title: Structural Analysis of Polynomial Time Query Learnability

author: Osamu Watanabe and Ricard Gavalda

affiliation: Osamu Watanabe

Department of Computer Science

Tokyo Institute of Technology

Meguro-ku, Ookayama, Tokyo 152, Japan

(watanabe@cs.titech.ac.jp)

Ricard Gavalda

Department of Software (L.S.I.)

Universitat Politecnica de Catalunya

Pau Gargallo 5, 08028 Barcelona, Spain

(gavalda@lsi.upc.es)

footnote:

This is an extended version of a part of the paper "A Formal Study of Learning via Queries", which was presented at the 17th International Colloquium on Automata, Languages and Programming. The first author was supported in part by Grant in Aid for Scientific Research of the Ministry of Education, Science and Culture of Japan under Grant-in-Aid for Co-operative Research (A) 02302047 (1991). The second author was supported in part by ESPRIT II Basic Research Actions Program of the EC under contract No. 3075 (project ALCOM), and by the National Science Foundation under grant CCR89-13584, while visiting the University of California, Santa Barbara.

mailing address:

Osamu Watanabe

Department of Computer Science

Tokyo Institute of Technology

Meguro-ku, Ookayama 2-12-1, Tokyo 152, Japan

ABSTRACT

In this paper, we study computational power of polynomial time query learning systems for several query types, and computational complexity of "learning problem" for several representation classes. As corollaries of our results, we prove some polynomial time nonlearnability results, and relate polynomial time learnability of some representation classes to the complexity of representation finding problems of P/poly oracles. For example, for *CIR*, a representation class by logical circuits, it is shown that $\mathrm{P}(\mathrm{NP}_1^{()})$ is an upper bound of power of query learning systems for *CIR*, and that $\mathrm{P}(\mathrm{NP}_1^{()})$ is also a lower bounds of power of query learning systems for *CIR* when they are used to learn a certain subclass R of *CIR*. It is also shown that the problem of learning *CIR* is $\mathrm{P}(\mathrm{NP}(\mathrm{NP}_1^{()}))$-solvable. Then using these results, the following relations are proved: (1) If for some $A \in \mathrm{P/poly}$, the representation finding problem of A is not in $\mathrm{P}(\mathrm{NP}_1^A)$, then *CIR* is not polynomial time query learnable even by using queries such as membership, equivalence, subset, superset, etc. (2) On the other hand, if the above mentioned subclass R of *CIR* is not polynomial time query learnable by using subset and superset queries, then some $B \in \mathrm{P/poly}$ exists such that its representation finding problem is not in $\mathrm{P}(\mathrm{NP}_1^B)$.

1. Introduction

The following type of learning is called *query learning* or *learning via queries*: a *teacher* has a representation of some set, *concept*, in his mind, and a *learner* tries to identify it by asking queries to the teacher. In the preceding paper [16], we have established a framework for investigating learnability problems in computational complexity theory. Here one such investigation is demonstrated; we discuss relations between polynomial time query learnability and computational complexity of representation finding problems.

One can define many kinds of "polynomial time query learnability" questions by varying the following factors: (1) a learning notion that determines a learning criteria, (2) a computational model (more precisely, a class of learning systems) that is to be used, and (3) a target concept class (more precisely, a representation class) that is to be learned. Here we consider "learnability" in the following context. (See [16] and the next section for the definition of notions and notations.)

(1) For a learning notion, the *bounded-learning* is adopted and used throughout this paper. That is, our learning goal is to obtain a representation that denotes a target set correctly up to a given length.

(2) We use "learning system" to specify learning computation. In general, we consider any query types discussed in [1]: membership query, equivalence query, subset query, superset query, etc. On the other hand, each query is answered by yes/no, yes/counterexample, or yes/min-counterexample, where "min-counterexample" means the lexicographically smallest counterexample. In this section, however, we mainly use the following two particular query-answer types: (i) membership query (answered by yes/no), subset query (answered by yes/min-counterexample), and superset query (answered by yes/min-counterexample), and (ii) subset query (answered by yes/no) and superset query (answered by yes/no). These query-answer types are respectively denoted as (Mem;Sub[min],Sup[min]) and (Sub,Sup;).

(3) A target concept class is specified by "representation class" [10]. For a representation class, we consider classes *DFA*, *NFA*, *CFG*, and *CIR*, which respectively correspond to the nfa, cfg, and circuit representations. (In this paper, by *dfa*, *nfa*, *cfg*, and *circuit*, we mean deterministic finite automaton, nondeterministic finite automaton, context-free grammar, and logical circuit expression[1] respectively.) We sometimes

[1]A logical circuit denotes a set of strings of some fixed length. Thus, in order to express sets containing strings of different length, we use *circuit expressions*. The notion of "circuit expression" is a natural generalization of "regular expression". See Section 2.1.

consider the case where a target set has certain structure in common. For example, we investigate the problem of learning circuit representations that denote sets of the form $L \cup L\#L \cup \cdots$ for some set $L \subseteq \{0,1\}^*$. (Such sets are called *repetitive*.)

For any representation class C and any query-answer type, we take the following strategy for investigating the polynomial time α-learnability of C. First estimate the computational power of α-learning systems for C. Then using this estimation, we relate the computational complexity of certain problems to the polynomial time α-learnability of C. In the following, we explain this strategy in more detail and state our main results.

Query learning may be regarded as some variation of query computation. Thus, we measure "computational power of learning systems" in terms of *query machine types* such as $\mathrm{NP}^{()}$, $\mathrm{P}^{\mathrm{NP}^{()}}$, etc. These machine types are defined inductively; for example, an $\mathrm{NP}^{()}$-machine is a polynomial time nondeterministic machine that may ask queries to an oracle, a $\mathrm{P}^{\mathrm{NP}^{()}}$-machine is a polynomial time deterministic machine that may ask queries to some $\mathrm{NP}^{()}$-machine, etc. By restricting the way of asking queries, some more query machine types are defined. An $\mathrm{NP}_1^{()}$-machine is an $\mathrm{NP}^{()}$-machine that asks one query on each nondeterministic computation path, and $\mathrm{P}^{\mathrm{NP}_1^{()}}$-machine is a polynomial time deterministic machine that may ask queries to some $\mathrm{NP}_1^{()}$-machine.

Consider any representation class C and any query-answer type α. In order to estimate the power of α-learning systems for C, we try to get some query machine types $\mathcal{M}_1^{()}$ and $\mathcal{M}_2^{()}$ that satisfy the following conditions:

(a) Every α-learning system for C is simulated by some query machine of type $\mathcal{M}_1^{()}$.

(b) Every query machine of type $\mathcal{M}_2^{()}$ (that is used to compute a representation of a given oracle set) is simulated by some α-learning system for C.

If $\mathcal{M}_1^{()}$ satisfies (a), it is regarded as an upper bound of the power of α-learning systems for C. On the other hand, a query machine type $\mathcal{M}_2^{()}$ satisfying (b) is considered as a lower bound of the power of α-learning systems for C.

