Exact Learning via Teaching Assistants

V. Arvind and N. V. Vinodchandran
Institute of Mathematical Sciences, C.I.T Campus
Madras 600113, India
email: {arvind,vinod}@imsc.ernet.in

Abstract

In this paper we introduce and study a new model of exact learning called the teaching assistant model. The new ingredient in this model, as compared to Angluin's model, is that apart from the learner and teacher there is a third agent called teaching assistant. We compare the teaching assistant model with Angluin's model and show that in this model we can do a fine classification of concept classes with respect to the complexity of exact learning.

In particular, we consider two algebraic concept classes, namely, permutation groups and linear spaces over finite fields. These concept classes can be seen as special subclasses of the concept class of circuits. In Angluin's exact-learning model these concept classes are, just as the concept class 3-CNF, learnable with equivalence queries but not learnable with membership queries. However, we show that in the teaching assistant model permutation groups are exactly learnable with an LWPP-assistant, and linear spaces over finite fields are exactly learnable with an SPP-assistant. As a negative result, we show that if 3-CNFs are exactly learnable with an LWPP-assistant (SPP-assistant), then NP $\subseteq$ LWPP (respectively NP $\subseteq$ SPP).

1 Introduction

A model of learning that has received much attention from researchers in computational learning theory is the exact learning model introduced by Angluin [1]. In this model a learner learns a target concept by querying an honest teacher about the concept. The two types of queries in Angluin's model are membership and equivalence queries. A major area of research in computational learning theory is the classification of different concept classes with respect to the difficulty of learning them in any reasonable learning model. In Angluin's model, one way to quantify the complexity of a concept class is to consider the type and number of queries that a learner has to ask the teacher to learn any concept in the concept class. In this direction many interesting results are known. Angluin [2] showed that polynomially many equivalence queries are insufficient for learning concept classes which have a combinatorial property called approximate fingerprints. Applying this result she showed that deterministic finite automata, context free languages, CNF and DNF are not learnable with polynomially many equivalence queries. In [8], Gavalda proved that the nonexistence of approximate fingerprints is actually a sufficient condition for learnability with polynomially many equivalence queries. In particular, he showed that the concept class of circuits are equivalence-query learnable with the aid of an oracle in polynomial-time hierarchy. This shows that circuits are, in a sense, easier to learn than concept classes like DFAs, NFAs, CFGs, CNFs or DNFs. More recently, Hellerstein et al. [12] have shown that a concept class is polynomial-query learnable with membership and equivalence queries, if and only if the concept class has polynomial-size certificates. It is also to be noted here that Watanabe [16] (see also [17]) used complexity-theoretic ideas to capture the complexity of various query types. He introduced the notion of machine types to quantify the complexity of exact learning via various query types.

The results mentioned above help us in classifying various concepts according to the difficulty of exactly learning them. However, the classification could be finer. For example, how do we compare two concept classes that are both exactly learnable with polynomially many equivalence queries? It could be the case that one of these concepts is easier than the other because the full power of equivalence queries may not be needed to learn one although it is required to exactly learn the other.

The motivation of this paper is to develop a finer criteria for classifying the complexity of exact learning. For example, we show in this paper that the concept class SYM of permutation groups is polynomial-time exactly learnable with equivalence queries. But using the refined learning model of this paper we show that SYM is a concept that is easier to learn than 3-CNFs (which are also polynomial-time exactly learnable with equivalence queries[1]) unless the complexity class NP is included in the complexity class LWPP[1] (which is believed to be unlikely).

The new ingredient in our model is the concept of *teaching assistants*. Intuitively, a teaching assistant for a concept class helps a polynomial-time learner to learn that concept class. In the model the learner communicates only with the teaching assistant, and the teaching assistant in turn makes membership queries to the teacher. By imposing resource bounds on the complexity of the assistant (more precisely, on the resources used by a Turing machine that accepts the assistant) we get different learning models of varying power.

Summary of results

When we compare the teaching assistant model (henceforth TA model) with Angluin's model, we first observe that the TA model with Σ_2^p-assistants[2] are all-powerful in that sense that every honest representation class can be efficiently learnt with Σ_2^p-assistants in the TA model. On the other hand, the TA model with P-assistants is equivalent to learnability with only membership queries in Angluin's model. Furthermore, we show that equivalence and membership queries in Angluin's model can be simulated in the TA model with NP-assistants. But it turns out that the TA model with NP assistants is more powerful than equivalence and membership queries. Indeed, we observe that for no class of teaching assistants can the TA model capture exactly the power of equivalence queries in Angluin's model. The reason is that equivalence queries in Angluin's model give rise to the following well-known

[1]The definition of this class and other related notions used in this paper are given in Section 2.
[2]These assistant classes are formally defined in Section 3.

anomaly: Let $\mathcal{P}$ and $\mathcal{P}'$ be concept classes such that $\mathcal{P}'$ is a subclass of $\mathcal{P}$ (precise definitions are in Section 2). Although it appears reasonable to expect that $\mathcal{P}'$ is no harder to learn than $\mathcal{P}$, such is not the case in Angluin's model [1]: The set of singletons *with the empty concept* is easier to learn than the set of singletons *without the empty concept*. It is known that this anomaly arises from the fact that in Angluin's model equivalence queries must be representations of concepts that *belong* to the concept class. In the TA model this anomaly does not occur since the teaching assistant makes only membership queries to the teacher.

Next, we turn to our main results which demonstrate the TA model's ability to do a fine classification of concept classes. We develop learning algorithms for certain algebraic concept classes and show that these are easier to learn than 3-CNFs in the TA model; this distinction is not brought out in Angluin's model. To this end, we have to consider teaching assistant classes that are weaker than NP-assistants. The two algebraic concept classes we consider are: permutation groups (SYM), and the class of linear spaces over a fixed finite field F_p, (LS(p)). It is easily seen that neither concept class is exactly learnable with polynomially many membership queries in Angluin's model. It is also not hard to see that both these concept classes are polynomial-time learnable with equivalence queries. Consequently, these concept classes are also learnable with NP-assistants in the TA model. Using the algebraic structure of these classes, we show that permutation groups are polynomial-time exactly learnable with an LWPP-assistant, and linear spaces over a finite field F_p are polynomial-time exactly learnable with an SPP-assistant. Here the assistant classes LWPP($\mathcal{P}$) and SPP($\mathcal{P}$) with respect to representation class $\mathcal{P}$ are defined analogous to the complexity classes SPP and LWPP[6]. Furthermore, we show that 3-CNFs are *not* polynomial-time exactly learnable with LWPP-assistants unless NP $\subseteq$ LWPP. Similarly, we show that 3-CNFs are not polynomial-time exactly learnable with SPP-assistants unless NP $\subseteq$ SPP. Both these inclusions would imply that NP is low for the probabilistic class PP which is believed to be unlikely. This shows that even though all these concepts are exactly learnable with equivalence queries, 3-CNFs are harder to learn than class of permutation groups and class of linear spaces over a finite field.

Since SPP is a smaller complexity class than LWPP, the above results indicate that LS(p) is easier to exactly learn than SYM. This is quite according to our intuition because linear spaces are essentially special kinds of abelian groups which are algebraically simpler structures than arbitrary permutation groups.

The layout of the paper is as follows. In Section 2 we give definitions and notation. In Section 3 we give definitions of the new model of exact learning. In Section 4 we compare the new model with Angluin's model and in Sections 5 and 6 we develop efficient learning algorithms for SYM and LS(p) respectively for the TA model. In Sections 7 and 8 we examine the power of the TA model for different teaching assistant classes and also establish some separations between the power of these classes. Figure 1. at the end of the paper summarizes the inclusions and separations of different teaching assistant classes.

2 Notations and preliminary definitions

Fix the finite alphabet $\Sigma = \{0,1\}$. Let $\overline{b}$ denote the complement of a bit b. Let $|X|$ denote the cardinality of a set X. Let $|w|$ denote the length of a string $w \in \Sigma^*$. For sets X_1 and X_2,

$X_1 \triangle X_2$ denotes their symmetric difference and $X_1 \oplus X_2$ denotes the set $0X_1 \cup 1X_2$. Let λ denote the empty string. For $y_1, y_2 \in \Sigma^*$, $y_1 \leq y_2$ denotes that y_1 is lexicographically smaller than y_2. Let $\mathbf{Z}$ denote the set of integers and $\mathbf{N}$ the set of natural numbers.

For any class of languages $\mathcal{L}$, co$\mathcal{L}$ denotes the class of languages L such that $\Sigma^* - L$ is in $\mathcal{L}$. A function $f : \Sigma^* \to \mathbf{Z}$ is *gap-definable* if there is a polynomial-time nondeterministic (in short, NP) machine M such that, for each $x \in \Sigma^*$, $f(x)$ is the difference between the number of accepting paths and the number of rejecting paths of M on input x (denoted by $M(x)$). Let GapP [6] denote the class of gap-definable functions. For each NP machine M let gap_M denote the GapP function defined by it. The corresponding language class PP is defined as follows: A language L is in PP if there is a GapP function f such that: $x \in L$ iff $f(x) > 0$. The counting classes of interest in this paper are UP, SPP and LWPP. A language L is in UP, if there exists a NP machine M accepting L with the promise that M has at most one accepting path on any input. The classes SPP and LWPP are defined using GapP [6]. The class SPP consists of all those languages whose characteristic functions are in GapP. A language L is in LWPP if there is a function $h \in$ FP such that the following function f is in GapP: $f(x) = h(0^{|x|})$ if $x \in L$ and $f(x) = 0$ if $x \notin L$. The inclusions UP $\subseteq$ SPP $\subseteq$ LWPP are a direct consequence of the definitions. We will make frequent use of the following closure properties of GapP (see [6] for details): if f and g are GapP functions then so is $f - g$, which is defined as $(f - g)(x) = f(x) - g(x)$. If $f(i, x)$ is a GapP function for $1 \leq i \leq p(|x|)$, for some polynomial p, then $\prod_{i=0}^{p(|x|)} f(i, x)$ is also a GapP function.

Other definitions from complexity theory used in the paper can be found in [3].

We recall next some definitions from learning theory. We only deal with the learnability of finite concepts. A *representation class* $\mathcal{P}$ is a tuple $\langle R, \mu, p\rangle$ where p is a polynomial, $R \subseteq \Sigma^*$ and μ is a collection $\mu = \{\mu_n\}_{n \in \mathbf{N}}$ such that $\mu_n : R \to 2^{\Sigma^{p(n)}}$ is a many-one mapping. Any element in the range of μ is called a *concept* of $\mathcal{P}$. The set of all concepts of $\mathcal{P}$ is called the *concept class* represented by $\mathcal{P}$. When there is no confusion, we denote the concept class represented by $\mathcal{P}$ also by $\mathcal{P}$. For any n, let $\mathcal{P}_n$ denote the concepts of $\mathcal{P}$ in $\Sigma^{p(n)}$. For any concept $c \in \mathcal{P}_n$ let *size*(c) denote $\min\{|r| : \mu_n(r) = c\}$. Let $\mathcal{P}_{n,m}$ denote the concepts of $\mathcal{P}$ in $\Sigma^{p(n)}$ which have representations of size less than or equal to m. For two representation classes $\mathcal{P} = \langle R, \mu, p\rangle$ and $\mathcal{P}' = \langle R', \mu', p'\rangle$, $\mathcal{P}'$ is said to be a *subclass* of $\mathcal{P}$, denoted by $\mathcal{P}' \subseteq \mathcal{P}$, if $R' \subseteq R$, $p' = p$, and $\mu' = \mu$ when restricted to R'. An *honest* representation class has the following properties: for $y \in \Sigma^*$ and $n \in \mathbf{N}$, it can be checked in polynomial time if y is a valid representation of a concept in $\mathcal{P}_n$. On input $\langle 0^n, r, x\rangle$, it can be checked in polynomial time if x is in $\mu_n(r)$. In this paper, we consider only honest representation classes.

3 Definitions of exact-learning models and variations

We recall necessary definitions to describe Angluin's exact-learning model [1]. Let $\mathcal{P}$ be a representation class. In Angluin's model there is a deterministic learner which can query a teacher who selects a *target* concept $c \in \mathcal{P}_n$ and answers the query according to it. More specifically, the learner α (throughout this paper we use the first few letters of the greek alphabet to denote learners) on input 0^n, outputs a representation $r \in R$ such that $\mu_n(r) = c$, where c is the target concept. In the course of learning, α can make *equivalence queries* and *membership queries* to the teacher. For an equivalence query $h \in R$ made by α the teacher responds with 'Yes' if $\mu_n(h) = c$, or gives a *counterexample* $x \in \mu_n(h) \triangle c$ if $\mu_n(h) \neq c$. For a membership query $x \in \Sigma^{p(n)}$ made byα the teacher responds with 'Yes' if $x \in c$ and 'No' if $x \notin c$.

We also consider some variations of Angluin's original model.

Firstly, in the present paper we restrict ourselves to *bounded learnability* which we next define. (A similar definition of bounded learnability can be found in [17].)

