that for input 1^n the output of the algorithm is identically distributed to $\mathcal{D}_n$.

Following Kharitonov [24] we define a prediction with membership and equivalence queries algorithm (pwme-algorithm). Let $\mathcal{C}$ be a concept class and $\mathcal{D}$ be an efficiently sampleable distribution. The error parameter function $\epsilon : \mathbb{N} \to \mathbb{Q}^+$ determines the accuracy of the learning algorithm. A pwme-algorithm L is a probabilistic algorithm that gets inputs n and $\epsilon(n)$ and, after a target concept $c_k \in \mathcal{C}$ has been chosen, may make in addition to internal computations

- membership queries, i.e., query the oracle c_k for arbitrary $x \in X_n$
- equivalence queries, i.e., give $k' \in \{0,1\}^n$ to the oracle and receive the answer "yes" if $c_k = c_{k'}$ resp. a counterexample $x \in X_n$ with $c_k(x) \neq c_{k'}(x)$
- exactly one challenge query, where an example $z \in X_n$ is randomly generated according to the distribution $\mathcal{D}_n$ and returned to L; algorithm L then outputs a guess for $c_k(z)$ and stops

We say that L *successfully predicts* $\mathcal{C}$ with respect to $\mathcal{D}$ and ϵ iff, for all $n \in \mathbb{N}$ and $c_k \in \mathcal{C}_n$, the probability that L's guess is correct, i.e., equals $c_k(z)$, is at least $1 - \epsilon(n)$. We call $\mathcal{C}$ *efficiently predictable* with respect to $\mathcal{D}$ and ϵ iff there is a pwme-alogithm L that successfully predicts $\mathcal{C}$ with respect to $\mathcal{D}$ and ϵ and runs in polynomial time in n and $1/\epsilon(n)$. We say that $\mathcal{C}$ is *weakly predictable* with respect to $\mathcal{D}$ iff it is efficiently predictable with respect to $\mathcal{D}$ and $\epsilon(n) = 1/2 - 1/p(n)$ for some polynomial $p : \mathbb{N} \to \mathbb{Q}^+$ and all but finitely many $n \in \mathbb{N}$. We call a pwme-algorithm L a pwm-algorithm if L does not put equivalence queries.

Note that $\mathcal{C}$ and $\mathcal{D}$ are fixed and therefore known by L. Note also that L does not need randomly generated examples (as in case of PAC algorithms), because we only consider efficiently sampleable distributions. Thus, L can generate an example by itself and then put a membership query for this example. Moreover, we remark that unpredictability implies impossibility of PAC-learnability with queries (see the discussion in [24]).

Next, we define the query depth of a pwme-algorithm. We assume wlog. that any pwme-algorithm L proceeds in rounds. At the beginning of each round, L puts in parallel membership and equivalence queries and receives the answers. Then it performs internal computations and starts the next round. After finishing the last round, it is allowed additional computations and finally gives its output. The pwme-algorithm has query depth $d(n)$ if it takes at most $d(n)$ rounds for inputs $n, \epsilon(n)$ and all target concepts of complexity n. A concept class $\mathcal{C}$ is *weakly predictable in query depth* $d(n)$ with respect to $\mathcal{D}$ if it is weakly predictable by a pwme-algorithm with query depth $d(n)$.

As for the positive result on the learnability of our concept class, we say that

a concept class $\mathcal{C}$ is *exactly learnable with membership queries* iff there exists a polynomial-time algorithm L such that for all $n \in \mathbb{N}$ and $c_k \in \mathcal{C}_n$, algorithm L with oracle access to c_k outputs a name $k' \in \{0,1\}^n$ such that $c_k = c_{k'}$. The query depth of such an algorithm is defined analogously to the depth of a pwme-algorithm. If this depth is bounded by $d(n)$, we call $\mathcal{C}$ *exactly learnable with membership queries in query depth* $d(n)$.

2.2 Cryptography

In this section we introduce the cryptographic background. A function $\delta : \mathbb{N} \to \mathbb{R}^+$ is called *negligible* iff it vanishes faster than any polynomial fraction, i.e., iff for any polynomial $p : \mathbb{N} \to \mathbb{R}^+$ there exists $n_0 \in \mathbb{N}$ such that $\delta(n) < 1/p(n)$ for all $n \geq n_0$. For instance, $\delta(n) = 2^{-n}$ is negligible. For the rest of the paper, we abbreviate "there exists n_0 such that ... for all $n \geq n_0$" by "for all sufficiently large n". In the sequel we use the following facts about negligble functions: Let $f(n) \geq 1/p_0(n)$ for some positive polynomial p_0 and infinitely many n and let $\delta(n)$ be a negligible function; then $f(n) - \delta(n) \geq 1/2p_0(n)$ for infinitely many n. Additionally, it is easy to see that $p(n) \cdot \delta(n)$ is negligible for any positive polynomial $p(n)$ if and only if $\delta(n)$ is negligible.

A collection $F = (F_n)_{n\in\mathbb{N}}$ of functions is a sequence of functions $F_n : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^n$. The first argument is called the key and usually denoted by $k \in \{0,1\}^n$. If it is fixed and n is understood, we write $F_k(\cdot)$ for the function $F_n(k,\cdot)$. As explained in the introduction, a collection F of functions is pseudorandom if, roughly speaking, a randomly chosen function from F looks like a uniformly drawn function from the set $\{g : \{0,1\}^n \to \{0,1\}^n\}$. In this paper, we use a different formalization which fits better in our scenario. Yet, this definition is equivalent to the one usually used in the literature (cf. [19]).

Consider the following experiment. Let D be a probabilistic polynomial-time algorithm. At the beginning, a random key $k \in_R \{0,1\}^n$ is chosen and kept secret from D. D is given 1^n (n in unary) as input and is allowed to adaptively query the oracle $F_k(\cdot)$ for values of its choice. Then D outputs a challenge $y \in \{0,1\}^n$ such that y has not been queried previously and D is disconnected from the oracle. A bit $b \in_R \{0,1\}$ is chosen at random as well as a random string $r \in_R \{0,1\}^n$ and D is given (Q_0, Q_1) where $Q_b = F_k(y)$ and $Q_{1-b} = r$. That is, D receives the value of F_k at y and a random string in random order. Finally, algorithm D outputs a guess $g \in \{0,1\}$ for b. The distinguishing advantage of D is the probability (over the choice of k and the coin tosses of D) that D's guess is correct minus the pure guessing probability: $\mathsf{Adv}_D^F = |\mathrm{Prob}[b = g] - 1/2|$. Note that Adv_D^F is a function of $n \in \mathbb{N}$, the input of D. Roughly speaking, F is pseudorandom if any distinguisher D cannot predict b essentially better than with probability $1/2$ for sufficiently large n.

Definition 1 *A collection $F = (F_n)_{n\in\mathbb{N}}$ of functions $F_n : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^n$ is called a collection of pseudorandom functions iff*

- *there exists a polynomial-time algorithm $\mathcal{F}$ such that $\mathcal{F}(k,x) = F_n(k,x)$ for any $k, x \in \{0,1\}^n$ and all $n \in \mathbb{N}$*
- *the distinguishing advantage $\mathsf{Adv}_D^F(n)$ of any probabilistic polynomial-time algorithm D is negligible*

Note that the first property means that $(F_n)_{n\in\mathbb{N}}$ is computable in polynomial time in n. It is well-known that collections of pseudorandom functions exists if and only if one-way functions exist [19,20]. One-way functions are believed to be the weakest assumption for non-trivial cryptography [22,25].