We prove some upper bound and lower bound results for representation classes *DFA*, *NFA*, *CFG*, and *CIR*. For example, we show that $\mathrm{P}^{\mathrm{NP}_1^{()}}$ is an upper bound of the power of (Mem;Sub[min],Sup[min])-learning systems for *CIR*. (Clearly, the same upper bound holds for any weaker query-answer types such as (Sub,Sup;).) On the other hand, we can also show that some (Sub,Sup;)-learning systems for *CIR* can achieve $\mathrm{P}^{\mathrm{NP}_1^{()}}$-computation when they are used to learn repetitive sets. That is, $\mathrm{P}^{\mathrm{NP}_1^{()}}$ is also a lower bound of the power of (Sub,Sup;)-learning systems for *CIR* that are used to learn repetitive sets. (Clearly, the same lower bound holds for any stronger query-answer types such as (Mem;Sub[min],Sup[min]).) Therefore, $\mathrm{P}^{\mathrm{NP}_1^{()}}$ exactly characterizes the power of (Sub,Sup;)-learning systems for *CIR* that are used to learn repetitive sets. (In the same way, $\mathrm{P}^{\mathrm{NP}_1^{()}}$ exactly characterizes the power of (Mem;Sub[min],Sup[min])-learning systems for *CIR* that are used to learn repetitive sets[2].) Similar (but partial) characterizations are obtained for *NFA* and *CFG*.

Take *CIR* for example. We relate the polynomial time learnability of *CIR* to the complexity of circuit representation finding problems. For any set $A \in \mathrm{P/poly}$, FIND-REP(A) is to find, for a given input 0^m, a circuit representation of $A^{\leq m}$ by using A as an oracle set. (P/poly [8] is the class of sets with polynomial size circuits.) For any query machine type $\mathcal{M}^{()}$, we say that *FIND-REP(A) is* $\mathcal{M}^{()}$*-solvable* if some $\mathcal{M}^{()}$-machine solves FIND-REP(A). From the above upper bound result, we can prove that if for some $A \in \mathrm{P/poly}$, FIND-REP(A) is not $\mathrm{P}^{\mathrm{NP}_1^{()}}$-solvable, then *CIR* is not (Mem;Sub[min],Sup[min])-learnable. On the other hand, it follows from the above lower bound result that if circuit representations denoting repetitive sets are not (Sub,Sup;)-learnable, then some $B \in \mathrm{P/poly}$ exists such that FIND-REP(B) is not $\mathrm{P}^{\mathrm{NP}_1^{()}}$-solvable. Therefore, the polynomial time query learnability of *CIR* is closely related to the following question: Is there any $X \in \mathrm{P/poly}$ such that FIND-REP(X) is not $\mathrm{P}^{\mathrm{NP}_1^{()}}$-solvable?

Polynomial time learnability has been discussed in various contexts, and some of those previous works are related to our subject. Angluin [2] proved that, e.g., *DFA* is not polynomial time (;Equ)-learnable, where (;Equ) means to use only equivalence counterexample-queries. Although no assumption is necessary for this result, the proof technique does not seem to apply some other query types nor more general representation classes.

Kearns and Valiant [9] proved that if the RSA function is hard to invert, then *DFA* is not PAC-learnable. Angluin and Kharitonov [3] strengthened it to show that if the RSA function is hard to invert, then *NFA* is not PAC-learnable even if a learner can use membership queries. As a consequence of this result, we have that if the RSA function is hard to invert, then *NFA* is not polynomial time (Mem;Equ)-learnable, where (Mem;Equ) means to use membership queries and equivalence counterexample-queries. In more general, it has been known [13, 7] that *CIR* is not polynomial time (Mem;Equ)-learnable if some cryptographic one-way function exists. Notice that these results use cryptographic assumptions, which concern the "average case" intractability. Such assumptions are in

[2]Hence, (Sub,Sup;) is as powerful as (Mem;Sub[min],Sup[min]). We prove this fact more directly at Proposition 4.5.

some sense necessary because they originally investigate PAC-learnability, i.e., approximate learnability. On the other hand, in this paper, we directly relate the polynomial time query learnability of *CIR* to the "worst case" intractability of representation finding problems.

2. Preliminaries

In this paper we follow standard definitions and notations in computational complexity theory. We assume that the reader is familiar with them and, in particular, with those concerning the structure of complexity classes; the reader will find them in [4].

We use the alphabet $\Sigma = \{0, 1, \#\}$, and by a *string* we mean an element of Σ^*. The length of a string x is denoted by $|x|$. Let $A^{\le n}$, $A^{\ge n}$, and $A^{m\le\cdot\le n}$ denote respectively $\{x \in A : |x| \le n\}$, $\{x \in A : |x| \ge n\}$, and $\{x \in A : m \le |x| \le n\}$. We use a total order $<$ on Σ^* that is defined as follows: (i) $0 < 1 < \#$, and (ii) $x < y$ if either (a) $|x| < |y|$ or (b) $|x| = |y|$ and there is a string z such that $x = zax' \ \wedge \ y = zby' \ \wedge \ a < b$. In this paper, by *lexicographic order* we mean this order.

We assume some pairing function from $\Sigma^* \times \Sigma^*$ to Σ^* that is polynomial time computable and polynomial time invertible. For inputs x and y, we denote the output of the pairing function by $\langle x, y\rangle$; this notation is extended to denote every k tuple. Furthermore, we assume that there exists some constant $d_{\rm pair}$ such that $\forall x_1, ..., x_k\ [\ |\langle x_1, ..., x_k\rangle| \le d_{\rm pair} \sum_{1\le i\le k} |x_i|\ ]$. In order to simplify our notation, we write, e.g., $f(x, y)$ for denoting $f(\langle x, y\rangle)$.

In the following, we prepare necessary notions and notations concerning polynomial time query learning, query machine types, difficulty of learning problems, and power of learning systems.

2.1. Polynomial Time Query Learning

Since the notions and notations are explained in the preceding paper [16], we keep our explanation to a minimum.

We specify a learning problem by using a "representation class" [10, 14].

Definition 2.1. A *representation class* is a three-tuple (R, Φ, ρ), where $R \subseteq \Sigma^*$ is a *representation language*, $\Phi : R \to 2^{\Sigma^*}$ is a *semantic function* or *concept mapping*, and $\rho : R \to \mathbf{N}$ is a *size function*.

In this paper, we assume that $\rho(r) = |r|$ and omit specifying ρ. Furthermore, we assume that every representation class $C = (R, \Phi)$ satisfies the following conditions:

(1) $R \in \mathrm{P}$, and

(2) for some polynomial time machine M,

$$\forall r \in R, \forall u \in \Sigma^* \; [\; u \in \Phi(r) \;\leftrightarrow\; M \text{ accepts } \langle r, u \rangle \;].$$

The second condition is called the *Uniform P-Computability Condition.*

We mainly consider the following representation classes. (Recall that dfa, nfa, and cfg are abbreviations respectively for deterministic finite automaton, nondeterministic finite automaton, and context-free grammar.)