Definition 3.1 In *bounded learnability*, the learner is given, as part of the input, an upper bound on the size of the target concept. More precisely, for a representation class $\mathcal{P}$, the teacher selects a target concept $c \in \mathcal{P}_n$ and the learner α takes as input a pair $\langle 0^n, 0^l \rangle$, and if $size(c) \leq l$, the learner has to output a representation $r \in R$ such that $\mu_n(r) = c$.

The learner's output is unspecified if $size(c) > l$. Notice that bounded learnability is a general notion that can be applied to not merely Angluin's model but any model of exact learning.

We also consider computationally more powerful learners than the deterministic learners of Angluin's model. Nondeterministic learners are useful technical tools in some of our proofs. A *nondeterministic single valued* learner on any input $\langle 0^n, 0^l \rangle$ has accepting and rejecting nondeterministic guess paths, such that the learner outputs a representation r of the target concept on *all* accepting paths and it outputs nothing on the rejecting paths. The learner must have at least one accepting path. An *unambiguous single valued* learner on input $\langle 0^n, 0^l \rangle$ has to output a representation of the target concept on *exactly* one of its nondeterministic guess paths and nothing on any other path. In all the above cases, if the size of the target concept is greater than l, then the behavior of the learner is unspecified.

When we want to measure the query complexity (in an information-theoretic sense) of learning a representation class, we allow the learner to augment its computational power by having access to an oracle, which is a prespecified language $A \subset \Sigma^*$. Notice that the oracle A aids the learner only in computation, not in retrieving information about the target concept. The queries to the teacher can be distinguished from the queries to A in a standard manner by having different query states for them in the learning algorithm.

We now look at some standard complexity measures on the resources of the learner. Let $\mathcal{P}$ be a representation class. Let α be a learner (any of the type defined above) learning $\mathcal{P}$. For an input 0^n, the *query complexity* of a learner is the sum of the lengths of equivalence queries, membership queries that α has written down on its oracle tape, and the counterexamples received. Notice that, intuitively the query complexity measures the amount of information transfer between the teacher and the learner. On the other hand, the *time complexity* of α is the number of time steps it takes before it outputs a representation of the target concept. Let $\mathcal{M}$ denote a deterministic, nondeterministic single-valued, or unambiguous single-valued learner. Then $\mathcal{P}$ is said to be polynomial-query $\mathcal{M}$-learnable, if there are a polynomial q and a $\mathcal{M}$ learner α learning $\mathcal{P}$, such that for all n, and for all $c \in \mathcal{P}_n$, the query complexity of learner α learning the target concept c, on input $\langle 0^n, 0^l \rangle$ is bounded by $q(n + l)$. $\mathcal{P}$ is said to be polynomial-time $\mathcal{M}$-learnable, if there are a polynomial q and an $\mathcal{M}$ learner α learning $\mathcal{P}$, such that for all n, and for all $c \in \mathcal{P}_n$, the time complexity of learner α

learning the target concept c, on input $\langle 0^n, 0^l \rangle$ is bounded by $q(n+l)$. It is easy to see that if $\mathcal{P}$ is polynomial-query $\mathcal{M}$-learnable then it is polynomial-time $\mathcal{M}$-learnable with access to some computational oracle A. We denote FP-learnable, ZPP-learnable, NPSV-learnable and UPSV-learnable for polynomial-time deterministic learnable, polynomial-time non-deterministic single-valued learnable and polynomial-time unambiguous single-valued learnable, respectively.

Remark. An explanation for considering bounded learnability is in order. We restrict ourselves to bounded learnability because it is difficult to deal with learners whose running time is sensitive to the output, particularly when we consider nondeterministic learners. For instance, consider (in the unbounded learning setting) a nondeterministic learner α learning a representation class $\mathcal{P}$. Let α take 0^n as input and run in time $q(n+size(c))$ where $c \in \mathcal{P}_n$ is the target concept. We can ensure that nondeterministic computation paths of α which outputs a representation of c are of length $q(n+size(c))$. But on computational paths where α is not going to output cannot be timed to run $t(n+size(c))$ steps, because the machine simply does not know when to stop on that path.

Bounded and unbounded learnability coincide for those representation classes $\mathcal{P}$ which have representations of size bounded by a polynomial in n for all concepts in $\mathcal{P}_n$, (in a complexity theory sense $\mathcal{P}$ is a subclass of the nonuniform complexity class P/poly).

3.1 The new teaching-assistant model

We introduce now the new model of this paper. A central idea in this model is the notion of *teaching assistants*. The learner, instead of communicating directly with the teacher, communicates with a predefined teaching assistant to learn the target concept. The teaching assistant actually communicates with the teacher to retrieve information about the target concept. Before formally defining teaching assistants, we give some motivation. In Angluin's model of exact learning, the learner communicates with the teacher in order to learn a concept, where the teacher is of unbounded computational power. Two standard query types that are studied are equivalence queries and membership queries. It is not hard to see that an equivalence query can be replaced by a sequence of queries to an NP oracle (for prefix-searching for a counterexample), which in turn uses membership queries to the concept (see Theorem 4.3 for a formal proof). Intuitively, we can think of the NP oracle as an intermediate agent between the learner and the teacher. We term this agent as the teaching assistant. In this setting, the learner makes queries to the assistant and the assistant, in turn, makes membership queries to the teacher. More precisely, a teaching assistant contains strings which encode information about the representation class. Assistants are of bounded computational power and help the learner by communicating with the teacher with only membership queries. The power of communication between the assistant and teacher depends on the computational power of the assistant. In other words, if a concept class is learnable with a teaching assistant with restricted computational power it intuitively implies that the concept class is relatively easy to learn. We give now the formal definition.

Definition 3.2 Let $\mathcal{P}$ be a representation class. Let $Q : \mathbf{N} \times \mathbf{N} \times \Sigma^* \times 2^{\Sigma^*} \to \{0,1\}$ be a predicate. A tuple $\langle n, l, x, c \rangle \in \mathbf{N} \times \mathbf{N} \times \Sigma^* \times 2^{\Sigma^*}$, is said to be $\mathcal{P}$-valid if $c \in \mathcal{P}_n$ and

$size(c) \leq l$. The *teaching assistant* defined by Q w. r. t. the representation class $\mathcal{P}$ is the set $L(\mathcal{P}) = \{\langle n, l, x, c\rangle | \langle n, l, x, c\rangle$ is P-valid and $Q(n, l, x, c)\}$.

In the above definition notice that, consistent with the notion of bounded leanability (see Definition 3.1), we have the parameter l as part of $\mathcal{P}$-valid tuples.

Now we formalize learning via teaching assistants. Let $\mathcal{P} = \langle R, \mu, p\rangle$ be a representation class and $L(\mathcal{P})$ be a teaching assistant for $\mathcal{P}$. As in Angluin's model, learners are deterministic oracle Turing machine transducers. The learning algorithm comprises of a learner and teaching assistant pair. The teacher selects a target concept $c \in \mathcal{P}_n$. The learner α, on input $\langle 0^n, 0^l\rangle$, has to output a representation $r \in R$ such that $\mu_n(r) = c$, provided $size(c) \leq l$. The learner α can query its teaching assistant, say $L(\mathcal{P})$. Notice that when α wants to query the teaching assistant about $\langle n, l, x, c\rangle$, it is enough that α communicates x to the teaching assistant, since other parameters are implicit. For such a query x by the learner, the learner receives the answer 'Yes' if $\langle n, l, x, c\rangle \in L(\mathcal{P})$ and 'No' if $\langle n, l, x, c\rangle \notin L(\mathcal{P})$. $\mathcal{P}$ is said to be learnable with assistant $L(\mathcal{P})$ if there is a learner α for $\mathcal{P}$ with queries to $L(\mathcal{P})$ as teaching assistant and we say α learns $\mathcal{P}$ with teaching assistant $L(\mathcal{P})$.

Let $\mathcal{P}$ be a representation class. Let α be a learner learning $\mathcal{P}$ with a teaching assistant $L(\mathcal{P})$. For an input $\langle 0^n, 0^l\rangle$, the *time complexity* of a learner is the number of steps it takes before it writes down a representation of the target concept. $\mathcal{P}$ is said to be deterministic polynomial-time learnable (FP-learnable) with a teaching assistant $L(\mathcal{P})$, if there is a polynomial q and a deterministic learner α learning $\mathcal{P}$ with $L(\mathcal{P})$ such that for all n, and for all $c \in \mathcal{P}_n$, on input $\langle 0^n, 0^l\rangle$, the running time of learner α learning c, is bounded by $q(n + l)$.

In order to quantify the complexity of teaching assistants we define certain classes of teaching assistants with respect to a given representation class. These classes are analogous to some well-studied complexity classes. Our motivation for considering the teaching assistant classes defined below is to pinpoint the exact learnability of some algebraic concept classes, and to carefully compare the teaching assistant (TA) model with Angluin's model. In subsequent sections we also investigate the interrelationship between these teaching assistant classes and show several separations.

Let $\mathcal{P}$ be a representation class and $L(\mathcal{P})$ be a teaching assistant.

Definition 3.3

1. $L(\mathcal{P})$ is said to be in the class $\Sigma_2^p(\mathcal{P})$ if there exists a polynomial-time Σ_2 oracle machine which for any $\mathcal{P}$-valid tuple $\langle n, l, x, c\rangle$, on input $\langle 0^n, 0^l, x\rangle$ uses c as oracle and accepts $\langle 0^n, 0^l, x\rangle$ if and only if $\langle n, l, x, c\rangle \in L(\mathcal{P})$

2. $L(\mathcal{P})$ is said to be in the class $\mathrm{P}(\mathcal{P})$ if there exists a polynomial-time deterministic oracle Turing machine M which for any $\mathcal{P}$-valid tuple $\langle n, l, x, c\rangle$, on input $\langle 0^n, 0^l, x\rangle$ uses $c \in \mathcal{P}_n$ as oracle and accepts $\langle 0^n, 0^l, x\rangle$ if and only if $\langle n, l, x, c\rangle \in L(\mathcal{P})$.

3. $L(\mathcal{P})$ is said to be in the class $\mathrm{NP}(\mathcal{P})$ if there exists a polynomial-time nondeterministic oracle Turing machine M which for any $\mathcal{P}$-valid tuple $\langle n, l, x, c\rangle$, on input $\langle 0^n, 0^l, x\rangle$ uses $c \in \mathcal{P}_n$ as oracle and accepts $\langle 0^n, 0^l, x\rangle$ if and only if $\langle n, l, x, c\rangle \in L(\mathcal{P})$.

4. $L(\mathcal{P})$ is said to be in the class $\mathrm{NP}(\mathcal{P}) \cap \text{co-NP}(\mathcal{P})$ if there exists a polynomial-time non-deterministic oracle Turing machine M which for any $\mathcal{P}$-valid tuple $\langle n, l, x, c\rangle$, on any path outputs 'accept', 'reject' or '?' such that: on input $\langle 0^n, 0^l, x\rangle$ uses c as oracle and outputs 'accept' on at least one of its paths and does not output 'reject' on any path, if $\langle n, l, x, c\rangle \in L(\mathcal{P})$ and outputs 'reject' on at least one of its paths and does not output 'accept' on any path, if $\langle n, l, x, c\rangle \notin L(\mathcal{P})$.

5. $L(\mathcal{P})$ is said to be in the class $\mathrm{UP}(\mathcal{P})$ if there exists a polynomial-time nondeterministic oracle Turing machine M which for any $\mathcal{P}$-valid tuple $\langle n, l, x, c\rangle$, on input $\langle 0^n, 0^l, x\rangle$ uses c as oracle and accepts $\langle 0^n, 0^l, x\rangle$ if and only if $\langle n, l, x, c\rangle \in L(\mathcal{P})$ with the promise that M has at most one accepting path on any input.

6. $L(\mathcal{P})$ is said to be in the class $\mathrm{UP}(\mathcal{P}) \cap \text{co-UP}(\mathcal{P})$ if there exists a polynomial-time nondeterministic oracle Turing machine M which for any $\mathcal{P}$-valid tuple $\langle n, l, x, c\rangle$, on any path outputs 'accept', 'reject' or '?' such that: on input $\langle 0^n, 0^l, x\rangle$ uses c as oracle and outputs 'accept' on a unique path and does not output 'reject' on any path if $\langle n, l, x, c\rangle \in L(\mathcal{P})$, and outputs 'reject' on a unique path and does not output 'accept' on any path, if $\langle n, l, x, c\rangle \notin L(\mathcal{P})$.

7. $L(\mathcal{P})$ is said to be in the class $\mathrm{SPP}(\mathcal{P})$ if there exists a polynomial-time nondeterministic oracle Turing machine M which for any $\mathcal{P}$-valid tuple $\langle n, l, x, c\rangle$, on input $\langle 0^n, 0^l, x\rangle$ uses c as oracle and produces a *gap*=1 if $\langle n, l, x, c\rangle \in L(\mathcal{P})$ and *gap*=0 if $\langle n, l, x, c\rangle \notin L(\mathcal{P})$.