We say that a collection F of pseudorandom functions is *non-uniformly secure* if it even holds that the distinguishing advantage of any polynomial-size circuit family D is negligible. Non-uniformly secure collections of pseudorandom functions can be derived from non-uniformly secure one-way functions, i.e., one-way functions that remain hard to invert on random values even for polynomial-size circuit families. We remark that security of one-way functions against non-uniform algorithms is also a widely accepted assumption in cryptography.

In the sequel we will use the following fact about pseudorandom functions. Consider the variation of the experiment above, where D, after querying the oracle $F_k(\cdot)$, outputs a pair (y,z) such that y has not been passed to the oracle yet. The prediction probability $\mathsf{Pred}_D^F(n)$ of D (as a function of n) is the probability that $F_k(y) = z$. That is, the prediction probability denotes the probability that D is able predict the function value at y without having seen it. The probability is taken over the random choice of the key and the coin tosses of D. The proof of the following fact can be found in [19]:

Fact 2 *Let F be a collection of pseudorandom functions. Then the prediction probability $\mathsf{Pred}_D^F(n)$ of any probabilistic polynomial-time algorithm D is negligible.*

Intuitively, for a collection of pseudorandom functions the prediction probability is negligible because if one was be able to predict a value then it would also be easy to distinguish it from a random string. The converse of Fact 2 does not hold: Consider for example the collection $G = (G_n)_{n\in\mathbb{N}}$ of functions defined by $G_n(k,x) = (F_n(k,x), x)$ for a collection F of pseudorandom functions. Clearly, this collection achieves negligible prediction probability (because one must be able to predict the left half, i.e., the output of the pseudorandom function). It is, however, not a collection of pseudorandom functions as the argument x is appended to the output, enabling us to distinguish function values from random strings.

Again, there is the non-uniform counterpart of Fact 2. That is, the prediction probability of any polynomial-size circuit family D is negligble if F a non-uniformly secure collection of pseudorandom functions.

2.3 Random-Self-Reducible and Coherent Sets

In this section we introduce the notions of random-self-reductions [8] and coherent sets [32]. The definition of the query depth of the corresponding primitive is a straightforward extension of the definition for learning algorithms. At the end of this section, we briefly recall the definitions of the complexity classes that we deal with in this paper.

A deterministic algorithm defines a random variable if we choose some part of the input of the algorithm at random. For instance, we write $A(x)$ for the random variable that describes the output distribution of algorithm A for input (x, r) if we choose r at random and x is fixed. Also, the omitted input r in $A(x)$ will be clear from the context. The following definition is taken from [16]:

Definition 3 *A function $f : \{0,1\}^* \to \{0,1\}^*$ is called nonadaptively $k(n)$-random-self-reducible if there exist polynomial-time algorithms ϕ, σ and a polynomial $p(n)$ such that for all x we have*

$$f(x) = \phi\Big(x, r, f\big(\sigma(1, x, r)\big), \ldots, f\big(\sigma(k(|x|), x, r)\big)\Big)$$

with probability at least $2/3$ over the choice of $r \in_R \{0,1\}^{p(|x|)}$. Additionally, for all $x, y \in \{0,1\}^n$ the random variables $\sigma(i, x)$ and $\sigma(i, y)$ are identically distributed.

From the definition it immediately follows that a single value $\sigma(i, x, r)$ does not yield any information about x. Yet, $\sigma(i, x)$ and $\sigma(j, x)$ are dependent in general and may therefore reveal x. More generally, we consider *adaptive* random-self-reductions where $\sigma(i, x, r)$ may also depend on the previous answers $f(\sigma(1, x, r)), \ldots, f(\sigma(i-1, x, r))$ for $i = 1, \ldots, k(|x|)$. It is easy to see that the error probability $1/3$ can be decreased to $2^{-q(n)}$ for any polynomial $q(n)$ by standard techniques for both adaptive and nonadaptive reductions. In particular, lowering the error probability by majority decision preserves the query depth. We remark that the notion of the query depth of random-self-reductions has been mentioned implicitely in [15] though, to best of our knowledge, it has not been investigated further — except for the special cases of adaptive and nonadaptive reductions.

A random-self-reduction is *oblivious* if the queries $\sigma(1, x, r), \ldots, \sigma(k(|x|), x, r)$ do not depend on x, i.e., $\sigma(i, x, r) = \sigma(i, 1^n, r)$ for $i = 1, \ldots, k(|x|)$. It is called

deterministic if the queries do not depend on r. In contrast to ordinary self-reductions we do not restrict the queries $\sigma(i,x,r)$ to be smaller than the input, but allow queries with arbitrary length. We say that a random-self-reduction is *length-preserving* if $|\sigma(i,x,r)| = |x|$ for all i,x,r. It is called *length-monotone* if $|\sigma(i,x,r)| \leq |x|$. We say that a set $\mathcal{L}$ is random-self-reducible if $\chi_{\mathcal{L}}$ is.

Closely related to random-self-reducible sets are so-called coherent sets, which we define next. Let $f : \{0,1\}^* \to \{0,1\}$ be a Boolean function. An *examiner* for f is a probabilistic polynomial-time oracle Turing machine E that, on input x, never queries the oracle f for x. Let $E^f(x)$ denote the random variable that describes the output.

Definition 4 *A set $\mathcal{L}$ is called coherent if there exists an examiner E such that $E^{\chi_{\mathcal{L}}}(x) = \chi_{\mathcal{L}}(x)$ with probability at least $2/3$ for all x.*

Again, the error probability can be decreased to $2^{-q(n)}$ while preserving the query depth. We say that $\mathcal{L}$ is *deterministic coherent* if E is (deterministic) polynomial-time. $\mathcal{L}$ is called *weakly coherent* if E is a polynomial-size circuit family. In this case, we say that E is a *weak examiner*. If $\mathcal{L}$ is not coherent it is called *incoherent*.

It is easy to see (for example [7]) that for every language $\mathcal{L}$ the set $\mathcal{L} \oplus \mathcal{L} = \{0x \mid x \in \mathcal{L}\} \cup \{1x \mid x \in \mathcal{L}\}$ is coherent. Additionally, Beigel and Feigenbaum [7] show that every random-self-reducible set is also weakly coherent, though it is not known whether this result also extends to uniform examiners. The converse is unlikely to hold, as every NP-complete set is coherent but, unless the polynomial hierarchy collapses at the third level, is not random-self-reducible in query depth $O(\log n)$. See [15] for details.

As our results involve some complexity classes, we give a brief overview about these classes. We refer the reader to [4] for a comprehensive treatment of structural complexity. A language $\mathcal{L}$ is in DSPACE($f(n)$) if there exist a deterministic Turing machine deciding membership in $\mathcal{L}$ with bounded work space $O(f(n))$. It is in NE if there exist a nondeterministic Turing machine computing $\chi_{\mathcal{L}}$ with running time at most $2^{O(n)}$. Similarly, $\mathcal{L} \in$ BPE if there is a probabilistic Turing machine that decides membership in $\mathcal{L}$ with bounded error in time $2^{O(n)}$. We say that $\mathcal{L} \in$ NEEE if there is a nondeterministic Turing machine that computes $\chi_{\mathcal{L}}$ with time bound $2^{2^{2^{O(n)}}}$. Membership in the class BPEEE is defined accordingly. Finally, we remark that a function $f(n)$ is *time-constructible* if it can be computed in $O(f(n))$ steps, i.e., if a deterministic Turing machine computes $1^{f(n)}$ on input 1^n in time $O(f(n))$. Time-constructible functions are very important for diagonalization techniques (cf. [4]).