- A representation class for dfa: *DFA*
  Let R_{dfa} be the set of dfa (encoded in Σ^*), and for each $r \in R_{dfa}$, let $\Phi_{dfa}(r)$ be the set of strings accepted by the dfa (denoted by) r. Then *DFA* is defined by $DFA = (R_{dfa}, \Phi_{dfa})$.
- A representation class for nfa: *NFA*
  Let R_{nfa} be the set of nfa (encoded in Σ^*), and for each $r \in R_{nfa}$, let $\Phi_{nfa}(r)$ be the set accepted by the nfa (denoted by) r. Then *NFA* is defined by $NFA = (R_{nfa}, \Phi_{nfa})$.
- A representation class for cfg: *CFG*
  Let R_{cfg} be the set of cfg, and for each $r \in R_{cfg}$, let $\Phi_{cfg}(r)$ be the set generated by the cfg r. Then *CFG* is defined by $CFG = (R_{cfg}, \Phi_{cfg})$.
- A representation class for circuits: *CIR*
  Since each circuit can accept only strings of a fixed length, we extend the notion of circuit so that a circuit can accept strings of various lengths, thereby accepting an *infinite* set of strings. A *circuit expression* is an extended regular expression where we can use circuits for constant symbols. For example, for any two circuits c_1 and c_2, $c_1 + c_2$ denotes a set of strings accepted by either c_1 or c_2. Notice that each circuit in circuit expressions accept strings of some fixed length, and thus, most of algorithms for regular expressions work for circuit expressions with slight modifications. In this paper, by "circuit" we mean a circuit expression. Let R_{cir} be the set of circuits, for each $r \in R_{cir}$, $\Phi_{cir}(r)$ is the language accepted by the circuit r. Now *CIR* is defined by $CIR = (R_{cir}, \Phi_{cir})$.

We use "learning system" for our computation model. A *learning system* $\langle S, T \rangle$ is a pair of a *learner* S and a *teacher* T, where S is a Turing machine and T is an oracle function. Let $\langle S, T \rangle$ be a learning system for some representation class $C = (R, \Phi)$. Before executing $\langle S, T \rangle$, a representation $r \in R$ (whose size is n) is given to T, and an input $\langle n, m \rangle$ is given to both S and T. Then in its computation, S can ask certain types of queries to T, and finally S yields some output (which is usually some representation $r' \in R$). This computation is denoted as $\langle S, T(r)\rangle(n,m)$. The goal of S, in such computation, is to output some representation that denotes $\Phi(r)$ up to length $\leq m$. (This type of learning is called *bounded-learning.*) More precisely, it is to output some "m-representation" for r.

Definition 2.2. Let $C = (R, \Phi)$ be any representation class. For any $r \in R$ and any $m \geq 0$, $r' \in R$ is a *m-representation* for r if $\Phi(r')^{\leq m} = \Phi(r)^{\leq m}$. In general, for any set X and any $m \geq 0$, $r' \in R$ is a *m-representation* for X if $\Phi(r')^{\leq m} = X^{\leq m}$.

A *(Mem;Sub,Sup)-learning system* consists of a learner S and a teacher T, where S asks membership queries (Mem), subset queries (Sub), and superset queries (Sup), and T provides yes/no to each membership query and yes/counterexample to each subset and superset query. Such learners (resp., teachers) are called *(Mem;Sub,Sup)-learners* (resp., *(Mem;Sub,Sup)-teachers*). A tuple such as (Mem;Sub,Sup) is called a *query-answer type.* Table 1 states variations of (Mem;Sub,Sup) considered in this paper. Here by "min-counterexample" we mean the lexicographically smallest counterexample. A query that is answered by yes/no is called a *yes/no-query*, answered by yes/counterexample is called a *counterexample-query*, and answered by yes/min-counterexample is called a *min-counterexample-query.*

query-answer type	query	answer
(Sub,Sup;)	subset, superset	yes/no
(Mem;Equ)	membership equivalence	yes/no yes/counterexample
(Mem;Sub[min],Sup[min])	membership subset, superset	yes/no yes/min-counterexample

Table 1: Query-Answer Types

In order to define "polynomial time learnability", the following two notions are important: "polynomial time learner" and "polynomially bounded teacher". A learner is *polynomial time* if for some polynomial p and for all inputs $\langle n, m\rangle$, it always halts within $p(n+m)$ steps. For any representation class $C = (R, \Phi)$ and any function b on nonnegative integers, a teacher T is a *b-bounded teacher for C* if in every computation $\langle S, T(r)\rangle(n,m)$, T answers to each query correctly w.r.t. $\Phi^{\leq b(n+m)}$. That is, the correctness of an answer from T is guaranteed if the answer concerns only strings of length $\leq b(n+m)$. More precisely, we follow the rule illustrated in Table 2. The bound $b(n+m)$ is called a (*counterexample*) *length bound*, and the function b is called a (*counterexample*) *length bound function* (or, simply *length bound*). For any polynomial b, a b-bounded teacher is generally called a *polynomially bounded teacher*.

Let X be a target set, and let $\langle n, m\rangle$ be an input.

query-answer type	response from a legitimate b-bounded teacher
membership query x (i.e., '$x \in X$?')	'yes' if $\lvert x\rvert \le b(n+m) \wedge x \in X$, 'no' if $\lvert x\rvert \le b(n+m) \wedge x \notin X$, and any answer otherwise.
subset yes/no-query r (i.e., '$\Phi(r) \subseteq X$?')	'yes' if $\Phi(r)^{\le b(n+m)} \subseteq X^{\le b(n+m)}$, and 'no' if $\Phi(r)^{\le b(n+m)} \not\subseteq X^{\le b(n+m)}$.
subset c.ex.-query r (i.e., '$\Phi(r) \subseteq X$?')	'yes' if $\Phi(r)^{\le b(n+m)} \subseteq X^{\le b(n+m)}$, and z if $z \in \Phi(r)^{\le b(n+m)} - X^{\le b(n+m)}$.

Table 2: Legitimate Response

Finally our learning notion — polynomial time bounded-learning — is formally defined as follows.

Definition 2.3. For any representation class $C = (R, \Phi)$, and any query-answer type α, C is *α-learnable from polynomially bounded teachers for C* if

$$\begin{aligned}&\exists S\ : \text{polynomial time } \alpha\text{-learner},\ \exists b\ : \text{polynomial length bound},\\ &\forall T\ : b\text{-bounded } \alpha\text{-teacher for } C,\ \forall r \in R,\ \forall n \ge |r|,\ \forall m \ge 0\\ &[\ \langle S, T(r)\rangle(n,m) \text{ outputs a } m\text{-representation for } r\ ].\end{aligned}$$

Remark. In the following we usually use simpler statements. The above notion is stated more simply as *C is α-learnable*, and the above learner S is called an *α-learner for C*. The notion is sometimes extended to the problem of learning some representation subclass. For any subset Q of R, Q is *α-learnable* if the condition in the above definition holds with R being replaced by Q.