8. $L(\mathcal{P})$ is said to be in the class $\mathrm{LWPP}(\mathcal{P})$ if there exists a polynomial-time computable function f and a polynomial time nondeterministic oracle Turing machine M which for any $\mathcal{P}$-valid tuple $\langle n, l, x, c\rangle$, on input $\langle 0^n, 0^l, x\rangle$ uses c as oracle and produces a *gap*=$f(n)$ if $\langle n, l, x, c\rangle \in L(\mathcal{P})$ and *gap*=0 if $\langle n, l, x, c\rangle \notin L(\mathcal{P})$.

In all the above definitions, the behavior of machines accepting the teaching assistants is not specified on inputs which are not $\mathcal{P}$-valid.

The following containments follow directly from the above definitions.

Proposition 3.4 *For any representation class* $\mathcal{P}$

- $\mathrm{P}(\mathcal{P}) \subseteq \mathrm{UP}(\mathcal{P}) \cap \text{co-UP}(\mathcal{P}) \subseteq \mathrm{UP}(\mathcal{P}) \subseteq \mathrm{SPP}(\mathcal{P}) \subseteq \mathrm{LWPP}(\mathcal{P})$.
- $\mathrm{P}(\mathcal{P}) \subseteq \mathrm{UP}(\mathcal{P}) \cap \text{co-UP}(\mathcal{P}) \subseteq \mathrm{NP}(\mathcal{P}) \cap \text{co-NP}(\mathcal{P}) \subseteq \mathrm{NP}(\mathcal{P}) \subseteq \Sigma_2^p(\mathcal{P})$

Similar to Angluin's model, in order to measure the query complexity of teaching assistants (in an information-theoretic sense) we define teaching assistant classes equipped with computational oracles which aid only in computation and not in information retrieval from the teacher. For any of the above-defined assistant classes $\mathcal{C}$, we denote the corresponding assistant class equipped with computational oracle A by $\mathcal{C}^A$. For instance, for a representation class $\mathcal{P}$, the assistant class $\mathrm{NP}(\mathcal{P})$ equipped with oracle A is denoted by $\mathrm{NP}^A(\mathcal{P})$.

Finally, we give the formal definition of learnability via teaching assistants.

Definition 3.5 Let $\mathcal{C}$ denote any of the above-defined assistant classes (possibly relativized). We say a representation class $\mathcal{P}$ is FP-learnable with a $\mathcal{C}$-assistant if there exists a teaching assistant $L(\mathcal{P}) \in \mathcal{C}(\mathcal{P})$ and an FP-learner α, such that α learns $\mathcal{P}$ with the teaching assistant $L(\mathcal{P})$.

4 Comparison with Angluin's model

In this section we first prove a universal upper bound on learnability of any representation class. Then we show the relationship between Angluin's model and the TA model for specific assistant classes.

Theorem 4.1 *For any representation class $\mathcal{P}$, $\mathcal{P}$ is* FP*-learnable with a Σ_2^p-assistant.*

Proof. Let $\mathcal{P} = \langle R, \{\mu_n\}_{n\geq 1}, p\rangle$ be any representation class. Define a teaching assistant as follows: $L(\mathcal{P}) = \{\langle n, l, x, c\rangle \text{ is } \mathcal{P}\text{-valid} \mid \exists y \text{ such that } |xy| \leq l \text{ and } \mu_n(xy) = c\}$. We define a Σ_2^p machine M accepting $L(\mathcal{P})$. M on input $\langle 0^n, 0^l, x\rangle$ existentially guesses y of length bounded by $l - |x|$ and then universally verifies whether $\mu_n(xy) = c$ by universally guessing a string $z \in \Sigma^{p(n)}$ and verifying whether $z \in \mu_n(xy)$ if and only if $z \in c$. Since $\mathcal{P}$ is honest, checking membership in $\mu_n(xy)$ can be done in polynomial time, given xy. Membership of z in c can be done by querying the oracle c.

Now we give the description of the learner α for $\mathcal{P}$ with teaching assistant $L(\mathcal{P})$. Let $c \in \mathcal{P}_n$ be the target concept and l be such that $size(c) \leq l$.

Σ_2^p-LEARNER $\alpha(0^n, 0^l)$
$i \leftarrow l$
while $\langle n, i-1, \lambda, c\rangle \in L(\mathcal{P})$
    do $i \leftarrow i - 1$
$x \leftarrow \lambda$
for $j \leftarrow 1$ **to** i
    do if $\langle n, i, x1, c\rangle \in L(\mathcal{P})$
        then $x \leftarrow x1$
        else $x \leftarrow x0$
Output x

■

We now compare the power of Angluin's model with the TA model. We show that membership queries are characterized by P-assistants. On the other hand, the combination of membership and equivalence queries can be simulated with NP-assistants. However, we show that NP-assistants are strictly more powerful than the combination of membership and equivalence queries. In fact we prove in this section that it is impossible to capture the power of equivalence queries in the TA model.

Theorem 4.2 *For any representation class $\mathcal{P}$, it is* FP*-learnable with membership queries if and only if it is* FP*-learnable with a* P*-assistant.*

Proof. Let α be a FP-learner for $\mathcal{P} = \langle R, \mu, p \rangle$ where $\mu = \{\mu_n\}_{n \geq 1}$ which uses membership queries to the teacher. Define a teaching assistant $L(\mathcal{P})$ as follows. $L(\mathcal{P}) = \{\langle n, l, x, c \rangle | x \in c\}$. It is obvious that $L(\mathcal{P}) \in \mathrm{P}(\mathcal{P})$. Let $c \in \mathcal{P}_n$ be the target concept and let l be such that $size(c) \leq l$. Consider a learner α' which behaves as follows. α' on input $\langle 0^n, 0^l \rangle$ simulates α on input$\langle 0^n, 0^l \rangle$. Whenever α makes a membership query x, α' makes a query $\langle n, l, x, c \rangle$ to $L(\mathcal{P})$ and treats the answer as answer to the membership query for α and continues the simulation of α. It is easily verified that α' outputs a representation of c in time polynomial in $n + l$, if and only if α outputs a representation of c in time polynomial in $n + l$.

To show the other containment, let β be a FP-learner which learns $\mathcal{P}$ with teaching assistant $L(\mathcal{P}) \in \mathrm{P}(\mathcal{P})$. Let M be the polynomial-time oracle Turing machine witnessing $L(\mathcal{P}) \in \mathrm{P}(\mathcal{P})$. Let $c \in \mathcal{P}_n$ be the target concept and let l be such that $size(c) \leq l$. Consider a learner β' which on input $\langle 0^n, 0^l \rangle$ and c as target concept, simulates β. Whenever β makes a query $\langle n, l, x, c \rangle$ to $L(\mathcal{P})$, β' writes down $\langle 0^n, 0^l, x \rangle$ on its work tape and start simulating M on input $\langle 0^n, 0^l, x \rangle$. During this simulation of M, whenever M makes a membership query y to c, β' makes a membership query y to the teacher and treats the answer to this as answer to the query y of M. Now, β' outputs a representation of c in time polynomial in $n + l$, if and only if β outputs the same representation of c in time polynomial in $n + l$. ■

Theorem 4.3 *For any representation class $\mathcal{P}$, If $\mathcal{P}$ is* FP*-learnable with equivalence and membership queries, then $\mathcal{P}$ is* FP*-learnable with an* NP*-assistant.*

Proof. Let α be a learner for $\mathcal{P} = \langle R, \{\mu_n\}_{n \geq 1}, p \rangle$ which uses membership and equivalence queries to the teacher. Define a teaching assistant $L(\mathcal{P})$ as follows.

$L(\mathcal{P}) = \{\langle n, l, x, c \rangle |$ if $x = 1y$ implies $y \in c$, and if $x = 0y\#z$ there is $w \in \Sigma^{p(n)-|y|}$ such that $yw \in c \triangle \mu_n(z)\}$.

It is easy to see that $L(\mathcal{P}) \in \mathrm{NP}(\mathcal{P})$, since $\mathcal{P}$ is honest. Now, consider the following machine α'. Let $c \in \mathcal{P}_n$ be the target concept and l be such that $size(c) \leq l$. Consider a learner α' which behaves as follows. α' on input $\langle 0^n, 0^l \rangle$ simulates α on input $\langle 0^n, 0^l \rangle$. Whenever α makes a membership query x, α' makes a query $\langle n, l, 1x, c \rangle$ to $L(\mathcal{P})$ and takes the answer as answer to the membership query of α and proceeds. Whenever α makes an equivalence query x, α' first makes $\langle n, l, 0\#x, c \rangle$ to $L(\mathcal{P})$. If the answer to this query is 'No', then α' outputs x. Otherwise α' uses $L(\mathcal{P})$ to prefix search for a string $u \in c \triangle \mu_n(x)$ and uses u as counterexample and proceeds. It is easy to see that α' outputs a representation of c in time polynomial in $n + l$, if and only if α outputs a representation of c in time polynomial in $n + l$. ■

As mentioned in the introduction, equivalence queries gives rise to the following anomaly in Angluin's model [1].

Theorem 4.4 *[1] There exist honest representation classes $\mathcal{P}$ and $\mathcal{P}'$ such that $\mathcal{P}'$ is a subclass of $\mathcal{P}$, and $\mathcal{P}$ is* FP*-learnable with equivalence queries but $\mathcal{P}'$ is not* FP*-learnable with equivalence and membership queries in Angluin's model.*

Remark. As observed in the introduction, the TA model does not have the above anomaly. It follows from the definitions of a subclass and learning with teaching assistants that if $\mathcal{P}$

is FP-learnable with a $\mathcal{C}$-assistant ($\mathcal{C}$ is any of the above-defined teaching assistant classes) then $\mathcal{P}'$ is also FP-learnable with a $\mathcal{C}$-assistant. Hence, any upper bound on learning $\mathcal{P}$ is an upper bound on learning $\mathcal{P}'$ and any lower bound on learning $\mathcal{P}'$ is a lower bound on learning $\mathcal{P}$. We summarize this as a proposition.

Proposition 4.5 *Let $\mathcal{P}$ and $\mathcal{P}'$ representation classes such that $\mathcal{P}'$ is a subclass of $\mathcal{P}$. Let $\mathcal{C}$ is any of the above-defined teaching assistant classes. Then if $\mathcal{P}$ is* FP-*learnable with a $\mathcal{C}$-assistant, $\mathcal{P}'$ is also* FP-*learnable with a $\mathcal{C}$-assistant.*

¿From Proposition 4.5, and Theorem 4.4 we have the following corollary.

Corollary 4.6 *There is no teaching assistant class $\mathcal{C}$ such that the TA model captures equivalence query learnability in Angluin's model.*

5 Learning permutation groups

The goal of this section is to develop a polynomial-time exact learning algorithm in the TA model with LWPP-assistants for the concept class SYM. We recall first some group theory. The reader can find details in textbooks (e.g. [11]).

Definition 5.1 [11] A *group* is a nonempty set G endowed with an associative binary operation $*$. There is an *identity* element $e \in G$ with the property: $x * e = e * x = x$, for all $x \in G$. Each $x \in G$ has a unique *inverse* $x^{-1} \in G$, i.e. $x * x^{-1} = x^{-1} * x = e$.

When there is no ambiguity we denote the composition $x * y$ simply by xy.

In the sequel let G denote a finite group. The *order* of G is the cardinality $|G|$ of G. For $g \in G$, the *order* of g (denoted $o(g)$) is the smallest positive integer such that $g^{o(g)} = e$, where e is the identity of G. A subset H of G is a *subgroup* of G (denoted $H < G$) if H is a group under the group operation of G. Let G be a group and $S \subseteq G$. The group *generated by S* is the smallest subgroup of G containing S and is denoted by $\langle S \rangle$. S is a *generator set* for G if $G = \langle S \rangle$. A subgroup of G is that is generated by a single element is called a cyclic subgroup of G. For $g \in G$ and $H < G$, $Hg = \{hg \mid h \in H\}$ is a *right coset* of H in G and $gH = \{gh \mid h \in H\}$ is a *left coset* of H in G.

The *symmetric group* S_n consists of all permutations on {1,. . . ,n} with permutation composition as group operation. Subgroups of S_n are called *permutation groups*. An m-cycle is a permutation $\sigma \in S_n$ such that there are indices $i_1, \ldots, i_m$, $1 \le i_j \le n$: $\sigma(i_j) = i_{j+1 (\mathrm{mod}\ m)}$, and $\sigma(i) = i$ if $i \ne i_j$ for $1 \le j \le m$. We call σ a cycle, if it is an m-cycle for some $m \le n$. Recall that any $\sigma \in S_n$ can be written as a composition of cycles. Finally, An *m-cyclic subgroup* of S_n is a cyclic subgroup of S_n that is generated by an m-cycle. Notice that not every cyclic subgroup of S_n is m-cyclic.