3 Limitations on Parallelizing Queries

In this section we define our concept class based on any collection of pseudorandom functions. We show that this class cannot be predicted with membership queries in depth $d(n)$, though it can be learned exactly in depth $d(n)+1$. Finally, we discuss that prediction remains hard even if we add equivalence queries.

3.1 Definition and Positive Result

Let $F = (F_n)_{n\in\mathbb{N}}$ be a collection of pseudorandom functions and let $d(n)$ be a fixed polynomial. The basic idea is to define a concept $c_k \in \mathcal{C}_n$ for each function $F_n(k,\cdot) \in F_n$ such that querying this concept for a specific sequence of values in depth $d(n)+1$ always yields the key, but such that any algorithm with depth $d(n)$ essentially faces a pseudorandom function which in turn implies weak unpredictability. To this end, we modify F to a collection F^* which then allows us to construct the desired concept class.

For a function $F_k(\cdot) = F_n(k,\cdot)$ and $i = 0,\ldots,d(n)$ define

$$y_k^{(i)} = \begin{cases} 0^n & \text{if } i = 0 \\ F_k\left(y_k^{(i-1)}\right) & \text{else} \end{cases}$$

That is, $y_k^{(i)}$ is obtained by iterating i times $F_k(\cdot)$ at 0^n. For each $k \in \{0,1\}^n$ alter $F_k(\cdot)$ to a function $F'_k(\cdot)$ by setting

$$F'_k(x) = \begin{cases} k & \text{if } x = y_k^{(d(n))} \\ F_k(x) & \text{else} \end{cases}$$

Thus, the only difference between F'_k and F_k is that F'_k reveals the key if it is evaluated at $y_k^{(d(n))}$. Do we always obtain the key k when iterating $F'_k(\cdot)$ *exactly* $d(n)$ times at 0^n? Not necessarily. The reason is that if $y_k^{(i)} = y_k^{(d(n))}$ for some $i \in \{0,\ldots,d(n)-1\}$ then by definition $F'_k(y_k^{(i)})$ equals k instead of $y_k^{(i+1)}$ and hence $F'_k(\cdots(F'_k(0^n)))$ does not necessarily yield k. This means that we might not be able to learn this class exactly in depth $d(n)+1$, because we cannot verify errorless that we have in fact obtained a key k' with $F'_k(\cdot) = F'_{k'}(\cdot)$. To overcome this problem we change F' to another collection F^*. We first show that for the collection F of pseudorandom functions collisions $y_k^{(i)} = y_k^{(j)}$ occur only with very small probability:

Lemma 5 *The probability (over the choice of the key k) that $y_k^{(i)} = y_k^{(j)}$ for some $i < j$ with $i,j \in \{0,\ldots,d(n)\}$ is negligible.*

Intuitively, this is clear because it obviously holds for truly random functions and pseudorandom functions have a similar randomness property.

PROOF. We prove that otherwise there exist a polynomial-time algorithm D that contradicts the unpredictability of the collection F of pseudorandom functions. Specifically, we show that in this case D successfully predicts the value $F_k(y)$ for an appropriate y with probability at least $1/p(n)$ for a polynomial $p(n)$ and infinitely many $n \in \mathbb{N}$, where F_k is the randomly chosen function D is given oracle access to.

Assume that the probability that there exist i, j as in the claim is not negligible. More precisely, let this probability be greater than $1/q(n)$ for a polynomial q and infinitely many n. For a fixed key k we call a pair (i, j) bad if $0 \leq i < j \leq d(n)$ and $y_k^{(i)} = y_k^{(j)}$. If a bad pair exist then there is also a minimal bad pair (i_0, j_0), i.e., such that there does not exist another bad pair (i, j) with $j < j_0$. We construct D as follows. D tries to guess (i_0, j_0) by choosing $J \in_R \{1, \ldots, d(n)\}$ and $I \in_R \{0, \ldots, J-1\}$ at random. Then D computes $y_k^{(0)}, \ldots, y_k^{(J-1)}$ by querying the oracle $F_k(\cdot)$. If $y_k^{(J-1)} \neq y_k^{(0)}, \ldots, y_k^{(J-2)}$ then D outputs $(y_k^{(J-1)}, y_k^{(I)})$. Else D gives an arbitrary output. If there exist a (minimal) bad pair (i_0, j_0) then $(I, J) = (i_0, j_0)$ with probability at least $1/d^2(n)$. In this case, $y_k^{(J-1)} \neq y_k^{(0)}, \ldots, y_k^{(J-2)}$ because (i_0, j_0) is minimal. Additionally, $F_k(y_k^{(J-1)}) = y_k^{(J)} = y_k^{(I)}$. Hence, the prediction of D is correct with probability at least $1/d^2(n)q(n)$ infinitely often, which is not negligible. This contradicts the unpredictability of F. $\square$

In the sequel we denote by $K_n \subseteq \{0,1\}^n$ the set of keys $k \in \{0,1\}^n$ for which $y_k^{(0)}, \ldots, y_k^{(d(n))}$ are distinct and, for technical reasons, we also demand that $k \neq 0^n$. Then the set K_n contains all but a negligible fraction of the keys $k \in \{0,1\}^n$ by Lemma 5 and since excluding the single key 0^n from each K_n does not affect this asymptotic property. Change each function $F'_k(\cdot)$ to a function $F^*_k(\cdot)$ according to

$$F^*_k(x) = \begin{cases} F'_k(x) & \text{if } k \in K_n \\ 0^n & \text{else} \end{cases}$$

That is, if there is a collision $y_k^{(i)} = y_k^{(j)}$ or $k = 0^n$ then we reset the function for key k to a trivial function. Moreover, we now have that iterating $F^*_k(\cdots(F^*_k(0^n)))$ for $d(n)$ times always reveals a key k' such that $F^*_{k'}(\cdot) = F^*_k(\cdot)$. Futhermore, for all keys k in K_n we have $F^*_k(\cdot) = F'_k(\cdot)$, i.e, for all but a negligible fraction of the keys, $F^*_k(\cdot)$ equals the pseudorandom function $F_k(\cdot)$ (except for the value at $y_k^{(d(n))}$). Thus, F^* is the collection we are looking for: On one hand, it is easy to obtain a key in depth $d(n)+1$ by querying for $y_k^{(0)}, \ldots, y_k^{(d(n))}$.

On the other hand, it is hard to predict a function value if one does not query for $y_k^{(d(n))}$, which any learning algorithm with depth $d(n)$ cannot, unless it can already guess one of the values $y_k^{(i)}$ without having seen $y_k^{(i-1)}$.

Define the concept class $\mathcal{C} = (\mathcal{C}_n)_{n\in\mathbb{N}}$ by $\mathcal{C}_n = \{c_k \mid k \in \{0,1\}^n\}$, where

$$c_k = \left\{(x,j) \in \{0,1\}^{n+\lceil \log n\rceil} \;\middle|\; \pi_j(F_k^*(x)) = 1\right\}$$

Recall that $\pi_j(F_k^*(x))$ is the projection of $F_k^*(x)$ onto bit j. The distribution $\mathcal{D}_n$ on $\{0,1\}^{n+\lceil \log n\rceil}$ is described by picking $x \in_R \{0,1\}^n$ and $j \in_R \{1,\ldots,n\}$ independently. Obviously, $\mathcal{D}$ is efficiently sampleable. At the end of this section we discuss that any sufficiently "smooth" and sampleable distribution works, too.