2.2. Machine Types and Complexity Classes

In this paper, we often use "machine type" in order to describe the power of a learning system and th difficulty of a learning problem. Furthermore, since computation achieved by learning systems corresponds to relativized computation, "query machine type" is usually used. For example, we discuss whether a given learning system is "simulated" by some polynomial time deterministic query machine (in short, $\mathrm{P}^{()}$-machine). For this purpose, we clarify concepts concerning "query machine type" and prepare necessary notations.

We use the following Turing machine computation model. In general, machines are *transducers*, and *acceptors* are considered as special kind of transducers that always output 0 (i.e., reject) or 1 (i.e., accept). A transducer is deterministic; on the other hand, an acceptor is either nondeterministic or deterministic. We assume that Turing machines are "oracle machines" (in this paper we call them *query machines*); that is, each machine can ask queries to an *oracle*, where an oracle is either a set or another query machine that is fixed prior to the computation.

Machines are usually denoted as M, M_1, etc. By $M^X(x)$, $M(x)$, and M^X, we mean respectively "M's execution on input x relative to oracle X", "M's execution on input x", and "M with oracle X". When M is a transducer, the notation $M^X(x)$ is also used to denote M's output on input x relative to X.

We define query machine types inductively. A $\mathrm{P}^{()}$*-machine* (resp., $\mathrm{NP}^{()}$*-machine*) is a polynomial time deterministic query machine (resp., polynomial time nondeterministic query acceptor). For any machine type $\mathcal{M}$, $\mathrm{P}^{\mathcal{M}}$-machine is a machine consisting of some $\mathrm{P}^{()}$-machine M_a that uses some $\mathcal{M}$-machine M_b as an oracle. $\mathrm{NP}^{\mathcal{M}}$*-machine* is defined similarly. A machine like M_a is called a *base machine*. In this paper, we consider a robust time bound. That is, for any $\mathrm{P}^{()}$-machine M, there exists a polynomial p such that for every oracle X and every input x, $M^X(x)$ halts within $p(|x|)$ steps.)

Most of our results are stated by using a restricted type of relativized computation. Thus, we introduce the following restricted query machine types:– $\mathrm{NP}^{()}_{\mathrm{m}}$, $\mathrm{NP}^{()}_{\overline{\mathrm{m}}}$, and $\mathrm{NP}^{()}_{1}$. Intuitively, $\mathrm{NP}^{()}_{\mathrm{m}}$-, $\mathrm{NP}^{()}_{\overline{\mathrm{m}}}$-, and $\mathrm{NP}^{()}_{1}$-machine is a $\mathrm{NP}^{()}$-machine that asks one query on each nondeterministic computation path for ever input. Furthermore, on each nondeterministic computation path, $\mathrm{NP}^{()}_{\mathrm{m}}$- (resp., $\mathrm{NP}^{()}_{\overline{\mathrm{m}}}$-) machine enters an accepting state if and only if it receives 'yes' (resp., 'no') to the query. (Recall that nondeterministic machines are always used as acceptors.)

To be more precise, we assume that each $\mathrm{NP}^{()}_{\mathrm{m}}$-machine M is specified by some polynomial p_M and some polynomial time computable function f_M in the following way:

for any oracle set X, and any input string x,

(a) there is a one-to-one and onto correspondence between the set of $M^X(x)$'s nondeterministic paths and $\{0,1\}^{p_M(|x|)}$,

(b) for any $w \in \{0,1\}^{p_M(|x|)}$, $f_M(x,w)$ is the query asked by M on the nondeterministic path (denoted by) w, and

(c) M^X accepts $x \ \leftrightarrow\ \exists w \in \{0,1\}^{p_M(|x|)}$ [$f_M(x,w) \in X$].

$\mathrm{NP}^{()}_{\mathrm{m}}$-machines have similar functions. On the other hand, for each $\mathrm{NP}^{()}_{1}$-machine M, we assume some polynomial p_M, polynomial time computable function f_M, and polynomial time computable predicate E_M that satisfy the above (a) and (b), and the following (c'):

(c') M^X accepts $x \ \leftrightarrow\ \exists w \in \{0,1\}^{p_M(|x|)}$ [$E_M(\,x, w, \chi_X(f_M(x,w))\,)$]
(where χ_X is X's characteristic function).

Finally, let us define language classes by using these machine types. For any oracle X, we consider the following language classes (relative to X):

$$\begin{aligned}
&\mathrm{P}^X\ (=\Sigma_0^{\mathrm{P},X} = \Delta_0^{\mathrm{P},X} = \Delta_1^{\mathrm{P},X}) \stackrel{\mathrm{def}}{=} \{\ L\ :\ L \text{ is accepted by some } \mathrm{P}^X\text{-machine}\ \},\\
&\mathrm{NP}^X\ (=\Sigma_1^{\mathrm{P},X}) \stackrel{\mathrm{def}}{=} \{\ L\ :\ L \text{ is accepted by some } \mathrm{NP}^X\text{-machine}\ \},\\
&\Delta_{2,1}^{\mathrm{P},X} \stackrel{\mathrm{def}}{=} \{\ L\ :\ L \text{ is accepted by some } \mathrm{P}^{\mathrm{NP}_1^X}\text{-machine}\ \},\\
&\Delta_{2}^{\mathrm{P},X} \stackrel{\mathrm{def}}{=} \{\ L\ :\ L \text{ is accepted by some } \mathrm{P}^{\mathrm{NP}^X}\text{-machine}\ \},\\
&\Sigma_{2,1}^{\mathrm{P},X} \stackrel{\mathrm{def}}{=} \{\ L\ :\ L \text{ is accepted by some } \mathrm{NP}^{\mathrm{NP}_1^X}\text{-machine}\ \},\ \text{ and}\\
&\Sigma_{2}^{\mathrm{P},X} \stackrel{\mathrm{def}}{=} \{\ L\ :\ L \text{ is accepted by some } \mathrm{NP}^{\mathrm{NP}^X}\text{-machine}\ \}.
\end{aligned}$$

For other $k \geq 3$, $\Delta_k^{\mathrm{P},X}$ and $\Sigma_k^{\mathrm{P},X}$ are defined similarly. The *polynomial time hierarchy* (in short, *PH*) relative to X is the family of classes $\{\Sigma_k^{\mathrm{P},X}\}_{k\geq 0}$ [12]. Non-relativized classes such as P, NP, etc., are defined to be the classes $\mathrm{P}^{\emptyset}$, $\mathrm{NP}^{\emptyset}$, etc.

2.3. Representation Finding Problem

In order to investigate "complexity of learning problems" in the conventional framework for computational complexity theory, we introduce some classes of problems related to learning problems.