Now we formally define the representation classes of interest to us in this section. First we explain how we encode permutations. Let the n-tuple $\sigma = (i_1, \ldots, i_n)$ represents the permutation $j \to i_j$ in S_n. Let $m = \lceil \log n \rceil$ and the binary representation of i_j be $x_{j1} \ldots x_{jm}$. Then we represent σ with the binary string

$$x_{11}x_{11} \ldots x_{1m}x_{1m}01x_{21}x_{21} \ldots x_{2m}x_{2m}01 \ldots 01x_{n1}x_{n1} \ldots x_{nm}x_{nm}$$

of length $2nm + 2(n-1)$. It is clear that the group operations can be performed in time linear in n.

Let SYM $= \langle R, \{\mu_n\}_{n\geq 1}, p\rangle$ denote the representation class of permutation groups where $R = \cup_{n\geq 1} R_n$, and $r \in R_n$ encodes a set of permutations of S_n, namely, it is a concatenation of some strings, each encoding a permutation as explained above. For $r \in R_n$, $\mu_n(r)$ is the subgroup of S_n generated by permutations encoded in r. Here, $p(n) = 2n\lceil \log n\rceil + 2(n-1)$.

Remark. We recall Lagrange's theorem which states that if G is a finite group and $H < G$ then $|G|$ is divisible by $|H|$. It can be deduced from Lagrange's theorem that every subgroup $G < S_n$ has a generator set of size at most $n \log n$. Hence, each concept $c \in \mathrm{SYM}_n$ has a representation of size polynomial in n. As a consequence, bounded learnability coincides with unbounded learnability when we consider polynomial-time learnability of subclasses of SYM. So, we can do away with the parameter 0^l which denoted the size bound on the target concept's representation, both from the learner's input and from the tuples defining teaching assistants.

Theorem 5.2 SYM *is* FP*-learnable with only equivalence queries. Hence* FP*-learnable with an* NP*-assistant.*

Proof. Consider the following learner for learning a subgroup $G < S_n$.

EQUIV-LEARN(0^n)

$S \leftarrow \{e\}$
while Answer to query S is 'No'
    do $g \leftarrow$ COUNTER EXAMPLE
        $S \leftarrow S \cup \{g\}$
return S

For the proof of correctness of the above algorithm, we need the following claim. The claim follows from Lagrange's theorem about finite groups.

Claim 5.2.1 *Let G and H be two groups such that $H < G$. Let $g \in G$ but $g \notin H$. Then $|\langle H, g\rangle| \geq 2|H|$.*

Consider the i^{th} iteration of the **while** loop. Let $S = \{g_1, \ldots, g_{i-1}\}$ be the set constructed so far. It follows from the fact that G is a group and the identity $e \in G$, $\langle S\rangle < G$. So the counterexample g_i given by the teacher at the i^{th} iteration is in G but not in $\langle S\rangle$. Since each addition of elements in to the set S grows the size of the group by at least twice the previous size, it follows that the iteration stops after polynomially many steps. ∎

To prove the next theorem, we need to recall some permutation group theory. Let $G < S_n$ and let $G^{(i)}$ denote the subgroup $\{g \in G \mid \forall j \leq i : g(j) = j\}$ of G. Notice that this gives rise to the following chain of subgroups of G: $\{e\} = G^{(n)} < G^{(n-1)} < \ldots < G^{(0)} = G$. Consider the left cosets of $G^{(i)}$ in $G^{(i-1)}$. It is easy to verify that two elements $g_1, g_2 \in G^{(i-1)}$ are in the same left coset of $G^{(i)}$ iff $g_1(i) = g_2(i)$. A set $T_i = \{g_{i1}, g_{i2}, \ldots, g_{id_i}\}$ of distinct (left) coset representatives of $G^{(i)}$ in $G^{(i-1)}$ is called a *transversal*. Notice that $G^{(i-1)} = \bigcup_{j=1}^{d_i} g_{ij}G^{(i)}$, where $d_i = |T_i|$. The set $\bigcup_{i=1}^{n} T_i$ generates G and we call it a *canonical generator set* for G.

Let $G < S_n$ and $X \subseteq [n]$. Let $G_{[X]}$ denote the subgroup $\{g \in G \mid g(i) = i \quad \forall i \in X\}$. For any $\sigma \in S_n$ let $G_{[\sigma,X]}$ denote the set $\{g \in G \mid g(i) = \sigma(i) \quad \forall i \in X\}$. Notice that $G_{[\sigma,X]} = \sigma G_{[X]}$.

In the following lemma we state some useful group-theoretic facts which are easy to derive.

Lemma 5.3 *Let $G < S_n$.*

1. *$|G^{(i)}| = \prod_{j>i} d_j$ for each i, $1 \leq i \leq n$.*
2. *The number of canonical sets of generators for G is $\prod_{i=1}^{n} |G^{(i)}|^{d_i}$.*
3. *$|G_{[\sigma,X]}| = |G_{[X]}|$, $\forall X \subseteq [n]$ and $\sigma \in S_n$.*

Theorem 5.4 *The representation class* SYM *is* FP*-learnable with an* LWPP*-assistant.*

Proof. Let G be the target group. Define the assistant $L(\mathrm{SYM})$ as:
$L(\mathrm{SYM}) = \{\langle 0^n, k, i, j_1, \ldots, j_k, G\rangle | \exists g \in G$ such that g fixes 1 to i; $g(i+l) = j_l$ for $1 \leq l \leq k\}$. We first show that this set defines an LWPP-assistant by constructing a nondeterministic machine which produces zero *gap* if the input string is not in the set and produces a $gap = (n!)^{2n^2+1}$ if it is in the set. Then we give a FP-learner which uses $L(\mathrm{SYM})$ as assistant for the class SYM.

Claim 5.4.1 *There is a nondeterministic oracle machine M such that given a group $G < S_n$ as oracle it has the following behavior:*

- *If $\langle 0^n, k, i, j_1, \ldots, j_k, G\rangle \in L(\mathrm{SYM})$ then M on input $\langle 0^n, k, i, j_1, \ldots, j_k\rangle$ produces a $gap = (n!)^{2n^2+1}$.*
- *If $\langle 0^n, k, i, j_1, \ldots, j_k, G\rangle \notin L(\mathrm{SYM})$ then M on input $\langle 0^n, k, i, j_1, \ldots, j_k\rangle$ produces zero gap.*

Proof For ease of exposition, we will give a structured description of machine M. Intuitively speaking, M simulates another machine N on all computation paths on which a "correct" guess is made. We need to introduce the new machine N (invoked as subroutine by M) in order to ensure that the gap produced by M is the easily computable quantity stated in the claim. We will also further structure the description N by designing another machine N' which is a subroutine to N. All these machines access G as oracle.

Description of $M(\langle 0^n, k, i, j_1, \ldots, j_k\rangle)$

```
Guess g ∈ S_n
if g ∈ μ_n(r) and g(j) = j; ∀j; 1 ≤ j ≤ i and g(i + l) = j_l; ∀l; 1 ≤ l ≤ k
   then Simulate Machine N on input ⟨0^n, i + k⟩
   else Branch into two paths
        accept on one path and reject on the other path
```

Now we argue that M has the claimed properties (assuming the behavior of N which we later establish). Suppose $\langle 0^n, k, i, j_1, \ldots, j_k, G\rangle \notin L(\mathrm{SYM})$. Then there exists no $g \in G^{(i)}$

such that $g(i+l) = j_l$ for $1 \leq l \leq k$. In this case the **if** test in *line 2* of M fails for all guesses and M produces zero *gap*. Now, suppose $\langle 0^n, k, i, j_1, \ldots, j_k, G\rangle \in L(\mathrm{SYM})$. It follows from part 3 of Lemma 5.3 that the number of guessed permutations g that fulfill the **if** test is $|G^{(i+k)}|$. Now, suppose N on input $\langle 0^n, i+k\rangle$ produces $gap = (n!)^{2n^2+1}/|G^{(i+k)}|$. Since N is being simulated on $|G^{(i+k)}|$ paths, the total gap produced by M is $((n!)^{2n^2+1}/|G^{(i+k)}|) \times |G^{(i+k)}| = (n!)^{2n^2+1}$.

Now, the task is to design N so that it has the desired behavior. Before we describe N we describe its subroutine N'.

Description of $N'(0^n, i, j)$

Guess $g \in S_n$
if $g \in G^{(i-1)}$ such that $g(i) = j$
  then if $g \in G$
    then accept
  else reject

Observe that N' has the following behavior: If there is $g \in G^{(i-1)}$ such that $g(i) = j$ then there are $|G^{(i)}|$ accepting paths, otherwise N has no accepting paths.

The machine N produces the required *gap* as follows. On input $\langle 0^n, i\rangle$, N tries to compute $|G^{(i)}|$ nondeterministically with G as oracle. For that it first guesses an encoding of a canonical generator set $X_i = \langle T_{i+1}, \ldots, T_n\rangle$ of $G^{(i)}$ and verifies that it actually encodes a canonical generator set of some subgroup of S_n. By making queries to G it verifies that the elements encoded in X_i are all in $G^{(i)}$. The bad case that still remains to be handled is that the guessed set X_i does not generate all of $G^{(i)}$ but only a *proper* subgroup of it. On computation paths where a generator set for a proper subgroup of $G^{(i)}$ is guessed, it must produce zero *gap*. We now give details.

Let $X_i = \langle T_{i+1}, \ldots, T_n\rangle$ where T_j, for $(i+1) \leq j \leq n$, encodes in lexicographic order a transversal of $G^{(i)}$ in $G^{(i-1)}$. (We use T_i to denote both a transversal of $G^{(i)}$ in $G^{(i-1)}$ and its lexicographic encoding.) Let $S_i = \{\langle j, k\rangle \mid (i+1) \leq j \leq k \leq n \text{ and } \nexists\ g \in X_i \cap G^{(j-1)}$ such that $g(j) = k\}$. (By '$g \in X_i$' we mean g is encoded in X_i.)

Description of $N(0^n, i)$

Guess an encoding $X_i = \langle T_{i+1}, \ldots, T_n\rangle$ of a canonical generator set for $G^{(i)}$
If X_i does not correctly encodes a canonical generator set
produce a $gap = 0$ and stop
For all $g \in X_i$ if $g \notin G^{(i)}$
produce a $gap = 0$ and stop
Using X_i compute $\alpha = \prod_{n \geq k > i} |T_k|$
Using X_i compute $\beta = \prod_{n \geq j > i} (\prod_{n \geq k > j} |T_k|)^{|T_j|}$
Branch into $(n!)^{n^2+1}/\alpha\beta$ and continue
Using X_i construct the set S_i
Compute $\gamma = \prod_{n \geq j > i} (\prod_{n \geq k > j} |T_k|)^{n-j-|T_j|+1}$
Branch into $(n!)^{n^2}/\gamma$ paths
Produce a gap of $\prod_{\langle j,k\rangle \in S_i} ((\prod_{n \geq l > j} |T_l|) - acc_{N'}\langle 0^n, j, k\rangle)$

Now, let us analyze the behavior of N. It is clear from the description of N that after *line* 5 only those computation paths matter which correctly encode a subgroup of $G^{(i)}$. All other paths contribute zero *gap* to the overall *gap* produced by N. The remaining steps of N ensure that those paths on which the guessed X_i generates a proper subgroup of $G^{(i)}$ contribute zero *gap*.

First, let ρ be a computation path on which the guessed X_i encodes a canonical generator set for $G^{(i)}$. We now show that N produces $gap = (n!)^{2n^2+1}/|G^{(i)}|$. By part 1 of Lemma 5.3, α computed in *line* 6 is $|G^{(i)}|$. Also, from part 2 of Lemma 5.3, β computed in *line* 7 is K_i, which is the number of canonical generators for $G^{(i)}$. So, in *line* 8 ρ contributes $gap = (n!)^{n^2+1}/(|G^{(i)}|K_i)$. (Notice that $(n!)^{n^2+1}$ is chosen to be divisible by the denominator.) In *line* 11, each such path again branch into $(n!)^{n^2}/\gamma$ paths. So at this point the path ρ has contributed $(n!)^{2n^2+1}/(|G^{(i)}|K_i\gamma)$ paths. In *line* 12, since X_i generates $G^{(i)}$ it follows that $acc_{N'}(\langle 0^n, j, k\rangle) = 0$, for each $\langle j, k\rangle \in S_i$. So, the *gap* produced in *line* 12 is $\prod_{\langle j,k\rangle \in S_i}(\prod_{n \geq l > j} |T_l|)$ which, by rearranging terms, is easily seen to be $\prod_{n \geq j > i}(\prod_{n \geq k > j} |T_k|)^{(n-j-|T_j|+1)} = \gamma$. Therefore, at end of *line* 12 ρ contributes a $gap = \gamma((n!)^{2n^2+1}/|G^{(i)}|K_i\gamma)$. Since there are K_i different paths ρ, one for each canonical generator set for $G^{(i)}$, the total contribution to the *gap* from the K_i generator sets adds up to $(n!)^{2n^2+1}/|G^{(i)}|$.