Lemma 6 *The concept class $\mathcal{C}$ is exactly learnable with membership queries in query depth $d(n)+1$.*

PROOF. Let c_k be the target concept. In each round $i = 1,\ldots,d(n)+1$ query in parallel the oracle c_k for $(y^{(i-1)},1),\ldots,(y^{(i-1)},n)$, where $y^{(0)} = 0^n$ and $y^{(i)}$ is the concatenation of the answers in round i. Since iterating $d(n)$ times F_k^* at 0^n reveals a key k' with $F_{k'}^*(\cdot) = F_k^*(\cdot)$, we finally obtain a name k' of a concept such that $c_{k'} = c_k$. $\square$

3.2 Impossibility of Learning with Membership Queries

We prove the impossibility result. In this part, we restrict ourself to learning algorithm with membership queries only.

Lemma 7 *$\mathcal{C}$ is not weakly predictable with membership queries in query depth $d(n)$ with respect to $\mathcal{D}$.*

The outline of the proof is as follows. If $\mathcal{C}$ was weakly predictable then this would also hold if we choose the target concept at random, namely select $k \in_R \{0,1\}$ and let c_k be the target concept. Since the query depth of the learning algorithm is bounded by $d(n)$, it cannot query for $y_k^{(d(n))}$ and therefore obtain the key k, unless it can guess at least one of the values $y_k^{(1)},\ldots,y_k^{(d(n))}$ — or if $k \notin K_n$. But since the latter event occurs with negligible probability only, this would contradict the unpredictability of the pseudorandom function. Hence, as the learning algorithm cannot obtain the key for almost all choices of k, predicting a random example is almost as hard as distinguishing between the value of the pseudorandom function and a random string.

PROOF. Assume that there exists a pwm-algorithm L that weakly predicts $\mathcal{C}$ with respect to $\mathcal{D}$. Let $p(n)$ denote the polynomial such that L predicts correctly with probability at least $1/2 + 1/p(n)$ for infinitely many $n \in \mathbb{N}$.[1] Since L predicts $\mathcal{C}_n$ for all target concepts c_k, it also predicts $\mathcal{C}_n$ if we choose $k \in_R \{0,1\}^n$ and thus c_k at random. From L we construct a successful distinguisher D for the collection of pseudorandom functions $F = (F_n)_{n\in\mathbb{N}}$. The distinguisher D is given oracle access to a function $F_k(\cdot)$ in F_n, where $k \in_R \{0,1\}^n$ is chosen at random. D is allowed to query the oracle $F_k(\cdot)$ about values of its choice. Then D is supposed to distinguish a random string from the value $F_k(y)$ without having queried about y (or, as in the variation, D is supposed to predict the value $F_k(y)$ for some new y). Basically, D simulates L answering L's queries using the oracle $F_k(\cdot)$. Let us describe the simulation in more detail. D runs L until L outputs a membership query (x, j) or, more generally, a sequence of queries. D then passes x to the oracle and receives the answer $z = F_k(x)$ resp. proceeds each query of the sequence sequentially in the same way. D extracts the j-th bit from z, returns it to L, and continues to run algorithm L.

We first show that the probability that the learning algorithm L queries $(y_k^{(d(n))}, j)$ for some j or that $k \notin K_n$ is negligible. If L does not query $(y_k^{(d(n))}, j)$ for any j and if $k \in K_n$ then D is able to answer all queries of L using its oracle $F_k(\cdot)$. This is possible as $c_k(x, j) = \pi_j(F_k(x))$ except for $x = y_k^{(d(n))}$ or $k \notin K_n$. If L queries about $(y_k^{(d(n))}, j)$ for some j and $k \in K_n$ then D is supposed to return the j-th bit of the key k to L, because $c_k(y_k^{(d(n))}, j) = \pi_j(F_k^*(y_k^{(d(n))})) = \pi_j(k)$. But D does not know the secret key k. Hence, if L queries about $y_k^{(d(n))}$ then the simulation fails.[2] The simulation also fails if $k \notin K_n$ because $F_k^*(x) = 0^n$ for such k while it is unlikely that the same holds for the oracle $F_k(\cdot)$. Fortunately, the probability that any of these events happens is negligible and, given that the simulation succeeds, it then follows that L cannot weakly predict the concept class.

Claim 8 *The probability that L queries $\left(y_k^{(d(n))}, j\right)$ for some $j \in \{1, \ldots, n\}$ or that $k \notin K_n$ is negligible.*

PROOF. First note that for any events A, B we have

$$\mathrm{Prob}[A \vee B] \leq \mathrm{Prob}[A \mid \neg B] + \mathrm{Prob}[B]$$

[1] Note that we only demand that L predicts correctly infinitely often. Weak predictability actually requires L to predict correctly for *all sufficiently large* n. This even strengthens our result.

[2] D might be able to guess some bits of the secret key k, but we even assume that D cannot guess *any* of the bits. So we say that the simulation fails if $y_k^{(d(n))}$ appears in any of L's queries.

Let A be the event that L queries $y_k^{(d(n))}$ and B that $k \notin K_n$. We conclude that if the probability $\mathrm{Prob}[A \vee B]$ were not negligible, then from Lemma 5 (i.e., $\mathrm{Prob}[B]$ is negligible) it follows that $\mathrm{Prob}[A \mid \neg B]$ cannot be negligible. So suppose, towards contradiction, that given $k \in K_n$ algorithm L asks a membership query for $(y^{(d(n))}, j)$ for some j with probability at least $1/q(n)$ for a polynomial q and infinitely many $n \in \mathbb{N}$. We show how to derive a predictor D' for F with prediction probability $1/q'(n)$ for a polynomial q' and infinitely many n.

Recall that the query depth of L is $d(n)$. Thus, given that L queries $y_k^{(d(n))}$ and that $y_k^{(0)}, \ldots, y_k^{(d(n))}$ are pairwise different, there exist $i, r \in \{1, \ldots, d(n)\}$ such that L queries $y_k^{(i)}$ in round r without having queried $y_k^{(i-1)}$ in the proceding $r-1$ rounds. Since D' does not necessarily know i and r it tries to guess these values by picking $I, R \in_R \{1, \ldots, d(n)\}$ uniformly at random. D' computes $y_k^{(1)}, \ldots, y_k^{(I-1)}$ via the function oracle and then simulates L until L has output the membership queries for round R. Note that we have $F_k^*(\cdot) = F_k'(\cdot)$ by assumption $k \in K_n$. Let $p_L(n)$ denote the polynomial that bounds the running time of L and thus the number of queries in each round. D' uniformly picks a query (y, j) of the at most $p_L(n)$ queries. The value y will be the guess for $y_k^{(I)} = F_k\left(y_k^{(I-1)}\right)$. If $y_k^{(I-1)}$ has not been among L's queries in the previous rounds, D' outputs the pair $(y_k^{(I-1)}, y)$. With probability at least $1/d^2(n)p_L(n)$, more specifically, if $I = i$ and $R = r$ and $y = y_k^{(I)}$, the value $y_k^{(I)}$ has not been queried previously. If $y_k^{(I-1)}$ has already been queried, D' outputs an arbitrary pair. Assume that $y_k^{(I-1)}$ has not appeared among the queries. Then D' predicts $F_k(y_k^{(I-1)})$ correctly with probability at least $1/d^2(n)p_L(n)q(n)$ for infinitely many n, which is not negligble. The claim follows. $\square$

We conclude that with probability at least $1 - 1/8p(n)$ (for all large n) algorithm L does not query $\left(y_k^{(d(n))}, j\right)$ for any j and that $k \in K_n$. In this case, D is able to answer all queries correctly. After L has stopped and asked for a challenge, D generates a random challenge distributed according to $\mathcal{D}_n$ by picking $x \in_R \{0,1\}^n$ and $j \in_R \{1, \ldots, n\}$. With probability $1-(p_L(n)+1)\cdot 2^{-n}$ (which is greater than $1-1/8p(n)$ for all large n) we have $x \neq y_k^{(d(n))}$ and x has not been queried by L previously. We call such x fresh. Let x be D's challenge, i.e., D is given $F_k(x)$ and $r \in_R \{0,1\}^n$ in random order (Q_0, Q_1). Let ℓ denote L's prediction for $c_k(x, j)$. D outputs a guess $g \in \{0,1\}$ as follows:

- if $\pi_j(Q_0) = \pi_j(Q_1)$ then g is chosen at random
- if $\pi_j(Q_0) \neq \pi_j(Q_1)$ then define g such that $\pi_j(Q_g) = \ell$

We remark that each case occurs with probability $1/2$, depending only on the choice of r, since $\pi_j(r)$ is a random bit. In the former case, D is successful with probability $1/2$. In the latter case, D's guess is correct if and only if ℓ is.