The class P/poly [4] is the class of sets accepted by a sequence of polynomial size circuits. In other words, P/poly is the class of sets L such that each $L^{=n}$ is represented by some circuit representation of size $\leq p(n)$ for some polynomial p. Here we extend this notion in the following way.

Definition 2.4. For any representation class $C = (R, \Phi)$, and for any function p on nonnegative integers, $\mathrm{P}/p(m)[C]$ is the class of sets $L \subseteq \Sigma^*$ such that

$\forall m \geq 0,\ \exists r \in R^{\leq p(m)}\ [\ L^{\leq m} = \Phi(r)^{\leq m}\ ]$.

(I.e., L has a m-representation of length $\leq p(m)$ for every $m \geq 0$.)

P/poly$[C]$ is the class of sets $L \subseteq \Sigma^*$ such that $L \in$ P$/p(m)[C]$ for some polynomial p.
Remark. By a straightforward argument, one can show that for every C, P/poly$[C] \subseteq$ P/poly, and that P/poly$[CIR]$ = P/poly.

Let $C = (R, \Phi)$ be any representation class. For any $A \in$ P/poly$[C]$, we consider the following problem. (Let p be a polynomial such that $A \in$ P$/p(m)[C]$.)

FIND-REP(A): Representation Finding Problem for A.
Instance: 0^m, where m is any nonnegative integer.
Question: By using A as an oracle, find $r \in R$ such that $A^{\leq m} = \Phi(r)^{\leq m}$.
Remark: When the representation class C considered is not clear from the context, the problem is written as FIND-REP($A|C$).

We say that a query machine M *solves* FIND-REP(A) if for every $m \geq 0$, $M^A(0^m)$ yields a required output. We will see that the difficulty of learning C is closely related to the complexity of FIND-REP(A) for some $A \in$ P/poly$[C]$.

Notice that FIND-REP(A) is not a decision problem. For the corresponding decision problem, we consider the following.

PREF-REP(A): Prefix of Representation for A.
Instance: 0^m and u, where $m \geq 0$ and $u \in \Sigma^{\leq p(m)}$.
Question: Is there any $v \in \Sigma^{\leq p(m)-|u|}$ such that $A^{\leq m} = \Phi(uv)^{\leq m}$?

Clearly, some $\mathrm{P}^{()}$-machine solves FIND-REP(A) using PREF-REP(A) as an oracle.

3. Upper Bounds and Their Applications

Our goal is to describe power of polynomial time learners in terms of query machine types. In this section, we discuss upper bounds and their applications. For example, we prove that the power of (Mem;Sup)-learning systems cannot exceed that of $\mathrm{P}^{\mathrm{NP}^{()}_{\mathrm{m}}}$-machines, and then prove non-learnability results based on the limitation of $\mathrm{P}^{\mathrm{NP}^{()}_{\mathrm{m}}}$-machines.

Here we had better clarify the notions of "upper bound" and "lower bound". First consider the intuitive meaning of these notions. Suppose that we are estimating the power of α-learning systems for some representation class C, where α is some query-answer type, and that $\mathcal{M}_1^{()}$ and $\mathcal{M}_2^{()}$ are query machine types with the following relations: (i) every α-learning system for C is simulated by some $\mathcal{M}_1^{()}$-machine, and (ii) every $\mathcal{M}_2^{()}$-machine is simulated by some α-learning system for C. Then, we may intuitively regard $\mathcal{M}_1^{()}$ and $\mathcal{M}_2^{()}$, respectively, as an upper bound and a lower bound of the power of α-learning systems for C.

We first state the meaning of "simulation" precisely, and then define the "upper/lower bound" notions. In the following definitions, let α be any answer-query type, and $C = (R, \Phi)$ be any representation class.

Since we are interested in learning systems, and each learning system takes a pair $\langle n, m\rangle$ for input, we consider query machines that take a pair $\langle 0^n, 0^m\rangle$ for input and simulate a learning system on $\langle n, m\rangle$. (Usually, Turing machine's resource complexity is measured based on input length. In order to adjust to this convention, we choose tally representations, i.e., $\langle 0^n, 0^m\rangle$.)

Definition 3.1. Let b be any polynomial length bound. For any query machine M and any α-learner S for C, M *simulates* S *with a* b*-bounded teacher* if (a) implies (b) for every $r \in R$ and every $n, m \geq 0$. Conversely, if (b) implies (a) for every $r \in R$ and every $n, m \geq 0$, then S *simulates* M *with a* b*-bounded teacher.*

(a) $\forall T$: b-bounded α-teacher for C

  [$\langle S, T(r)\rangle(n, m)$ outputs some m-representation for r].

(b) $M^{\Phi(r)}(0^n, 0^m)$ outputs some m-representation for r.

Definition 3.2. Let $\mathcal{M}^{()}$ be any query machine type.

(1) $\mathcal{M}^{()}$ is an *upper bound* of the power of α-learning systems for C if for every α-learner S for C and every polynomial length bound b, there exists some $\mathcal{M}^{()}$-machine that simulates S with a b-bounded teacher.

(2) $\mathcal{M}^{()}$ is a *lower bound* of the power of α-learning systems for C if for every $\mathcal{M}^{()}$-machine M, there exists some α-learner S and some polynomial length bound b such that S with a b-bounded teacher simulates M.

As for upper bounds, we have the following general results.

Theorem 3.3. Let $C = (R, \Phi)$ be any representation class.

(1) $\mathrm{P}^{\mathrm{NP}_{\mathrm{m}}^{()}}$ is an upper bound of the power of (Mem;Sup[min])-learning systems for C.

(2) $\mathrm{P}^{\mathrm{NP}_{1}^{()}}$ is an upper bound of the power of (Mem;Sub[min],Sup[min])-learning systems for C.

Remark. (1) and (2) respectively holds for any query-answer type weaker than (Mem;Sup[min]) and (Mem;Sub[min],Sup[min]). For example, $\mathrm{P}^{\mathrm{NP}_1^{()}}$ is an upper bound of the power of (Mem;Equ)-learning systems for C.

Proof. Since both (1) and (2) are proved similarly, we prove only (1), and omit the proof of (2).

Let S be any polynomial time (Mem;Sup[min])-learner for C, and let b be any polynomial length bound. We show that some $\mathrm{P}^{\mathrm{NP}_\mathrm{m}^{()}}$-machine simulates S with a b-bounded teacher. More precisely, the machine simulates S with the b-bounded (Mem;Sup[min])-teacher T_C for C that answers as follows in the execution $\langle S, T_C(r)\rangle(n,m)$:

for a membership query x (i.e., '$x \in \Phi(r)$?'),

- $|x| \le b(n+m) \wedge x \in \Phi(r) \to$ 'yes',
- $|x| \le b(n+m) \wedge x \notin \Phi(r) \to$ 'no', and
- $|x| > b(n+m) \to$ 'no';

for a superset query r' (i.e., '$\Phi(r') \supseteq \Phi(r)$?'),

- $\Phi(r')^{\le b(n+m)} \supseteq \Phi(r)^{\le b(n+m)} \to$ 'yes', and
- $\Phi(r')^{\le b(n+m)} \not\supseteq \Phi(r)^{\le b(n+m)} \to$ the lexicographically smallest element in $\Phi(r)^{\le b(n+m)} - \Phi(r')^{\le b(n+m)}$.