Now, let ρ be a computation path on which the guessed set X_i generates a proper subgroup of $G^{(i)}$. We show that this path contributes a $gap = 0$. Notice that in order to prove that ρ contributes $gap = 0$, it suffices to show that the *gap* produced in *line* 10 is zero. Observe that corresponding to X_i there is a least $j < i$, say j_0, such that T_{j_0} is not a transversal of $G^{(j_0)}$ in $G^{(j_0-1)}$. Therefore, there exists $k \leq n$ and $g \in G$ such that $\langle j_0, k\rangle \in S_i$, $g \in G^{(j_0-1)}$, and $g(j_0) = k$. On the other hand, since j_0 is the largest such index $\bigcup_{n \geq l > j_0} T_l$ is a canonical generating set for $G^{(j_0)}$. Putting the preceding two statements together, we can see that machine N' on input $\langle 0^n, j_0, k\rangle$ has $|G^{(j_0)}|$ accepting paths. Furthermore, since $\prod_{n \geq l > j_0} |T_l| = |G^{(j_0)}| = acc_{N'}(\langle 0^n, j_0, k\rangle)$, it holds that the product $\prod_{\langle j,k\rangle \in S_i}(acc_{N'}(\langle 0^n, j, k\rangle) - (\prod_{n \geq l > j} |T_l|)) = 0$. Hence the gap produced on path ρ is zero. Finally, notice that the *gap* prescribed in *line* 12 is indeed gap-definable from the closure properties of GapP given in Section 2. This proves Claim 5.4.1. ∎

Claim 5.4.2 *There is a polynomial time deterministic learning algorithm* Learn SYM *with* L(SYM) *as assistant which, given a target concept* G *from the class* SYM *outputs a representation (a canonical generator set) for* G.

Proof The objective of the learner Learn SYM is to compute a canonical generator set as a union $\bigcup_{i=1}^{n} T_i$, where T_i is a transversal of $G^{(i)}$ in $G^{(i-1)}$, for $1 \leq i \leq n$. To this end, the learner first computes, with the help of L(SYM), the set of pairs $\{\langle i, j\rangle \mid \exists\ g \in G$ such that $g \in G^{(i-1)}$ and $g(i) = j\}$. For each such pair $\langle i, j\rangle$ it invokes a subroutine Construct on input $\langle 0^n, i, j\rangle$ in order to compute the lexicographically first $g \in G^{(i-1)}$ such that $g(i) = j$. This is done by a prefix search with the help of L(SYM). It is easy to see that the set of all these computed elements g is a canonical generator set. We give a formal description of this simple learning algorithm below.

Learn SYM(0^n)
for pairs $\langle i, j\rangle; 1 \le i \le j \le n$ **do**
if $\langle 0^n, 1, i-1, j, G\rangle \in L(\text{SYM})$
  then Construct($0^n, i, j$)

Construct($0^n, i, j$)
$k \leftarrow 1$
repeat
    Find first $l_k; i < l_k \le n$ such that
    $\langle 0^n, k+1, i-1, j, x_1, \ldots, x_{k-1}, l_k, G\rangle \in$L(SYM)
    $x_k \leftarrow l_k$
    $k \leftarrow k+1$
  until $k = n - i$
return $(1, \ldots, i-1, j, x_1, \ldots, x_{n-i})$

It is easy to see that Learn SYM has time complexity $O(n^4)$. ■

6 Learning linear spaces via teaching assistants

In this section we give an algorithm for exactly learning linear spaces over the finite field F_p. We recall first some basic definitions and results. Details can be found in any linear algebra textbook, e.g. [13].

Let U be a vector space over a field F. We denote the zero vector of U by $\mathbf{0}$. U is *nontrivial* if $U \neq \{\mathbf{0}\}$. Let $V < U$ denote that V is a subspace of U. For $X \subseteq U$ let $\langle X\rangle$ denote the subspace of U spanned by X. The direct sum of U and V is denoted by $U \oplus V$, and the n-fold direct sum of F is denoted by F^n.

We define now the representation class for subspaces of F_p^n. A vector $u \in F_p^n$ is represented as an n-tuple $\langle u[1], \ldots, u[n]\rangle$. We use a standard pairing function to encode this tuple whose length is bounded by $q(n)$ for some polynomial q. We denote by $\text{LS}(p)$ the representation class of linear spaces over F_p: $\text{LS}(p) = \langle R, \{\mu_n\}_{n\ge1}, p\rangle$. Here $R = \cup_{n\ge1} R_n$, with each $r \in R_n$ encoding a set of vectors in F_p^n. For $r \in R_n$, $\mu_n(r)$ is the subspace spanned by the vectors encoded in r. Clearly $\text{LS}(p)$ is an honest representation class.

Before describing the learning algorithm we recall some more linear algebra.

Proposition 6.1 *Let $W < V < U$ be finite dimensional vector spaces over a field F. Let A be a basis for U and B a basis for W. Let A' be a maximal subset of A such that $A' \cup B$ is linearly independent. Then $A' \cup B$ is a basis for U. Moreover, $V = W \oplus (\langle A'\rangle \cap V)$.*

Let F be a field, and U be a nontrivial subspace of F^n. Let $U^{(i)}$ denote the subspace $\{u \in U \mid \text{for } 1 \le j \le i: \; u[j] = o\}$ of U, for $1 \le i \le n$, and let $U^{(0)}$ denote U. The *0-index* of U is the maximum k such that $U^{(k)}$ is nontrivial. We have the following easy proposition.

Proposition 6.2 *Let F be a field and U be a nontrivial subspace of F^n. If k is the 0-index of U then $U^{(k)}$ is a 1-dimensional subspace.*

We now present the learning algorithm of this section.

Theorem 6.3 *For any prime p, the representation class* $\mathrm{LS}(p)$ *is* FP*-learnable with an SPP-assistant.*

Proof. We first define an assistant $L(\mathrm{LS}(p))$ and show that $\mathrm{LS}(p)$ is FP-learnable with assistant $L(\mathrm{LS}(p))$. Then we prove that $L(\mathrm{LS}(p)) \in \mathrm{SPP}(\mathrm{LS}(p))$.

On input 0^n, let $V < F_p^n$ be the target subspace $\mu_n(r)$. Let vectors F_p^n be uniformly encoded in strings of length $q(n)$ for some polynomial q. Define $L(\mathrm{LS}(p))$ as: $L(\mathrm{LS}(p)) \triangleq \{\langle 0^n, A, i, y, V\rangle | i$ is the 0-index of $(\langle A\rangle \cap V); \exists z \in \Sigma^{q(n)}$ with $yz \in \langle A\rangle \cap V; yz \neq \mathbf{0}\}$.

We first give an intuitive idea of how the learner works. The aim of the learner is to construct a basis for the target subspace V. Suppose we have constructed an independent set of vectors $B \subseteq V$. We use $L(\mathrm{LS}(p))$ to prefix search for a vector $v \in V - \langle B\rangle$ as follows: Let A be a basis for F_p^n. Construct $A' \subseteq A$ such that $A' \cup B$ is a basis for F_p^n. (This can be done with some rank computations, which can be carried out in polynomial time.) Next, compute the 0-index of the space $(\langle A'\rangle \cap V)$ with $L(\mathrm{LS}(p))$ as oracle, by making queries of the form $\langle 0^n, A', i, \epsilon, V\rangle$ for $1 \leq i \leq n$. We now construct v by doing a prefix search with queries to $L(\mathrm{LS}(p))$. Notice that if $\langle B\rangle \neq V$ then by Proposition 6.1 $\langle A'\rangle \cap V$ is nontrivial and every nonzero element in it is independent of vectors in $\langle B\rangle$. Therefore, including v in B increases the dimension of $\langle B\rangle$. This process of including new vectors in $\langle B\rangle$ will therefore terminate within n steps. We now formally describe the learning algorithm.

LEARN-LINEAR(0^n)
$B \leftarrow \{\mathbf{0}\}$
$A \leftarrow \{e_1, \ldots, e_n\}$ (* Standard basis for F_p^n *)
Construct $A' \subseteq A$ such that $B \cup A'$ is maximally independent
$i \leftarrow 1$
while $\langle 0^n, A', i, \lambda, V\rangle \notin L(\mathrm{LS}(p))$
  do $i \leftarrow i + 1$
if $i = n + 1$
  then **output** B and stop
$y \leftarrow \lambda$
for $j \leftarrow 1$ **to** $q(n)$
  do if $\langle 0^n, A', i, y1, r\rangle \in L(\mathrm{LS}(p))$
    then $y \leftarrow y1$
    else $y \leftarrow y0$
$B \leftarrow B \cup \{y\}$
goto 3

We now show that $L(\mathrm{LS}(p)) \in \mathrm{SPP}(\mathrm{LS}(p))$. We construct a GapP machine M which on input $\langle 0^n, A', i, y\rangle$, uses V as oracle, and has the property that $\langle 0^n, A', i, y, V\rangle \in L(\mathrm{LS}(p))$ if $M(\langle 0^n, A', i, y\rangle)$ produces $gap = 1$, and if the input is not in $L(\mathrm{LS}(p))$, M produces $gap = 0$.

To this end we define another NP machine N (which uses $\mu_n(r)$ as oracle). On input $\langle 0^n, A', i, y, k\rangle$ the machine N guesses $p - 1$ distinct nonzero vectors $v_1, \ldots, v_{p-1} \in \langle A'\rangle^{(k)}$ in lexicographically increasing order, such that y is a prefix of v_1 and N accepts along this path iff each $v_i \in V$. It follows from the definition of 0-index that if k is more than the 0-index of $A' \cap V$ then N has no accepting paths. Furthermore, if k is the 0-index then, from Proposition 6.2, N has a unique accepting path.

Let $acc_N(\langle 0^n, A', i, y, k\rangle)$ be the number of accepting paths of N on input $\langle 0^n, A', i, y, k\rangle$. We design now the desired GapP machine M that invokes N as subroutine.

Description of $M(\langle 0^n, A, i, y\rangle)$

1 Produce a *gap* of $gap_N(\langle 0^n, A', i, y, i\rangle) \cdot \prod_{k=i+1}^{n}(1 - acc_N(\langle 0^n, A', i, y, k\rangle))$.

Notice that on input $\langle 0^n, A', i, y\rangle$, if i is not the 0-index then M has a $gap = 0$. Furthermore, if i is the 0-index then M has $gap = 1$ if y is the prefix of the guessed vector v_1, and otherwise has $gap = 0$. Hence $L(\mathrm{LS}(p))$ is in $\mathrm{SPP}(\mathrm{LS}(p))$. Again, notice that M can produce the *gap* prescribed in *line* 1 above since the desired value is easily seen to be gap-definable from the closure properties of GapP given in Section 2. ∎

7 Varying the power of teaching assistants

In the previous section, we had introduced P-assistants, NP-assistants, and Σ_2^p-assistants. As mentioned earlier, the different assistant classes that we have defined is in analogy to standard complexity classes. Furthermore, the previous section indicates that some of these teaching assistant classes have some correspondence with exact learnability in Angluin's model. Motivated by this connection, and again in analogy with complexity classes, we study teaching assistant classes, particularly, NP $\cap$ co-NP-assistants and UP $\cap$ co-UP-assistants and some other assistant classes. In the next two sections we show that these assistant classes play a role in a finer classification of exact learnability of some algebraic concept classes. Another reason for interest in NP $\cap$ co-NP-assistants is our result that learnability NP $\cap$ co-NP-assistants is well characterized by a combinatorial property which is an interesting variant of teaching dimension [10].

To begin with, the next two theorems give machine characterizations of concept classes that are FP-learnable with an NP $\cap$ co-NP-assistant and FP-learnable with an UP $\cap$ co-UP-assistant. In next sections we use this characterization to prove that some classes are not FP-learnable with an NP $\cap$ co-NP-assistant.

The proof of the next theorem is similar to the proof of a result implicitly in [5], which says that a function $f \in \mathrm{FP}^{\mathrm{NP}\cap\text{co-NP}}$ iff f is NPSV-computable.

Theorem 7.1 *For any representation class $\mathcal{P}$, and any language $A \subseteq \Sigma^*$*

1. *$\mathcal{P}$ is NPSV-learnable with membership queries with the aid of a computational oracle A if and only if $\mathcal{P}$ is* FP*-learnable with an* $(\mathrm{NP} \cap \text{co-NP})^A$*-assistant.*
2. *$\mathcal{P}$ is* UPSV*-learnable with membership queries with the aid of a computational oracle A if and only if $\mathcal{P}$ is* FP*-learnable with an* $(\mathrm{UP} \cap \text{co-UP})^A$*-assistant.*

Proof. We prove part 1 of the above theorem. The second part can be proved similarly.