It remains to analyze D's success probability. Denote by $\mathsf{correct}(\ell)$ the event that L classifies random examples correctly, and by SimOK the event that L does not query for $(y_k^{(d(n))}, j)$, that $k \in K_n$ and that x is fresh. By assumption, we have $\mathrm{Prob}[\mathsf{correct}(\ell)] \geq 1/2 + 1/p(n)$ for infinitely many n, where the probability is taken over the choice of the target concept, the random challenge and the coin tosses of L. In our case, we require that L predicts correctly given that the simulation is good. This implies, for instance, that we need to consider L's success probability for keys $k \in K_n$. More generally, we are interested in the conditional probability $\mathrm{Prob}[\mathsf{correct}(\ell) \mid \mathsf{SimOK}]$. As we will show this probability is only slightly smaller than $\mathrm{Prob}[\mathsf{correct}(\ell)]$. This also captures the intuition: If L predicts correctly with probabilility $1/2 + 1/p(n)$, then the fact that the simulation fails cannot contribute significantly to this success probability, because this event is very unlikely. First observe that $\mathrm{Prob}[\neg\, \mathsf{SimOK}] \leq 1/4p(n)$ for all but finite many n by the union bound, since the probability that L queries for $y_k^{(d(n))}$ or that $k \notin K_n$ and the probability that x is not fresh are both negligible and therefore each less than $1/8p(n)$. It follows that

$$\mathrm{Prob}[\mathsf{correct}(\ell) \mid \mathsf{SimOK}] \geq \mathrm{Prob}[\mathsf{correct}(\ell)] - \mathrm{Prob}[\neg\, \mathsf{SimOK}] \geq \frac{1}{2} + \frac{1}{2p(n)}$$

infinitely often. Now we are ready to calculate the advantage of D. Let $\mathsf{case1}$ and $\mathsf{case2}$ denote the events that the first case ($\pi_j(Q_0) = \pi_j(Q_1)$) resp. the other case ($\pi_j(Q_0) \neq \pi_j(Q_1)$) occurs. Recall that the events $\mathsf{case1}, \mathsf{case2}$ are independent of SimOK and $\mathsf{correct}(\ell)$. Then

$$\begin{aligned}
&\mathrm{Prob}[b = g] \\
&\quad \geq \mathrm{Prob}[b = g \wedge \mathsf{SimOK}] \\
&\quad = \mathrm{Prob}[b = g \wedge \mathsf{SimOK} \wedge \mathsf{case1}] \;+\; \mathrm{Prob}[b = g \wedge \mathsf{SimOK} \wedge \mathsf{case2}] \\
&\quad \geq \mathrm{Prob}[b = g \mid \mathsf{SimOK} \wedge \mathsf{case1}] \cdot \mathrm{Prob}[\mathsf{SimOK}] \cdot \mathrm{Prob}[\mathsf{case1}] \\
&\qquad + \mathrm{Prob}[b = g \mid \mathsf{SimOK} \wedge \mathsf{case2}] \cdot \mathrm{Prob}[\mathsf{SimOK}] \cdot \mathrm{Prob}[\mathsf{case2}] \\
&\quad = \mathrm{Prob}[b = g \mid \mathsf{SimOK} \wedge \mathsf{case1}] \cdot \mathrm{Prob}[\mathsf{SimOK}] \cdot \mathrm{Prob}[\mathsf{case1}] \\
&\qquad + \mathrm{Prob}[\mathsf{correct}(\ell) \mid \mathsf{SimOK} \wedge \mathsf{case2}] \cdot \mathrm{Prob}[\mathsf{SimOK}] \cdot \mathrm{Prob}[\mathsf{case2}] \\
&\quad \geq \frac{1}{2} \cdot \left(1 - \frac{1}{4p(n)}\right) \cdot \frac{1}{2} \quad + \quad \left(\frac{1}{2} + \frac{1}{2p(n)}\right) \cdot \left(1 - \frac{1}{4p(n)}\right) \cdot \frac{1}{2} \\
&\quad \geq \frac{1}{2} + \frac{1}{16p(n)}
\end{aligned}$$

for infinitely many $n \in \mathbb{N}$. That is, we obtain a distinguisher with distinguishing advantage that is not negligible. This contradicts the pseudorandomness of F and concludes the proof of Lemma 7. $\square$

We obtain:

Theorem 9 *If one-way functions exist, then for any polynomial $d(n)$ there is a concept class $\mathcal{C}$ and a distribution $\mathcal{D}$ such that $\mathcal{C}$ is not weakly predictable with membership queries in query depth $d(n)$ with respect to $\mathcal{D}$, but can be learned exactly with membership queries in query depth $d(n)+1$.*

3.3 Impossibility of Learning with Membership and Equivalence Queries

It remains to show that adding equivalence queries does not help learning in query depth $d(n)$. The idea is similar to Angluin's well-known technique [1] replacing an equivalence query by a polynomial number of parallel membership queries. In our case this is even much simpler than in general. Assume that L puts an equivalence query for $k' \in \{0,1\}^n$. Then, for a randomly chosen $x \in_R \{0,1\}^n$, we have $F_k^*(x) \neq F_{k'}^*(x)$ with probability at least $1 - 1/q(n) \geq 1/2$ for every polynomial q and sufficiently large n. Otherwise we could use L to construct a successful predictor for pseudorandom functions, because guessing the key is even harder than predicting a single value. Thus, with probability at least $1/2n$ it holds $\pi_j(F_k^*(x)) \neq \pi_j(F_{k'}^*(x))$ for $j \in_R \{1,\ldots,n\}$. If we execute $2n^2$ such membership queries in parallel then with probability at least $1-e^{-n}$ we find a counterexample. Summing over all (at most polynomial) equivalence queries we find counterexamples for all queries with probability at least $1 - \mathrm{poly}(n) \cdot e^{-n}$. Hence, this simulation only fails with negligible probability and we can therefore apply the argument of the previous theorem.

Theorem 10 *If one-way functions exist, then for any polynomial $d(n)$ there is a concept class $\mathcal{C}$ and a distribution $\mathcal{D}$ such that $\mathcal{C}$ is not weakly predictable with membership and equivalence queries in query depth $d(n)$ with respect to $\mathcal{D}$, but can be learned exactly with membership queries in query depth $d(n)+1$.*

The proof of Lemma 7 shows that we do not require $\mathcal{D}$ to be uniform over $\{0,1\}^{n+\lceil \log n \rceil}$. It suffices that the x-part of the random challenge (x,j) appears only with negligible probability among the queries of the learning algorithm (so that x is fresh according to the terminology of the proof of Lemma 7). To formalize this requirement let $\|X_n\| = \max\{\mathrm{Prob}[X = x] \mid x \in \{0,1\}^n\}$ denote the *infinity norm* of a random variable X_n over $\{0,1\}^n$. For a distribution $\mathcal{D}_n$ over $\{0,1\}^{n+\lceil \log n \rceil}$ define a random variable X_n^x by

$$\mathrm{Prob}[X_n^x = x] = \sum_{j=1}^{n} \mathrm{Prob}[(X_n, J_n) = (x,j)],$$

where (X_n, J_n) is distributed according to $\mathcal{D}_n$. Denote the distribution of X_n^x by $\mathcal{D}_n^x$. Then the results of Theorem 10 also holds for any sampleable distribution $\mathcal{D}$ such that $\delta(n) = \|\mathcal{D}_n^x\|$ is negligible in n. We remark that for the uniform distribution we have $\delta(n) = 2^{-n}$. Also note that we do not presume anything about the distribution of the j-part.