Then, by definition, it suffices to describe some $\mathrm{P}^{\mathrm{NP}_\mathrm{m}^{()}}$-machine M with the following property:

$$\forall r \in R,\ \forall n,m \ge 0\ [\ M^{\Phi(r)}(0^n, 0^m) \text{ outputs } \langle S, T_C(r)\rangle(n,m)\ ]$$

We first show that T_C's answer to a superset query is computed by some $\mathrm{P}^{\mathrm{NP}_\mathrm{m}^{()}}$-machine M_1; more precisely, $M_1^{\Phi(r)}(0^n, 0^m, r')$ gives the same answer as $T_C(r)$ does to the superset query r'. For any $r \in R$, define $L(r)$ by

$L(r) \ = \ \{\ \langle 0^l, r', u\rangle : u$ is a prefix of some element in $U(r, r', l)\ \}$,
where $U(r,r',l)$ is the set of strings w such that $|w| = l$ and $w \in \Phi(r) - \Phi(r')$.

For any given r' and l, one can get the lexicographically smallest element in $\Phi(r) - \Phi(r')$ of length l by using standard binary search w.r.t. $L(r)$. Furthermore, by changing l from 0 to $b(n+m)$, one can obtain the lexicographically smallest counterexample to the superset query '$\Phi(r') \supseteq \Phi(r)$?' if any counterexample exists within the length bound $b(n+m)$. Hence, using $L(r)$ as an oracle, some $\mathrm{P}^{()}$-machine computes $T_C(r)$'s answer to every superset query. On the other hand, from the above definition of $L(r)$ one can easily define some $\mathrm{NP}^{()}_{\mathrm{m}}$-machine M that accepts $L(r)$ using $\Phi(r)$ as an oracle. That is, define M to accept $\langle 0^l, r', u\rangle$ if and only if there exists some v such that $|uv| = l$ and $uv \in \Phi(r) - \Phi(r')$. Thus, combining them, we obtain a $\mathrm{P}^{\mathrm{NP}^{()}_{\mathrm{m}}}$-machine M_1 such that $M_1^{\Phi(r)}$ computes $T_C(r)$'s answer to each superset query. We should note here that one fixed M_1 works for every $r \in R$.

Now, with an oracle $\Phi(r)$, we simulate $\langle S, T_C(r)\rangle(n,m)$ in the following way: (i) every time S asks a membership query, just ask the query to the oracle $\Phi(r)$, and (ii) every time S asks a superset query r', execute $M_1^{\Phi(r)}(0^n, 0^m, r')$ and obtain the answer. It is easy to show that some $\mathrm{P}^{\mathrm{NP}^{()}_{\mathrm{m}}}$-machine M achieves this simulation. That is, M is the desired machine. □

Now we have some upper bounds on the power of learning systems, our next task is to use them for investigating the learnability of a given representation class C. First we show some general relationship between the learnability of C and the complexity of representation finding problems for sets in P/poly[C].

Theorem 3.4. Let $C = (R, \Phi)$ be any one of the following four representation classes:- *DFA*, *NFA*, *CFG*, and *CIR*. For any query-answer type α, let $\mathcal{M}^{()}$ be a query machine type that is an upper bound of the power of α-learning systems for C. Then if some $A \in$ P/poly[C] exists such that no $\mathcal{M}^{()}$-machine solves FIND-REP($A|C$), then C is not α-learnable.

Remark. $C = (R, \Phi)$ can be any representation class that satisfies the following for every $A \in$ P/poly[C].

$$\exists q: \text{ polynomial},\ \forall m \geq 0,\ \exists r \in R^{\leq q(m)}\ [\ A^{\leq m} = \Phi(r)\ ].$$

It is not hard to show that *DFA*, *NFA*, *CFG*, and *CIR* have this property.

Proof. We show that if C is α-learnable, then for every $A \in$ P/poly[C], FIND-REP(A) is solved by some $\mathcal{M}^{()}$-machine. (Throughout this proof, FIND-REP(A) means FIND-REP($A|C$).)

Now assume that C is α-learnable; that is, there exist some α-learner S and polynomial length bound b such that

$$\begin{array}{l}\forall T: b\text{-bounded } \alpha\text{-teacher for } C,\ \forall r \in R,\ \forall n \geq |r|,\ \forall m \geq 0 \\ [\ \langle S, T(r)\rangle(n,m) \text{ outputs some } m\text{-representation for } r\ ].\end{array}$$

Since $\mathcal{M}^{()}$ is an upper bound of the power of α-learning systems for C, some $\mathcal{M}^{()}$-machine M_1 simulates S with a b-bounded teacher. Thus,

$\forall r \in R,\ \forall n \geq |r|,\ \forall m \geq 0$
$[\ M_1^{\Phi(r)}(0^n, 0^m)$ outputs some m-representation for r $]$.

Consider any set $A \in \mathrm{P/poly}[C]$. Let q be a polynomial and $\{r_m\}_{m\geq 0}$ be a set of representations such that $\forall m \geq 0\ [\ |r_m| \leq q(m)\ \wedge\ A^{\leq m} = \Phi(r_m)\ ]$ (see the above Remark). We modify M_1 to define M_2 such that $M_2^X(0^m) = M_1^X(0^{q(m)}, 0^m)$. Then for any $m \geq 0$, $M_2^{A^{\leq m}}(0^m) = M_1^{A^{\leq m}}(0^{q(m)}, 0^m) = M_1^{\Phi(r_m)}(0^{q(m)}, 0^m)$. Recall that $M_1^{\Phi(r_m)}(0^{q(m)}, 0^m)$ is some m-representation for r_m; that is, it is a m-representation of A since $\Phi(r_m)^{\leq m} = A^{\leq m}$. Hence, for every $m \geq 0$, $M_2^{A^{\leq m}}(0^m)$ outputs a m-representation of A. Then, by modifying M_2, we obtain M_3 such that $M_3^A(0^m) = M_2^{A^{\leq m}}(0^m)$. Clearly, M_3 solves FIND-REP(A). $\square$

Thus, from our upper bound results, we have the following observations.

Corollary 3.5. Consider any one of the following representation class $C = (R, \Phi)$:- *DFA*, *NFA*, cfg, and *CIR*. (Or, in more general, C can be any representation class satisfying the property stated in Remark of Theorem 3.4.)

(1) If some $A \in \mathrm{P/poly}[C]$ exists for which no $\mathrm{P}^{\mathrm{NP}^{()}_{\mathrm{m}}}$-machine solves FIND-REP($A|C$), then C is not (Mem;Sup[min])-learnable.