Let $\mathcal{P} = \langle R, \{\mu_n\}_{n\geq 1}, p\rangle$ be any representation class and A be a computational oracle. Let β be an FP-learner for $\mathcal{P}$ with teaching assistant $L(\mathcal{P})$ in $(\mathrm{NP}(\mathcal{P})\cap\text{co-NP}(\mathcal{P}))^A$. Let M^A be a nondeterministic oracle Turing machine witnessing $L(\mathcal{P})$ in $(\mathrm{NP}(\mathcal{P})\cap\text{co-NP}(\mathcal{P}))^A$. Let $c \in \mathcal{P}_n$ be the target concept and l be such that $\mathit{size}(c) \leq l$. Consider the following learner β' which on input $\langle 0^n, 0^l\rangle$ simulates the learner β on input $\langle 0^n, 0^l\rangle$. Whenever β makes a query $\langle n, l, x, c\rangle$ to $L\mathcal{P}$, β' start simulating M on input $\langle 0^n, 0^l, x\rangle$ by guessing a path ρ of M. During the simulation of path ρ of M by β, whenever M makes a query y to A, β' also makes query y to A. Whenever M makes a query z to the target concept c, β' makes a membership query z to the teacher and treats the answer as the answer to M's query z. If the path ρ outputs '?', then β' abandons that path. If the path ρ outputs 'accept' then β' treats this as answer 'Yes' to the query $\langle n, l, x, c\rangle$ of β and proceeds. If the path ρ outputs 'reject' then β' treats this as answer 'No' to the query $\langle n, l, x, c\rangle$ of β and proceeds. Now, on any single simulation of M by β', the paths which are not abandoned by β' has the same answer, it clear that, in the end, β' running in time bounded by a polynomial in $n + l$, outputs the same string as output by β.

To show the containment in the other direction, let $\mathcal{P} = \langle R, \{\mu_n\}_{n\geq 1}, p\rangle$ be any representation class that is NPSV-learnable with membership queries. Let α be an NPSV-learner for $\mathcal{P}$ which uses A as computational oracle. Let q be the polynomial bounding the running time of α. Without loss of generality, we can assume that for input $\langle 0^n, 0^l\rangle$, all the paths of α runs for $q(n+l)$ time steps. Consider the following teaching assistant $L(\mathcal{P}) = \{\langle n, l, (i, b), c\rangle | \alpha$ on input $\langle 0^n, 0^l\rangle$, with c as target concept, and A as computational oracle, runs for $q(n+l)$ steps and outputs a string y whose i^{th} bit is $b\}$. Now we show that $L(\mathcal{P}) \in (\mathrm{NP}(\mathcal{P})\cap\text{co-NP}(\mathcal{P}))^A$. Consider the following oracle Turing machine M^A which on input $\langle 0^n, 0^l, (i, b)\rangle$ simulates the learner α for $q(n+l)$ steps with c as oracle. On any path ρ of α, if α makes a query y to A, M^A also makes query y to A. Whenever α makes a membership query z to the teacher, M^A makes a membership query z to the target concept c and treats the answer as the answer to α's query z. At the end of $q(n + l)$ steps of computation, on the paths where α does not output anything, M^A outputs '?' and on the paths where α outputs a string y M^A outputs 'accept' if i^{th} bit of y is b and outputs 'reject' if i^{th} bit of y is not b or $|y| < i$. It is easy to verify that M^A witnesses $L(\mathcal{P}) \in (\mathrm{NP}(\mathcal{P})\cap\text{co-NP}(\mathcal{P}))^A$. Now we give an FP-learner α' for $\mathcal{P}$ with teaching assistant $L(\mathcal{P})$. Let $c \in \mathcal{P}_n$ be the target concept and l such that $\mathit{size}(c) \leq l$.

NP $\cap$ co-NP-LEARNER $\alpha'(0^n, 0^l)$

$i \leftarrow 1$
while $\langle n, l, (i, b), c\rangle$ or $\langle n, l, (i, \overline{b}), c\rangle \in L(\mathcal{P})$
  do if $\langle n, l, (i, b), c\rangle \in L(\mathcal{P})$
    then $y \leftarrow yb$
    else $y \leftarrow y\overline{b}$
   $i \leftarrow i + 1$
Output y

It is clear that α' constructs the representation of c which is output by α. Also, the running time of α' is bounded by $O(q(n + l))$. ■

What is the difference between NP $\cap$ co-NP-assistants and UP $\cap$ co-UP-assistants? We show in the next theorem that in an information-theoretic sense they are equivalent. More precisely, we show that UP $\cap$ co-UP-assistants with NP as a computational oracle are as powerful as NP$\cap$co-NP-assistants. Thus, separating these assistant classes in the polynomial-time learnability sense is as hard as separating P and NP.

Theorem 7.2 *Let $\mathcal{P}$ be any representation class and $A \subseteq \Sigma^*$ be a computational oracle. Then if $\mathcal{P}$ is* FP*-learnable with an* $(\mathrm{NP} \cap \text{co-NP})^A$*-assistant, then there is language $B \in \mathrm{NP}^A$ such that $\mathcal{P}$ is* FP*-learnable with a* $(\mathrm{UP} \cap \text{co-UP})^B$*-assistant.*

Proof. We use the machine characterization of FP-learnability with NP $\cap$ co-NP-assistants and UP$\cap$co-UP-assistants proved in Theorem 7.1. Let $\mathcal{P} = \langle R, \{\mu_n\}_{n\geq 1}, p\rangle$ be any representation class which is FP-learnable with NP $\cap$ co-NP^A-assistant. Then by Theorem 7.1, there is an NPSV-learner α, which learns $\mathcal{P}$ with only membership queries, with the aid of A as oracle. Consider the language (not to be confused with a teaching assistant) $B' = \{\langle 0^n, 0^l, x, y\rangle | \exists$ a path ρ of α , which is lexicographically lesser than x, and α on input $\langle 0^n, 0^l\rangle$, $\mu_n(y)$ as target concept and A as oracle, outputs a string on $\rho\}$ and the language $B = A \oplus B'$. We first show that B' is in NP^A. Consider a nondeterministic oracle machine N^A which on input $\langle 0^n, 0^l, x, y\rangle$ first guess a path ρ, which is lexicographically smaller than x, of α on input $\langle 0^n, 0^l\rangle$ and $\mu_n(y)$ as target concept and simulates α on that path. Whenever α on ρ makes a query z to A, N^A also makes query z to A. Since the representation class is honest, checking membership in $\mu_n(y)$ can be done in polynomial-time. So whenever α makes membership queries z to the target concept $\mu_n(y)$, N^A computes the answer to this query with the representation y (part of input) of $\mu_n(y)$. Hence $B' \in \mathrm{NP}^A$ and, consequently, $B \in \mathrm{NP}^A$.

To complete the proof, consider a UPSV-learner α', which on input $\langle 0^n, 0^l\rangle$ and target concept c, simulates the NPSV-learner α on input $\langle 0^n, 0^l\rangle$ and target concept c. If α on a computation path x outputs a string y as a representation of the target concept, α' makes a query $\langle 0^n, 0^l, x, y\rangle$ to B. If the answer is 'No', α' outputs y. If the answer to the query is 'Yes', α' halts without output. It is clear that α' outputs only on the unique lexicographically first path on which α outputs. Hence α' is a UPSV-learner with membership queries and B as oracle for $\mathcal{P}$. From Theorem 7.1, it follows that $\mathcal{P}$ is FP-learnable with a $(\mathrm{UP}\cap\text{co-UP})^B$-assistant with the help of an NP oracle. ∎

The next corollary is immediate.

Corollary 7.3 *If there exists a representation class $\mathcal{P}$ that is* FP*-learnable with an* NP $\cap$ co-NP*-assistant, and not* FP*-learnable with a* UP $\cap$ co-UP*-assistant, then* P $\neq$ NP*.*

It is known that for any constant k, the representation class k-CNF is polynomial time learnable with equivalence queries [1]. We next show that the representation class 3-CNF is unlikely to be FP-learnable with assistants in any UP(3-CNF), LWPP(3-CNF) or SPP(3-CNF)

Theorem 7.4

1. *The representation class* 3-CNF *is not* FP*-learnable with a* NP $\cap$ co-NP*-assistant.*

2. *If* 3-CNF *is* FP-*learnable with an* SPP-*assistant, then* NP $\subseteq$ SPP. *If* 3-CNF *is* FP-*learnable with an* LWPP-*assistant, then* NP $\subseteq$ LWPP. *If* 3-CNF *is* FP-*learnable with a* UP-*assistant, then* NP $\subseteq \mathrm{P}^{\mathrm{UP}}$.

Proof. For proving the first part, let $\mathcal{P}'$ be the subclass of 3-CNF defined as follows:

$$P'_n \triangleq \{\varphi \mid \varphi \text{ is a conjunction of } n \text{ literals }\}$$

For each n, the concept class is the set of subsets of $\{0,1\}^n$ of size at most 1. (Notice that there are 2^n+1 such concepts.) In particular, for each n, the empty set is also a concept in $\mathcal{P}'$. Let us denote it by $\emptyset$. Now, let α be an NPSV-learner for $\mathcal{P}'$ whose running time is bounded by a polynomial q. Consider the computation of α on input $\langle 0^n, 0^n\rangle$ with the concept $\emptyset$ as the target concept. Consider a path ρ which outputs a representation of $\emptyset$. Let $x_1, \ldots, x_k$, $k \le q(2n)$, be the membership queries asked by α on ρ (all have to be answered 'No' by the teacher). Let $x \in \{0,1\}^n$ be different from $x_1, \ldots, x_k$. Such an x exists for sufficiently large n. Now consider α learning the concept $\{x\}$ on input $\langle 0^n, 0^n\rangle$. Then, for the target concept $\{x\}$, learner α has a computation path identical to ρ on which it must output a representation for $\emptyset$ which contradicts the definition of NPSV-learners.

For the second part, we prove that if 3-CNF is FP-learnable with an SPP-assistant, then NP $\subseteq$ SPP. The other parts can be similarly proved. For proving this, we show that if 3-CNF has a learning algorithm with an SPP-assistant, then the co-NP-complete language TAUT= $\{\langle 0^n, 0^l, f\rangle | f$ encodes 3-CNF on n variables of size $\le l$ and f is a tautology$\}$ is in $\mathrm{P}^{\mathrm{SPP}}$ which is SPP [6] because SPP is closed under complement.

Let α be a learner, which on input $\langle 0^n, 0^l\rangle$ learns an n-variable 3-CNF of size less than or equal to l with an SPP-assistant $L($ 3-CNF$) = \{\langle n, l, y, c\rangle | y \in \Sigma^*; Q(n, l, y, c)\}$ accepted by a machine N, where Q is the predicate defining the SPP-assistant. Define a language $A = \{\langle 0^n, 0^l, y, f\rangle | y \in \Sigma^*; f$ encodes a 3-CNF formula and $Q'(n, l, y, f)\}$ where the predicate Q' is such that if $Q'(n, l, y, f)$ is true then N on input $\langle 0^n, 0^l, y\rangle$ with f as target formula has $gap = 1$ and if $Q'(n, l, y, f)$ is false then N will have a $gap = 0$. It is easy to see that $A \in$ SPP. Consider the deterministic machine M, for accepting TAUT, which on input $\langle 0^n, 0^l, f\rangle$ does the following. M simulates α on input n two times, sequentially. In the first simulation whenever α makes a query $\langle n, l, y, c\rangle$ to $L($ 3-CNF), M makes a query $\langle 0^n, 0^l, y, f\rangle$ to A. In the next simulation whenever α makes a query $\langle n, l, y, c\rangle$ to $L($ 3-CNF), M makes a query $\langle 0^n, 0^l, y, T\rangle$ to A (T is the trivial formula *true*). M accepts $\langle 0^n, 0^l, f\rangle$ if the output of M in both the cases are the same. It is easy to see that since α is a deterministic machine it outputs the same string in both simulations iff f is a tautology. ∎

7.1 Teaching dimension and teaching assistants

The notion of *teaching dimension* for a concept class was introduced in [10]. It is shown in [10] that the number of membership queries needed to learn a representation class is at least its teaching dimension (more precisely, the number of membership queries for exact FP-learnability with only membership queries in Angluin's model). Furthermore, in [12] it is shown that polynomial teaching dimension is also sufficient to learn *projection-closed* concept classes with polynomially many membership queries. We define now a variant of the teaching dimension called *weak teaching dimension*, and using this notion we characterize FP-learnability with (NP $\cap$ co-NP)-assistants. Specifically, we show that any representation class which does not have *polynomial weak teaching dimension* is not FP-learnable with an (NP $\cap$ co-NP)^A-assistant for any computational oracle A. Conversely, we also show that if a representation class has polynomial weak teaching dimension, then it is FP-learnable with a (UP $\cap$ co-UP)^A-assistant with computational oracle $A \in \Sigma_3^p$. Here, we note that from an information theoretic standpoint a (UP $\cap$ co-UP)-assistant is equivalent to an (NP $\cap$ co-NP)-assistant by Theorem 7.2. We now define the notion of weak teaching dimension.