4 Applications to Random-Self-Reductions and Coherent Sets

Feigenbaum et al. [16] present a set $\mathcal{L}$ in DSPACE($n^{\beta(n)}$) for any unbounded, nondecreasing function $\beta(n)$ (with $n^{\beta(n)}$ time-constructible) such that $\mathcal{L}$ is adaptively random-self-reducible, while nonadaptive random-self-reductions do not exist. This results holds unconditionally. Assuming NEEE $\not\subseteq$ BPEEE, they show that there exist such sets in NP. This assumption has been reduced to NE $\not\subseteq$ BPE by Hemaspaandra et al. [21]. Combining the idea of Feigenbaum et al. [16] with our result for learning algorithms we obtain the following:

Proposition 11 *Let $\beta(n)$ be an unbounded, nondecreasing function such that $n^{\beta(n)}$ is time-constructible and $n^{\beta(n)} \cdot 2^{-n}$ is negligible. Let $d(n)$ be a fixed polynomial. If one-way functions exist, there is a language $\mathcal{L}$ in DSPACE($n^{\beta(n)}$) such that there is no length-preserving random-self-reduction of query depth $d(n)$ for $\mathcal{L}$, though there exists a deterministic, oblivious, length-preserving random-self-reduction of query depth $d(n) + 1$.*

We remark that $n^{\beta(n)} \cdot 2^{-n}$ is negligible if, for instance, $\beta(n) \cdot \log n < n/2$ for sufficiently large n. This is true for $\beta(n) = \log^* n$.

PROOF. The proof is similar to the proof given in [16]. We view a random-self-reduction given by algorithms σ and ϕ as a single Turing machine M. The choice of $\beta(n)$ ensures that $n^{\beta(n)} > p(n)$ for any polynomial $p(n)$. We can therefore diagonalize against the length-preserving random-self-reductions $M_1, M_2, \ldots$ of query depth $d(n)$. See [4] for more background about diagonalization. The language $\mathcal{L}$ will consist of tuples $(x, j) \in \{0,1\}^{n+\lceil \log n \rceil}$ such that $\pi_j(F_k^*(x)) = 1$ for appropriate key $k \in \{0,1\}^n$. This key of length n will be determined by the diagonalization technique.

We enumerate the length-preserving random-self-reductions $M_1, M_2, \ldots$ of query depth $d(n)$ such that we consider every M_i in connexion with infinitely many complexity parameters $n \in \mathbb{N}$. We call such n related to M_i and denote the set of such n by N_i. For appropriate enumerations of $M_1, M_2, \ldots$ the sets $N_1, N_2, \ldots$ form a partition of $\mathbb{N}$; in fact, standard enumeration techniques work. Assume for the moment that M_i correctly predicts $\pi_j(F_k^*(x))$ with probability at least $2/3$ for *all* $k \in \{0,1\}^n$ and *all* $(x, j) \in \{0,1\}^{n+\lceil \log n \rceil}$ for infinitely many related n's. M_i's running time and therefore the number of queries is bounded above by $n^{\beta(n)}$. Additionally, any query $\sigma(j, x)$ of M_i is distributed independently of x. Thus, if we choose a random input $(x, j) \in_R \{0,1\}^{n+\lceil \log n \rceil}$ and let M_i run on that input, then with probability at most $n^{\beta(n)} \cdot 2^{-n}$ the value (x, j) appears among the queries. By assumption, $n^{\beta(n)} \cdot 2^{-n}$ is negligible. Hence, given that M_i decides membership correctly with probability $2/3$ for all inputs (x, j) and all keys k, it does so without querying about the input and

with probability at least 5/8 for uniformly chosen and sufficiently large input and all keys (of related length n). Similar to the negative result on the learnability of our concept class, we conclude that we can turn M_i into a successful distinguisher for the underlying collection of pseudorandom functions. This follows by choosing a random (x, j) and then simulating M_i with the help of the pseudorandom function oracle. Here we exploit the fact that the reduction is length-preserving so that M_i's queries fall within the domain of the oracle of the distinguisher. From this contradiction we derive that for every sufficiently large n (related to M_i) there exists some key k_0 and input (x_0, j_0) such that M_i fails to predict $\pi_{j_0}(F^*_{k_0}(x_0))$ with probability more than 1/3. Clearly, we can determine such a key k_0 in space $O\left(n^{\beta(n)}\right)$ by exhaustive search. Call the lexicographically smallest of such keys hard for i, n. Add exactly those (x, j) with $\pi_j(F^*_{k_0}(x)) = 1$ to $\mathcal{L}$:

$$\mathcal{L} = \bigcup_{i\in\mathbb{N}} \bigcup_{n\in N_i} \left\{(x, j) \;\middle|\; \pi_j(F^*_{k_0}(x_0)) = 1 \text{ and } k_0 \text{ is hard for } i, n\right\}$$

Since each length-preserving random-self-reduction of query depth $d(n)$ appears at some point in the enumeration, the language $\mathcal{L}$ is not randomly self-reducible in query depth $d(n)$.

The fact that this language is obliviously random-self-reducible is straightforward as we can determine the key k_0 in depth $d(n) + 1$. Then we can easily decide whether the input (x, j) is in $\mathcal{L}$ by computing $\pi_j(F^*_{k_0}(x))$. □

Presuming non-uniformly secure pseudorandom functions we immediately obtain:

Corollary 12 *Let $\beta(n)$ be an unbounded, nondecreasing function and assume that $n^{\beta(n)}$ is time-constructible and that $n^{\beta(n)} \cdot 2^{-n}$ is negligible. Let $d(n)$ be a fixed polynomial. If non-uniformly secure one-way functions exist, there is a language $\mathcal{L}$ in DSPACE($n^{\beta(n)}$) such that there is no length-monotone random-self-reduction of query depth $d(n)$ for $\mathcal{L}$, though there is a deterministic, oblivious, length-preserving random-self-reduction of query depth $d(n) + 1$.*

PROOF. The proof is a straightforward extension of the proof of Proposition 11. Again, if there was a length-monotone random-self-reduction we could construct a polynomial-size circuit family with distinguishing advantage that is not negligible. To answer queries that have smaller length we give the circuit that simulates the random-self-reduction for inputs of length $n + \lceil \log n \rceil$ the first $n - 1$ keys determined by $\mathcal{L}$ for complexity parameters $1, \ldots, n - 1$ as nonuniform advice. □

Beigel and Feigenbaum [7] prove that every random-self-reducible language is weakly coherent. Analyzing their proof it is easy to see that their transformation of a random-self-reduction to a weak examiner preserves the query depth. The negative result for examiners follows as in Proposition 11 because the examiner is not allowed to query about the given input.