(2) If some $A \in \mathrm{P/poly}[C]$ exists for which no $\mathrm{P}^{\mathrm{NP}^{()}_{1}}$-machine solves FIND-REP($A|C$), then C is not (Mem;Sub[min],Sup[min])-learnable.

Remark. For example, letting $C = CIR$, we have the following relations.

(3) If some $A \in \mathrm{P/poly}$ exists for which no $\mathrm{P}^{\mathrm{NP}^{()}_{\mathrm{m}}}$-machine solves FIND-REP(A), then *CIR* is not (Mem;Sup[min])-learnable.

(4) If some $A \in \mathrm{P/poly}$ exists for which no $\mathrm{P}^{\mathrm{NP}^{()}_{1}}$-machine solves FIND-REP(A), then *CIR* is not (Mem;Sub[min],Sup[min])-learnable.

This corollary suggests one way to prove nonlearnability for a given representation class C. For example, in order to prove *CIR* is not (Mem;Sup[min])-learnable, it suffices to show some set $A \in \mathrm{P/poly}$ whose representation finding problem is not solvable by any $\mathrm{P}^{\mathrm{NP}^{()}_{\mathrm{m}}}$-machine. Indeed, the following theorem states that we have such a set for representation classes *DFA*, *NFA*, *CFG*, and *CIR*.

Theorem 3.6. There exists $A_1 \in \mathrm{P/poly}[DFA]$ such that

(1) no $\mathrm{P}^{\mathrm{NP}^{()}_{\mathrm{m}}}$-machine solves FIND-REP($A_1|DFA$),

(2) no $\mathrm{P}^{\mathrm{NP}^{()}_{\mathrm{m}}}$-machine solves FIND-REP($A_1|NFA$),

(3) no $\mathrm{P}^{\mathrm{NP}^{()}_{\mathrm{m}}}$-machine solves FIND-REP($A_1|CFG$), and

(4) no $\mathrm{P}^{\mathrm{NP}^{()}_{\mathrm{m}}}$-machine solves FIND-REP($A_1|CIR$).

Remark. Since P/poly[DFA] $\subseteq$ P/poly[NFA] $\subseteq$ P/poly[CFG] $\subseteq$ P/poly[CIR], A_1 belongs to P/poly[NFA], P/poly[CFG], and P/poly[CIR].

Proof. We prove only (1); the others are proved similarly.

Let $\{M_i\}_{i\geq 1}$ be an enumeration of $\mathrm{P}^{\mathrm{NP}^{()}_{\mathrm{m}}}$-machines, where each M_i consists of $\mathrm{P}^{()}$-machine $M_{i,1}$ that uses $\mathrm{NP}^{()}_{\mathrm{m}}$-machine $M_{i,2}$ as an oracle. A_1 is defined to be $\cup_{k\geq 0} A_{1,i}$ where each $A_{1,i}$ is constructed by stage to diagonalize M_i.

At each stage $i \geq 0$, choose m to be the smallest integer that is large enough so that (i) no string of length m_i has been appeared in the previous stages, and (ii) the number of queries in $M_i(0^m)$ is less than 2^m. Let $X = \cup_{0\leq k\leq i-1} A_{1,k}$, and consider the execution of $M_i^{X\cup\Sigma^m}$ on input 0^m. In the execution, $M_{i,1}$ asks some queries $y_1, y_2, ..., y_t$ to $M_{i,2}$, and for each y_j, $M_{i,2}$ gives an answer by asking nondeterministically to the oracle set X. Notice that for each y_j and for each nondeterministic computation path on y_j, $M_{i,2}$ asks one query, and it enters an accepting state if and only if the query is in the oracle set. For each j, $1 \leq j \leq t$, let v_j be the query string asked by $M_{i,2}$ on the leftmost accepting path in the computation for the jth query y_j asked by $M_{i,1}$. (If $M_{i,2}(y_j)$ has no accepting path, then just let v_j be the null string.)

Now we define $A_{1,i}$ as follows. There are two cases. Suppose that $M_i^{X\cup\Sigma^m}(0^m)$ yields some dfa representation, and this dfa accepts some string $w \notin \{v_1, ..., v_t\}$. Then, define $A_{1,i} = \Sigma^m - \{w\}$. Otherwise (i.e., $M_i^{X\cup\Sigma^m}(0^m)$ does not yield any dfa representation, or the yielded dfa rejects some (in fact, every) string of length m), we let $A_{1,i} = \Sigma^m$. Note that, in both cases, $M_i^{X\cup A_{1,i}}(0^m)$ executes exactly the same way as $M_i^{X\cup\Sigma^m}(0^m)$ and yields the same output. From this, it is clear that M_i cannot solve FIND-REP($A_1|DFA$).

Notice that for every $m \geq 0$, $A_1^{=m}$ is either $\emptyset$, Σ^m, or $\Sigma^m - \{w\}$. Thus, every $A_1^{\leq m}$ is accepted by some polynomial size dfa. □

Corollary 3.7. None of DFA, NFA, CFG, and CIR is (Mem;Sup[min])-learnable.

Remark. For query-answer type (Mem;Sub[min]), we can prove theorems corresponding to Theorems 3.3, 3.5, and 3.6 similarly. (In this case, $\mathrm{NP}^{()}_{\overline{\mathrm{m}}}$-machines are used instead of $\mathrm{NP}^{()}_{\mathrm{m}}$-machines.) Thus, none of DFA, NFA, CFG, and CIR is (Mem;Sub[min])-learnable.

We have demonstrated one way to get non-learnability results by using observations like Corollary 3.5. It should be mentioned here that the above non-learnability results are also provable by some direct argument [1].

An interesting open problem is to get new non-learnability results following our strategy. For example, from Corollary 3.5 (4), the following condition is sufficient to prove *CIR* is not (Mem;Sub[min],Sup[min])-learnable:

(*) Some $A \in$ P/poly exists such that no $\mathrm{P}^{\mathrm{NP}_1^{()}}$-machine solves FIND-REP(A).

Thus, the existence of such A is an interesting open question. On the other hand, one may suspect that condition (*) is too strong and may not be necessary for obtaining the non-learnability result. In the next section, however, we prove that the converse of Corollary 3.5 (4) holds for some representation subclass R_{cir}^{rep} of R_{cir}. That is, in order to prove non-learnability of R_{cir}^{rep}, condition (*) is indeed necessary.

4. Lower Bounds and Their Applications

In this section, we show some lower bound results and their applications. For example, it is proved that some (Mem;Sub[min],Sup[min])-learning systems can achieve $\mathrm{P}^{\mathrm{NP}_1^{()}}$-computation when they are used to learn some subclass R_{cir}^{rep} of R_{cir}; that is, in this context, learning systems are at least as powerful as $\mathrm{P}^{\mathrm{NP}_1^{()}}$-machines, or $\mathrm{P}^{\mathrm{NP}_1^{()}}$ is a lower bound of the power of learning systems. As a consequence of this lower bound result, we show that the converse of Corollary 3.5 (4) holds for R_{cir}^{rep}; that is, if R_{cir}^{rep} is not (Mem;Sub[min],Sup[min])-learnable, then there is some B in P/poly such that no $\mathrm{P}^{\mathrm{NP}_1^{()}}$-machine solves FIND-REP($B|$*CIR*).