Let $\mathcal{P} = \langle R, \mu, p\rangle$ be a representation class where $\mu = \{\mu_n\}_{n\in\mathbf{N}}$. For any n, an *example* is a pair (x, b) where $x \in \Sigma^{p(n)}$ and $b \in \{0, 1\}$. A set X of examples is said to be consistent with a concept $c \in \mathcal{P}_n$ if for all $(x, b) \in X$ it holds that $b = 1 \Leftrightarrow x \in c$. For $c \in \mathcal{P}_n$ and $\mathcal{C} \subseteq \mathcal{P}_n$ we say that a set of examples X *specifies* c *over* $\mathcal{C}$ if X is consistent with c and X is not consistent with any concept $c' \in \mathcal{C} - \{c\}$. For any n, let $k_n \stackrel{\triangle}{=} max\{size(c) \mid c \in \mathcal{P}_n\}$. Notice that $\mathcal{P}_n = \bigcup_{i=1}^{k_n} \mathcal{P}_{n,i}$. The representation class $\mathcal{P}$ is said to have *polynomial weak teaching dimension* if there exists a polynomial q such that for all n the following holds:
For $c \in \mathcal{P}_n$ and for all $j; 1 \leq j \leq k_n$ there exists sets X_j of examples, such that $|X_j| \leq q(n + j + size(c))$ and X_j specifies c over $\mathcal{P}_{n,j}$.

Remark. The above definition can be seen as a variant of polynomial teaching dimension as defined in [12]. We briefly recall it. A representation class $\mathcal{P}$ is said to have polynomial teaching dimension if there exists a polynomial q such that for all n and $c \in \mathcal{P}_n$ there exists a set X of examples such that $|X| \leq q(n + size(c))$ and X specify c over $\mathcal{P}_n$. It follows from the definitions that if $\mathcal{P}$ has polynomial teaching dimension then it also has polynomial weak teaching dimension. In other words, in polynomial weak teaching dimension, for a given concept c, we have a *different* distinguishing example set X_j for specifying c over concepts of size j. It is easy to see that for representation classes $\mathcal{P}$ with polynomial-size (in n) representations for concepts in $\mathcal{P}_n$, (i.e., subclasses of the nonuniform complexity class P/poly) polynomial weak teaching dimension implies polynomial teaching dimension.

Theorem 7.5 *Let $\mathcal{P}$ be any representation class. If $\mathcal{P}$ is* FP*-learnable with an* NP $\cap$ co-NP*-assistant then $\mathcal{P}$ has polynomial weak teaching dimension.*

Proof.

Let $\mathcal{P}$ be representation class that is FP-learnable with an NP$\cap$co-NP-assistant. Suppose $\mathcal{P}$ does not have polynomial weak teaching dimension.

We use the characterization of FP-learnability with NP $\cap$ co-NP-assistant proved in Theorem 7.1. Let α be an NPSV-learner, which learns $\mathcal{P}$ with membership queries. Let the polynomial q bound the running time of α. If $\mathcal{P}$ does not have polynomial weak teaching dimension, then, in particular for q, there is an n and a concept $c \in \mathcal{P}_n$ such that for some $j : 1 \leq j \leq k_n$ it holds that all example sets X with $|X| \leq q(n + j + size(c))$ fail to specify c over $\mathcal{P}_{n,j}$. Let $l = j + size(c)$. Consider the computation of α on input $\langle 0^n, 0^l\rangle$ with c as target concept. Let ρ be a path on which α outputs a representation of c. Let $X = \{(x, b) \mid x \in \Sigma^{p(n)}$ is a string queried by α on ρ and b is the answer to query $x\}$. Clearly $|X| \leq q(n + l)$. Let $c' \in \mathcal{P}_{n,j} - \{c\}$ be a concept that is consistent with X. Such a c' exists since X fails to specify c over $\mathcal{P}_{n,j}$. Now consider the computation of α with input $\langle 0^n, 0^l\rangle$ and c' as target concept. Since c' is consistent with X, there is a computation of α which is the same as the computation path ρ when the target concept was c. Thus, α outputs a representation of c rather than c' on ρ, contradicting the definition of an NPSV-learner. ■

Theorem 7.6 *Let $\mathcal{P}$ be any representation class. If $\mathcal{P}$ has polynomial weak teaching dimension, then it is* FP*-learnable with a* $\mathrm{UP} \cap \text{co-UP}$*-assistant, with a computational oracle* $A \in \Sigma_3^p$.

Proof. The idea of the proof is as follows. We design an UPSV-learner α for $\mathcal{P}$ which on input $\langle 0^n, 0^l\rangle$ with c as target concept such that $size(c) \leq l$ does the following. α guesses a valid representation $\mathcal{P}$ of length less than or equal to l. On a guessed representation r, with a computational oracle $A_1 \in \Sigma_2^p$ as oracle it checks whether r is the lexicographically smallest representation of length $\leq l$ that represents the concept $\mu_n(r)$. If not, α stops computation. For a correctly guessed representation r, with a language $A_2 \in \Sigma_3^p$ as oracle, computes a set X_r of examples which specifies $\mu_n(r)$ over $\mathcal{P}_{n,l}$. With membership queries, it then verifies that X_r is consistent with the target concept c. If this is indeed the case, α outputs r as a representation of the target concept. We give the formal description of the learner α and the languages which α uses as oracles, below.

Define $A_1 \stackrel{\triangle}{=} \{\langle 0^n, 0^l, r\rangle | r \in \Sigma^{\leq l}; \exists r' \in \Sigma^{\leq l}; r'$ is lexicographically smaller that r and $\mu_n(r) = \mu_n(r')\}$. It is easy to see that $A_1 \in \Sigma_2^p$. To define language A_2, we first define a language A_2' and define A_2 as an appropriate prefix language of A_2'. For a polynomial q, $A_2' \stackrel{\triangle}{=} \{\langle 0^n, 0^l, r, X\rangle \mid r \in \Sigma^{\leq l}$; X is a set of examples of $\mathcal{P}_n$ such that $|X| \leq q(n+l)$; X specifies $\mu_n(r)$ over $\mathcal{P}_{n,l}\}$. Now define $A_2 \stackrel{\triangle}{=} \{\langle 0^n, 0^l, r, y, b, Y\rangle \mid b \in \{0,1\}$, Y a set of examples of $\mathcal{P}_n$, and $\exists$ a set X of examples of $\mathcal{P}_n$ and $z \in \Sigma^{p(n)-|y|}$ such that $\langle 0^n, 0^l, r, Y \cup X \cup \{(yz, b)\}\rangle \in A_2'\}$.

Notice that with A_2 as computational oracle, a polynomial-time deterministic machine can compute, for any n and l, a set of examples X (if it exists) for any concept $c \in \mathcal{P}_n$ which specifies c over $\mathcal{P}_{n,m}$.

Now we see that $A_2 \in \Sigma_3^p$. For this it is enough to show that $A_2' \in \Pi_2^p$. We give a co-NP oracle machine N with B as oracle that accepts A_2', where $B = \{\langle 0^n, r_1, r_2\rangle | \mu_n(r_1) \neq \mu_n(r_2)\}$. Notice that $B \in$ NP. N^B on input $\langle 0^n, 0^l, r, X\rangle$ universally guesses a representation $r' \in \Sigma^{\leq l}$ and rejects *only if* $\mu_n(r) \neq \mu_n(r')$ and X is consistent with both $\mu_n(r)$ and $\mu_n(r')$. Machine N checks $\mu_n(r) = \mu_n(r')$ with a single query to B. Clearly, N^B accepts A_2'. Therfore, $A_2' \in \Pi_2^p$ and $A_2 \in \Sigma_3^p$.

Let c be the target concept and $\langle 0^n, 0^l\rangle$ the input such that $size(c) \leq l$.

TD-Learner $\alpha(0^n, 0^l)$
guess $r \in \Sigma^{\leq l}$
If r is not a valid representation, **stop** without output
If $\langle 0^n, 0^l, r\rangle \notin A_1$ **stop** without output
Querying A_2, compute a set X of examples which specifies $\mu_n(r)$ over $\mathcal{P}_{n,m}$
for $(x, b) \in X$
  do If $x \in c \Leftrightarrow b = 0$ then **stop** without output
end-for

8 **output** r

In *line* 3, α stops all those guessed paths where a representations r, such that there is a lexicographically smaller representation r' which represents the same concept $\mu_n(r)$, is guessed. So after *line* 3, only a *unique* (lexicographically smallest) representation for all the concepts in $\mathcal{P}_{n,l}$ survives. Since $\mathcal{P}$ has polynomial weak teaching dimension, there exists a polynomial q such that for any $c \in \mathcal{P}_{n,l}$ there is a set X; $|X| \leq q(n+l)$ which specifies c over $\mathcal{P}_{n,l}$. So with A_2 as oracle, for any r representing a concept in $\mathcal{P}_{n,l}$, such a set X_r can be correctly computed deterministically in time polynomial in $n+l$. *line* 4 of α accomplishes this. In *lines* 5-7, α verifies if the guessed string r represents the target concept c by checking that X_r is consistent with c, with membership queries. It is clear that if *size*$(c) \leq l$ then α outputs the lexicographically smallest representation of the target concept on a unique path. ■

We end this section with a note on the complexity of minimum representation problem. Formally, let $\mathcal{P}$ be a representation class. For $\mathcal{P}$, the minimum representation problem MIN-REP$(\mathcal{P}) = \{\langle 0^n, r\rangle | r$ represents a concept in $\mathcal{P}_n$ and there is no r' such that r' is lexicographically smaller than r and $\mu_n(r) = \mu_n(r')\}$. It is easy to see that for any representation class $\mathcal{P}$, MIN-REP$(\mathcal{P}) \in \Pi_2^p$. We observe that for any representation class $\mathcal{P}$ which has polynomial weak teaching dimension, MIN-REP$(\mathcal{P}) \in \Sigma_2^p$ and hence cannot be complete for Σ_2^p or Π_2^p unless PH collapses to Σ_2^p. We state this as a proposition.

Proposition 7.7 *Let $\mathcal{P}$ be any representation class. Then if $\mathcal{P}$ has polynomial weak teaching dimension, then* MIN-REP$(\mathcal{P}) \in \Sigma_2^p \cap \Pi_2^p$.

Proof. For any representation class $\mathcal{P}$, MIN-REP$(\mathcal{P}) \in \Pi_2^p$. Suppose that $\mathcal{P}$ has polynomial weak teaching dimension. Let q be the polynomial witnessing this. Consider the following Σ_2^p machine M which on input $\langle 0^n, r\rangle$, existentially guesses a set X of examples such that $|X| \leq q(n+2|r|)$ and verifies that X is consistent with the concept represented by r (this example set X specifies $\mu_n(r)$ over $\mathcal{P}_{n,|r|}$). On all such computation paths, M universally guesses a valid representation r', lexicographically smaller than r and accepts if X is not consistent with r'. It is easy to see that M accepts MIN-REP$(\mathcal{P})$. ■

8 Separating the power of teaching assistants

We first define some subclasses of SYM. We define CYCLIC-SYM as $\langle R', \{\mu'_n\}_{n\geq 1}, p'\rangle$, where R' consists of generator sets that generate *cyclic* groups. The subclass n-CYCLES is defined as $\langle R'', \{\mu'_n\}_{n\geq 1}, p'\rangle$, where R'' consists of generator sets, each of which is just a single *n-cycle*. It is clear that n-CYCLES$\subseteq$CYCLIC-SYM and CYCLIC-SYM$\subseteq$ SYM. Using standard techniques in algorithmic permutation group theory [7] it can be checked whether a set of permutations generates a cyclic group. Hence CYCLIC-SYM is an honest representation class, and n-CYCLES is obviously an honest representation class.

For proving nonlearnability results we need the following propositions.

Proposition 8.1 *Let p be a prime. If g_1 and g_2 are two p-cycles in S_p then either $\langle g_1 \rangle \cap \langle g_2 \rangle = \langle e \rangle$ or $\langle g_1 \rangle = \langle g_2 \rangle$.*

Proof. Since order of any p-cycle is p, $|\langle g_1 \rangle| = p$. Suppose $\langle g_1 \rangle \neq \langle g_2 \rangle$. Then $\langle g_1 \rangle \cap \langle g_2 \rangle$ is a proper subgroup of $\langle g_1 \rangle$. Since p is prime, from Lagrange theorem, it follows that $|\langle g_1 \rangle \cap \langle g_2 \rangle| = 1$. So $\langle g_1 \rangle \cap \langle g_2 \rangle = \langle e \rangle$. ■

Proposition 8.2 *Let p be a prime. Then there are $(p-2)!$ cyclic subgroups of S_p generated by p-cycles.*

Proof. The number of p-cycles in S_p is $(p-1)!$. Let g be any arbitrary p-cycle. Then all the elements except identity in $\langle g \rangle$ are p-cycles and all of them generate the group $\langle g \rangle$. Now from Proposition 8.1, it follows that if $g_1 \in \langle g \rangle$ is an element other than identity, then g_1 cannot be member of any other group generated by a p-cycle. Since there are $(p-1)$ p-cycles in any group generated by a p-cycle, there are $(p-1)!/(p-1) = (p-2)!$ cyclic subgroups in S_p generated by p-cycles. ■

Now we prove that the subclass n-CYCLES of SYM is not FP-learnable with membership and equivalence queries. This is an example of the situation where concept class $\mathcal{P}$ is equivalence query learnable but a subclass of which is not equivalence and membership query learnable.