Corollary 13 *Let $\beta(n)$ be an unbounded, nondecreasing function. Assume that $n^{\beta(n)}$ is time-constructible and that $n^{\beta(n)} \cdot 2^{-n}$ is negligible. Let $d(n)$ be a fixed polynomial. If one-way functions exist, there is a language $\mathcal{L}$ in DSPACE($n^{\beta(n)}$) that is incoherent for length-preserving examiners of query depth $d(n)$, though there exists a weak, length-preserving examiner of query depth $d(n)+1$.*

Again, this conclusion can be extended to length-monotone examiners assuming non-uniformly secure one-way functions. Unfortunately, we do not know whether the positive result of Corollary 13 also holds for probabilistic polynomial-time examiners instead of weak examiners. But we achieve this using a somewhat stronger assumption, namely the existence of one-way permutations:

Proposition 14 *Let $\beta(n)$ be an unbounded, nondecreasing function and assume that $n^{\beta(n)}$ is time-constructible and that $n^{\beta(n)} \cdot 2^{-n}$ is negligible. Let $d(n)$ be a fixed polynomial. If one-way permutations exist, there is a language $\mathcal{L}$ in DSPACE($n^{\beta(n)}$) that is incoherent for length-preserving examiners of query depth $d(n)$, though there is a deterministic, length-preserving examiner of query depth $d(n)+1$.*

PROOF. Given a one-way permutation we can construct a collection of pseudorandom functions such that $F_n(k,1^n) \neq F_n(k',1^n)$ for $k \neq k'$; see [13]. Similar to the proof of Lemma 5 in Section 3 we conclude that $y_k^{(1)}, \ldots, y_k^{(d(n))} \neq 1^n$ for all but a negligible fraction of the keys $k \in K_n$. Hence, the impossibility result remains valid if we restrict ourself to such keys. Now it also suffices to show the positive result for those keys. Assume that the examiner E is given (x,j) as input. If $x \notin \left\{y_k^{(0)}, \ldots, y_k^{(d(n))}\right\}$ then E can compute k in depth $d(n)+1$ without querying for (x,j) and decide whether $(x,j) \in \mathcal{L}$ by computing $\pi_j(F_k^*(x))$ in polynomial time. Suppose that $x = y_k^{(i)}$ for some i. Then the examiner cannot query for $(y_k^{(i)}, j)$. Fortunately, there are only two possibilities, namely $(x,j) \in \mathcal{L}$ or $(x,j) \notin \mathcal{L}$. E tries both possibilities in parallel and also asks for $F_k(1^n)$ in a single concurrent step. This is possible as 1^n is different from $y_k^{(0)}, \ldots, y_k^{(d(n))}$ by assumption about k. Thus E derives two keys k_0 and k_1 and the value $F_k(1^n)$ in query depth $d(n)+1$. It determines the correct key by computing $F_{k_0}(1^n)$ and $F_{k_1}(1^n)$ in polynomial time and comparing it to the value obtained for $F_k(1^n)$. Given the key k the examiner can decide whether $(x,j) \in \mathcal{L}$. $\square$

Assuming non-uniformly secure one-way permutations we can extend the negative result to length-monotone examiners. An interesting open problem is the question if one can generalize the negative results about random-self-reducible and coherent sets to arbitrary reductions and examiners instead of length--monotone ones. Also, we leave it as an open problem to establish similar results for languages in the polynomial hierarchy.

Acknowledgements

We thank the anonymous referees and the program committee of ALT'98 for valuable and comprehensive comments. We also thank the anonymous referees of TCS who pointed out a bug concerning the positive learnability result and helped to improve the overall presentation by giving excellent comments.

References

[1] D.Angluin: *Learning Regular Sets from Queries and Counterexamples*, Information and Computation, vol. 75, pp. 87–106, 1987.

[2] D.Angluin: *Queries and Concept Learning*, Machine Learning, vol. 2, pp. 319–342, 1988.

[3] D.Angluin, M.Kharitonov: *When Won't Membership Queries Help?*, 23rd ACM Symposium on Theory of Computing, pp. 444–454, 1991.

[4] J.Balcázar, J.Díaz, J.Gabarró: *Structural Complexity I*, Second Edition, Springer-Verlag, 1995.

[5] J.Balcázar, J.Díaz, R.Gavaldà, O.Watanabe: *An Optimal Parallel Algorithm for Learning DFA*, 7th ACM Conference on Computational Learning Theory, 1994.

[6] R.Beigel, J.Feigenbaum: *Improved Bounds on Coherence and Checkability*, Yale University Technical Report, YALEU/DCS/TR–819, 1990.

[7] R.Beigel, J.Feigenbaum: *On Being Incoherent Without Being very Hard*, Computational Complexity, vol. 2, pp. 1–17, 1992.

[8] M.Blum, S.Micali: *How to Generate Cryptographically Strong Sequences of Pseudorandom Bits*, SIAM Journal on Computation, vol. 13, pp. 850–864, 1984.

[9] N.Bshouty: *Exact Learning of Formulas in Parallel*, Machine Learning, vol. 26, pp. 25–42, 1997.

[10] N.Bshouty, R.Cleve: *On the Exact Learning of Formulas in Parallel*, 33rd IEEE Symposium on the Foundations of Computer Science, pp. 513–522, 1992.

[11] N.Bshouty, S.Goldman, H.Mathias: *Noise-Tolerant Parallel Learning of Geometric Concepts*, 8th ACM Conference on Computational Learning Theory, 1995.

[12] B.Berger, J.Rompel, P.Shor: *Efficient NC Algorithms for Set Cover with Application to Learning and Geometry*, 30th IEEE Symposium on the Foundations of Computer Science, pp. 54–59, 1989.

[13] R.Canetti, D.Micciancio, O.Reingold: *Perfectly One-Way Probabilistic Hash Functions*, 30th ACM Symposium on Theory of Computing, 1998.

[14] J.Feigenbaum: *Locally Random Reductions in Interactive Complexity Theory*, DIMACS Series in Discrete Mathematics and Theoretical Computer Science, vol. 13, AMS, pp. 73–98, 1993.

[15] J.Feigenbaum, L.Fortnow: *On the Random-Self-Reducibility of Complete Sets*, 6th Annual IEEE Structure in Complexity Theory Conference, 1991.

[16] J.Feigenbaum, L.Fortnow, C.Lund, D.Spielman: *The Power of Adaptiveness and Additional Queries in Random-Self-Reductions*, Computational Complexity, vol. 4, pp. 158–174, 1994.

[17] J.Feigenbaum, M.Strauss: *An Information-Theoretic Treatment of Random-Self-Reducibility*, 14th Symposium on Theoretical Aspects of Computer Science, Lecture Notes in Computer Science, vol. 1200, Springer-Verlag, pp. 523–534, 1997.

[18] O.Goldreich: *Foundations of Cryptography (Fragments of a Book)*, Department of Computer Science and Applied Mathematics, Weizmann Institute of Science, Rehovot, Israel, 1995.

[19] S.Goldwasser, O.Goldreich, S.Micali: *How to Construct Random Functions*, Journal of ACM, vol. 33, pp. 792–807, 1986.

[20] J.Håstad, R.Impagliazzo, L.Levin, M.Luby: *Construction of Pseudorandom Generator from any One-Way Function*, to appear in SIAM Journal on Computing, preliminary versions in STOC'89 and STOC'90, 1989/1990.

[21] E.Hemaspaandra, A.Naik, M.Ogihara, A.Selman: *P-Selective Sets, and Reducing Search to Decision vs. Self-Reducibility*, Journal of Computer and System Sciences, vol. 53, pp. 194–209, 1996.

[22] R.Impagliazzo, M.Luby: *One-Way Functions are Essential for Complexity Based Cryptography*, 30th IEEE Symposium on Foundations of Computer Science, pp. 230–235, 1989.