Theorem 8.3 *The class n-CYCLES is not FP-learnable with membership and equivalence queries. In particular, n-CYCLES is not FP-learnable with a P-assistant.*

Proof. The proof is a straightforward adversary argument. Suppose there exist an FP-learner α learning n-CYCLES. Let t be the polynomial bounding the number of queries asked by M. Chose large enough prime p such that $(p-2)! > t(p) + 1$. The adversary keeps all the $(p-2)!$ p-cyclic groups with him. Let $(q_1, q_2 \ldots, q_{t(p)})$ be the set of elements queried by α, some of which are membership queries and the rest equivalence queries. For each membership query $q_i \neq e$ by α the adversary answers 'No', and the adversary answers 'Yes' if $q_i = e$. Each equivalence query is of the form q_j, where q_j is a single n-cycle. This is so because each representation in the representation class n-CYCLES consists of a single n-cycle. For this equivalence query the adversary gives q_j as counterexample. Also note α cannot ask an equivalence query with identity as in the case of the learner for SYM because the identity group is not a concept in the representation class n-CYCLES. Now, from Proposition 8.1, and the fact that $(p-2)! > t(p) + 1$, it follows that there are at least 2 p-cyclic groups which are consistent with the answers to the queries. Hence the theorem follows. ■

For the proof of the next theorem we need the following elementary proposition.

Proposition 8.4 *Let $\phi(n) = |\{i | 1 \leq i \leq n; gcd(i,n) = 1\}|$, for $n \in \mathbf{N}$. For each n-cyclic group $G < S_n$, there are precisely $\phi(n)$ n-cycles σ such that $G = \langle \sigma \rangle$.*

Theorem 8.5 *The class* n-CYCLES *is* FP*-learnable with a* UP $\cap$ co-UP*-assistant.*

Proof. We describe a UPSV-learner that learns n-CYCLES with membership queries. On input 0^n, the UPSV-learner first computes $\phi(n)$, then guesses a lexicographically ordered set of $\phi(n)$ n-cycles, and finally makes membership queries corresponding to each guessed n-cycle. The learner outputs the first of the guessed n-cycles, if answers to all the queries are 'Yes'. ∎

We now turn to CYCLIC-SYM. In order to pinpoint its learnability in the TA model as sharply as possible, we prove a lower bound and then an upper bound result.

Theorem 8.6 *The class* CYCLIC-SYM *is not* FP*-learnable with* NP $\cap$ co-NP*-assistants.*

Proof. It suffices to show that CYCLIC-SYM is not NPSV-learnable with membership queries. Suppose α is an NPSV-learner for CYCLIC-SYM. Let t be a polynomial such that $t(n)$ upper bounds the number of membership queries asked by $\alpha(0^n)$. Choose a prime p such that $(p-2)! > t(p)$. Consider the computation of $\alpha(0^p)$, with target group $\langle e\rangle$. Let ρ be a computation path on which α outputs a representation for $\langle e\rangle$. Furthermore, let $(q_1, q_2 \ldots, q_{t(p)})$ be the set of elements of S_p queried by α on the path ρ. Since the target group is $\langle e\rangle$ the answers to all queries except e is 'No'. Since $(p-2)! > t(p)$, Proposition 8.1 implies that there is a p-cycle g such that $\{q_i \mid 1 \leq i \leq t(p)\} - \{e\} \cap \langle g\rangle = \emptyset$. Now consider the same computation path ρ for $\alpha(0^p)$ for target group $\langle g\rangle$. Notice that the answers to queries $(q_1, q_2 \ldots, q_{t(p)}, e)$ is the same whether the target group is $\langle e\rangle$ or $\langle g\rangle$. Therefore, α ends up outputting a representation for $\langle e\rangle$ on path ρ, even for the target group $\langle g\rangle$. This contradicts the definition of NPSV-learnability. ∎

Now we give an upper bound on learning CYCLIC-SYM. For the proof of the next theorem we require the following simple group-theoretic results.

Proposition 8.7 *1. If G is a cyclic subgroup of S_n of order p^k, for prime p, then $p^k \leq n$.*

2. Let $G < S_n$ be a cyclic group such that $|G|$ has prime factorization $p_1^{e_1} \ldots p_l^{e_l}$. Then the number of elements in G of order $p_i^{k_i}$ is $p_i^{k_i} - p_i^{k_i-1}$, for each i and $k_i \leq e_i$.

Proof. The first part immediately follows from the fact if $o(\sigma) = p^k$, for $\sigma \in S_n$ and prime p, then the cycle decomposition of σ has a cycle of length p^k. For the second part, recall that if G is cyclic and d divides $|G|$ there is a unique cyclic subgroup of G of order d. Also, any cyclic group of order d has $\phi(d)$ generators, where ϕ is the Euler totient function. The proposition follows since $\phi(p_i^{k_i}) = p_i^{k_i} - p_i^{k_i-1}$ for $1 \leq i \leq l$. ∎

Theorem 8.8 CYCLIC-SYM *is* FP*-learnable with a* UP*-assistant.*

Proof. For CYCLIC-SYM we define a teaching assistant as follows:
$L(\mathrm{CYCLIC-SYM}) = \{\langle 0^n, p, k, x, G\rangle | G < S_n, p^k \leq n, p$ is a prime, $\exists y \in \Sigma^+$ xy encodes the lex. first $g \in G : o(g) = p^k\}$.

Now we describe a machine ACCEPT-L(CYCLIC-SYM) that accepts L(CYCLIC-SYM) with the target concept as oracle.

ACCEPT-L(CYCLIC-SYM)($\langle 0^n, p, k, x, G\rangle$)
1 $K \leftarrow p^k - p^{k-1}$
2 **if** $p^k > n$ or p not a prime
3   **then reject**
4 **Guess** K distinct strings $x_1, \ldots, x_K$ in lexicographical order
5 **for** $i = 1$ **to** K
6   **do if** x_i does not encode a permutation of S_n
7   **then reject**
8   **else** Compute the order o_i of the element encoded by x_i
9   **if** $o_i \neq p^k$
10   **then reject**
11 **end-of-for**
12 **if** $\exists y$ such that $x_1 = xy$
13   **then accept**
14   **else reject**

It is easy to see that the machine accepts the teaching assistant L(CYCLIC-SYM). Now we will see that the above non-deterministic machine has the required time bound and unambiguous behavior. Let G be the target cyclic group. *lines* 2-3 can be obviously done in time polynomial in n. Since $p^k \leq n$ the nondeterministic step in *line*-4 can also be done in time polynomial in n. Computing the order of $g \in S_n$, in *line*-8 can be done in time polynomial in n by writing the cycle decomposition of g and computing the *lcm* of the cycle lengths in the decompostion. Now, from Proposition 8.7, if there is an element of order p^k, then there are $p^k - p^{k-1}$ elements of order p^k in G. Hence at most *one* non-deterministic guess can be an accepting path after completing the **for**-loop. It follows that L(CYCLIC-SYM)$\in$ UP(CYCLIC-SYM).

Now we describe the CYCLIC-LEARNER for CYCLIC-SYM, which uses the assistant L(CYCLIC-SYM). Let G be the target concept. We first give an intuitive idea of how CYCLIC-LEARNER works. Let $p_1^{e_1} \ldots p_l^{e_l}$ be the prime factorization of $|G|$, where G is the target cyclic group. Let G_i denote the unique subgroup of G of order $p_i^{e_i}$, $1 \leq i \leq l$. If g_i is a generator of G_i, then it is easy to see that $\prod g_i$ generates G. CYCLIC-LEARNER exploits precisely this property: with assistant L(CYCLIC-SYM) it computes a generator g_i for each G_i and then finally computes a generator for G by multiplying the g_i's. The formal description follows.

CYCLIC-LEARNER(0^n)
$\sigma \leftarrow e$
for primes $p \leq n$
    do Compute the largest index d such that $p^d \leq n$
        while $\langle 0^n, p, d, \lambda, G\rangle \notin L$(CYCLIC-SYM)
            do $d \leftarrow d - 1$
        end-of-while

(* With L(CYCLIC-SYM) prefix search for g_p of order p^d *)
$z \leftarrow \lambda$
while $\langle 0^n, p, d, z0, G\rangle$ or $\langle 0^n, p, d, z1, G\rangle \in L$(CYCLIC-SYM)
    do if $\langle 0^n, p, d, z0, G\rangle \in L$(CYCLIC-SYM)
        then $z \leftarrow z0$
        else $z \leftarrow z1$
end-of-while
$g_p \leftarrow z$
$\sigma \leftarrow \sigma g_p$
end-of-for
output σ

■

9 Conclusions

In this paper we introduce and study a new model for exact learning, namely, the teaching assistant model. We compare this model with Angluin's model and also show separations between several teaching assistant classes. This help us in a fine classification of various concept classes with respect to their learnability. To summarize, the separations and containments of the various teaching assistant classes considered are depicted in the following figure. The main technical open problem is to develop an exact learning algorithm for SYM in the TA model with an SPP-assistant.

The TA model for learning is as yet of only theoretical interest. As the next step, we would like to develop more learning algorithms in the TA model for various other concept classes (not necessarily algebraic), and explore further the relationship between the TA model and other well-known models of learning.

Acknowledgments

We are very grateful to the referees for their meticulous refereeing and insightful comments which have considerably improved the presentation of the paper.

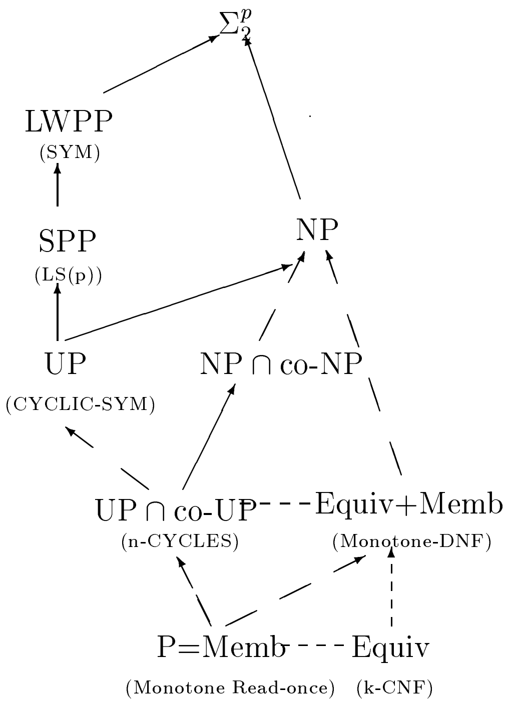

Figure 1. Inclusions and separations of different teaching assistant classes[3]

[3] In the above figure a bold line with arrow indicates inclusion, a dashed line with arrow indicates proper inclusion, and a dashed line without arrow indicates separation in both directions.

References

[1] D. Angluin. Queries and concept learning. *Machine Learning*, 2:319-342, 1988.

[2] D. Angluin. Negative results for equivalence queries. *Machine Learning*, 5:121-150, 1990.

[3] J. L. Balcázar, J. Díaz, and J. Gabarró. *Structural Complexity – I & II.* Springer Verlag, Berlin Hiedelberg, 1988.

[4] N. H. Bshouty, R. Cleve, R. Gavalda, S. Kannan and C. Tamon. Oracles and queries that are sufficient for exact learning. *Journal of Computer and System Sciences*, 52:421-433, 1996.

[5] R. Book, T. Long and A. Selman. Quantitative relativization of complexity classes. *SIAM Journal on Computing,* 13:461-487, 1984.

[6] S. Fenner, L. Fortnow, S. Kurtz. Gap-definable counting classes. *Proc. 6th Structure in Complexity Theory Conference,* 30-42, 1991.

[7] M. Furst, J. E. Hopcroft and E. Luks. Polynomial time algorithms for permutation groups. *Proc. 21st IEEE Symposium of Foundations of Computer Science*, 36-45, 1980.

[8] R. Gavaldà. On the power of equivalence queries. *EUROCOLT*, 193-203, 1993.

[9] R. Gavaldà. *Kolmogorov Randomness and its Application to Structural Complexity Theory.* Doctoral Dissertation, Universitat Politéchnica de Catalunya April 1992.

[10] S. A. Goldman and M. J. Kearns On the complexity of teaching. Journal of Computer and System Sciences 50: 20-31, 1995.

[11] M. Hall. *The Theory of Groups.* Macmillan, New York, 1959.

[12] L. Hellerstein, K. Pillaipakkamnatt, V. Raghavan and D. Vilkins. How many queries are needed to learn? *Proc. of the* 27^{th} *Symposium on the Theory of Computing*, 190-199, 1995.

[13] K. Hoffmann and R. Kunz. *Linear Algebra.* Prentice Hall Inc., 1971.

[14] M. Li and P. M. B. Vitányi. *Kolmogorov Complexity and its applications.* Springer-Verlag, New York, 1993.

[15] A. Sinohara and S. Miyano. Teachability in computational learning. *New Generation Computing*, 8:337-347, 1991

[16] O. Watanabe. A formal study of learning via queries. 17^{th} *International Colloquium on Automata, Languages and Programming,* LNCS 443: 139-152, 1990.

[17] O. Watanabe and R. Gavaldà. Structural analysis of polynomial time query learnability. *Mathematical Systems Theory,* 27:231-256, 1994.