[23] M.Kearns, L.Valiant: *Cryptographic Limitations on Learning Boolean Formulae and Finite Automata*, Journal of ACM, vol. 41, pp. 67–95, 1994.

[24] M.Kharitonov: *Cryptographic Hardness of Distribution-Specific Learning*, 25th ACM Symposium on the Theory of Computing, pp. 372–381, 1993.

[25] R.Ostrovsky, A.Wigderson: *One-Way Functions are Essential for Non-Trivial Zero-Knowledge*, Second IEEE Israel Symposium on Theory and Computing Systems, 1993.

[26] L.Pitt, L.Valiant: *Computational Limitations on Learning from Examples*, Journal of ACM, vol. 35, pp. 965–984, 1988.

[27] L.Pitt, M.Warmuth: *Prediction-Preserving Reducibility*, Journal of Computer and System Science, vol. 41, pp. 430–467, 1990.

[28] R.Rivest, Y.L.Yin: *Being Taught can be Faster than Asking Questions*, 8th ACM Conference on Computational Learning Theory, 1995.

[29] C.P.Schnorr: *Optimal Algorithms for Self-Reducible Problems*, 3rd International Colloqium on Automata, Languages, and Programming, pp. 322–347, Edingburgh University Press, 1976.

[30] L.Valiant: *A Theory of the Learnable*, Communications of ACM, vol. 27, pp. 1134–1142, 1984.

[31] J.Vitter, J.Lin: *Learning in Parallel*, Information and Computation, vol. 92, pp. 179–202, 1992.

[32] A.Yao: *Coherent Functions and Program Checkers*, 22nd ACM Symposium on Theory of Computing, pp. 84–94, 1990.

A Bounding the Query Depth and Width

In this section we consider learning algorithms where the query depth *and* the query width are bounded by fixed polynomials $d(n)$ and $w(n)$, respectively. Thus, the total number of queries is bounded by $t(n) = d(n)w(n)$. The concept class that we define is based on well-known techniques using polynomials and interpolation. Feigenbaum et al. [16] also take this approach to separate nonadaptively $(t(n)+1)$-random-self-reducible functions from adaptive $t(n)$-random-self-reductions. We first give an informal overview. Each concept $c \in \mathcal{C}_n$ of our concept class corresponds to a polynomial $P(\xi)$ of degree $t(n)$ over the finite field $\mathrm{GF}(2^n)$; a pair (x, j) is in the concept c if the j-th bit of $P(x)$ equals 1. Every polynomial $P(\xi) \in \mathrm{GF}(2^n)[\xi]$ of degree $t(n)$ can be uniquely identified by $t(n)+1$ values $P(x_i)$ at distinct elements $x_0, \ldots, x_{t(n)} \in \mathrm{GF}(2^n)$. Hence, every polynomial resp. concept can be learned exactly with $n \cdot (t(n)+1)$ membership queries. On the other hand, $t(n)$ or less values $P(x_1), \ldots, P(x_{t(n)})$ reveal no information about $P(y)$ for any $y \neq x_1, \ldots, x_{t(n)}$, i.e., there are $|\mathrm{GF}(2^n)| = 2^n$ polynomials of degree $t(n)$ that are consistent with the values $P(x_i)$. Therefore, any learning algorithm

making at most $t(n)$ queries is only able to classify a random example (x, j) correctly if it either guesses the classification or if x has been queried. The latter event happens with negligible probability, so any learning algorithm can only be correct with probability close to $1/2$.

For the moment we only consider algorithms with membership queries. We show afterwards how to replace equivalance queries by membership queries.

Proposition 15 *For any polynomial $t(n)$ there is a concept class $\mathcal{C}$ and a distribution $\mathcal{D}$ such that $\mathcal{C}$ is not weakly predictable with $t(n)$ membership queries with respect to $\mathcal{D}$, but can be learned exactly with $n \cdot (t(n) + 1)$ membership queries.*

PROOF. We start by showing the negative result. Consider the polynomials of degree $t(n)$ over $\mathrm{GF}(2^n)[\xi]$. The name of each polynomial

$$P_{p_0,\ldots,p_{t(n)}}(\xi) = \sum_{i=0}^{t(n)} p_i \xi^i$$

is the concatenation of its $t(n) + 1$ coefficients $p_0, \ldots, p_{t(n)} \in \mathrm{GF}(2^n)$. Thus, the length of the name is polynomially in n. In the sequel, we sometimes write $P(\xi)$ instead of $P_{p_0,\ldots,p_{t(n)}}(\xi)$ if the coefficients are clear from the context. Define the concept class $\mathcal{C} = (\mathcal{C}_n)_{n\in\mathbb{N}}$ by

$$\mathcal{C}_n = \left\{ c_{p_0,\ldots,p_{t(n)}} \;\middle|\; p_0, \ldots, p_{t(n)} \in \mathrm{GF}(2^n) \right\}$$

where

$$c_{p_0,\ldots,p_{t(n)}} = \left\{ (x, j) \;\middle|\; x \in \mathrm{GF}(2^n) \text{ and } \pi_j(P_{p_0,\ldots,p_{t(n)}}(x)) = 1 \right\}$$

The distribution $\mathcal{D}_n$ is described by picking $x \in_R \mathrm{GF}(2^n)$ and $j \in_R \{1, \ldots, n\}$.

Why is this concept class not weakly predictable with $t(n)$ queries, even for computationally unbounded learning algorithms? Let $c_{p_0,\ldots,p_{t(n)}} \in \mathcal{C}_n$ be the target concept. In advantage of the learning algorithm L we suppose that any of L's queries (x, j) is answered by the full value $P_{p_0,\ldots,p_{t(n)}}(x)$ instead of the j-th bit. After posing at most $t(n)$ queries, a random challenge (x, j) is generated. Given that the x-part of this challenge has not been among the queries, for every value $y \in \mathrm{GF}(2^n)$ there is exactly one polynomial $P^y(\xi)$ of degree $t(n)$ which is consistent with the (modified) answers to the queries, i.e., which might be the target polynomial, and such that $P^y(x) = y$. Hence, if x has not been queried then the learning algorithm cannot decide whether x lies in the target concept or not with probability more than $1/2$. We obtain that the prediction probability of L is at most $1/2 + t(n) \cdot 2^{-n}$.

To learn this class with $n \cdot (t(n)+1)$ queries, we query about the values (x, j) for $x = 0, \ldots, t(n)$ and $j = 1, \ldots, n$. Then we can reconstruct the target concept by Lagrange interpolation and output the name of the concept. The overall running time is polynomially in n. $\square$

It remains to show that adding equivalence queries does not help. Again, we apply Angluin's standard argument and replace each equivalence query by a single membership query. If the learning algorithm L depth puts an equivalence query $p_0^*, \ldots, p_{t(n)}^*$, then we answer this query as follows: Let $P^*(\xi) = P_{p_0^*, \ldots, p_{t(n)}^*}(\xi)$.

- If there has already been a membership query about (x, j) with answer b and it holds that $\pi_j(P^*(x)) \neq b$ then return (x, j) as a counterexample.
- If $P^*(\xi)$ is consistent with the previous membership queries, then we can find some $x \in \mathrm{GF}(2^n)$ that L has not queried yet (e.g., taking the smallest number between 0 and $t(n)$), and return the counterexample $(x, 1)$.

Note that the total number of queries remains unchanged. After having answered $t(n)$ queries in this way, for every value $y \in \mathrm{GF}(2^n)$ there is still a polynomial $P^y(\xi)$ that is consistent with the simulated answers and therefore the desired result follows.