Here we consider the problem of learning sets with certain structure, namely, repetitive sets.

Definition 4.1. A subset of Σ^* is *repetitive* if it is of the form $L \cup L\#L \cup L\#L\#L \cup \ldots$ for some $L \subseteq \{0,1\}^*$.

For each $L \subseteq \{0,1\}^*$, define $Rep(L)$ to be $L \cup L\#L \cup L\#L\#L \cup \ldots$. Hence, a set D is repetitive if and only if $D = Rep(L)$ for some $L \subseteq \{0,1\}^*$. Let R_{cir}^{rep} be the set of representations in R_{cir} that denote repetitive sets. In the following discussion, we investigate the power of learning systems when learners can assume that a target set has this repetitive structure. Although this assumption is important for obtaining our lower bound results, it can be replaced by some similar assumption on the target set structure.

Our goal is to show that some (Sub,Sup;)-learning system for *CIR* has $\mathrm{P}^{\mathrm{NP}_1^{()}}$-computational power when it is used to learn R_{cir}^{rep}. Recall that by (Sub,Sup;), we mean learning systems that use only subset and superset *yes/no*-queries, and that $\mathrm{P}^{\mathrm{NP}_1^{()}}$ is an upper bound of the power of many kinds of learning systems including (Sub,Sup;)-learning systems. Thus, this upper bound is best possible for any other answer-query types that are stronger than (Sub,Sup;).

Here we first consider the situation where learners are given some two elements y_1 and y_2 of a target set in advance, and show that some (Sub,Sup;)-learning system has the $\mathrm{P}^{\mathrm{NP}_1^{()}}$-computational power. (Then later, we discuss how to obtain these y_1 and y_2.)

Theorem 4.2. For any $\mathrm{P}^{\mathrm{NP}_1^{()}}$-machine M, there exist a polynomial time (Sub,Sup;)-learner S for *CIR* and a polynomial length bound b such that

$$\begin{array}{l} \forall T: \ b\text{-bounded (Sub,Sup;)-teacher for } \mathit{CIR}, \\ \forall r \in R_{cir}^{\mathrm{rep}},\ \forall n \geq |r|,\ \forall m \geq 0\ \ \forall y_1, y_2 \in \{0,1\}^{\leq m} \cap \Phi_{cir}(r) \\ [\ M^{\Phi_{cir}(r)}(0^n, 0^m) = \langle S, T(r)\rangle(n, m, y_1, y_2)\ ]. \end{array}$$

Remark. For any input $\langle n, m, y_1, y_2\rangle$, $b(n+m)$ is used for the length bound.

Intuitively, this theorem claims that, in the above situation, every $\mathrm{P}^{\mathrm{NP}_1^{()}}$-machine can be simulated by some polynomial time (Sub,Sup;)-learner. The key idea is the way to simulate $\mathrm{NP}_{\mathrm{m}}^{()}$- (resp., $\mathrm{NP}_{\overline{\mathrm{m}}}^{()}$-) computation by using superset (resp., subset) queries, which is illustrated in the proofs of the following lemmas.

For the following discussion, we extend the notion of learning system so that learning systems can be used as acceptors. For any learner S and teacher T, a learning system $\langle S, T\rangle$ is regarded as an acceptor if S always outputs 0 (i.e., reject) or 1 (i.e., accept) on every input. Here both the time bound of S and the length bound for T are determined depending on input length.

Lemma 4.3. For any $\mathrm{NP}_{\mathrm{m}}^{()}$-acceptor M, there exist a polynomial time (Sup;)-learner S for *CIR* and a polynomial length bound b such that

$$\begin{array}{l} \forall T: \ b\text{-bounded (Sup;)-teacher for } \mathit{CIR}, \\ \forall r \in R_{cir}^{\mathrm{rep}},\ \forall x \in \Sigma^*,\ \forall y_1, y_2 \in \{0,1\}^* \cap \Phi_{cir}(r) \\ [\ M^{\Phi_{cir}(r)} \text{ accepts } x\ \leftrightarrow\ \langle S, T(r)\rangle(x, y_1, y_2) = 1\ ]. \end{array}$$

Remark. The theorem holds for any polynomial length bound $b' \geq b$.

Proof. Recall our assumption on $\mathrm{NP}_{\mathrm{m}}^{()}$-acceptors (see Section 2.2). The computation of $\mathrm{NP}_{\mathrm{m}}^{()}$-machine M is specified by some polynomial p_M and some polynomial time computable function f_M in the following way:

$$\forall X, \forall x \; [\; M^X \text{ accepts } x \;\leftrightarrow\; \exists w \in \{0,1\}^{p_M(|x|)} \; [\; f_M(x,w) \in X \;]\;].$$

We also assume some polynomial q_M such that $M^X(x)$ halts within $q_M(|x|)$ steps. Thus, for every $w \in \{0,1\}^{p_M(|x|)}$, the length of $f_M(x,w)$ is at most $q_M(|x|)$.

We first develop a technique for simulating M by using superset queries. Let $X \subseteq \Sigma^*$ be any oracle set. Here we assume that $X = \mathit{Rep}(Z)$ for some $Z \in \{0,1\}^*$ (i.e., X is repetitive) and that z_0 and z_1 are two elements of Z.

For any x, define $Q(x, z_0, z_1)$ as follows:

$$\begin{aligned} Q(x,z_0,z_1) = \{ \;\; & u_1\#\cdots\#u_k\#f_M(x,w) \;:\; (\text{where } k = p_M(|x|)) \\ & (1)\ \forall i, 1 \le i \le k \; [\; u_i \in \{z_0, z_1\} \;],\ \text{and} \\ & (2)\ w \text{ is a string in } \{0,1\}^k \text{ such that } \forall i, 1 \le i \le k \; [\; u_i = z_{w(i)} \;], \\ & \quad (\text{where } w(i) \text{ denotes the } i\text{th symbol of } w) \;\; \}. \end{aligned}$$

For the sake of simplicity, an element $u_1\#\cdots\#u_k\#f_M(x,w)$ of $Q(x,z_0,z_1)$ is abbreviated as $u\#f$. It should be noticed here that the above w is uniquely determined for every $u\#f \in Q(x,z_0,z_1)$.

Let $\overline{Q(x,z_0,z_1)}$ denote the set $\Sigma^* - Q(x,z_0,z_1)$. Then we have the following relation (let $k = p_M(|x|)$):

$$\begin{aligned} M^X \text{ accepts } x \;\; & \leftrightarrow\; \exists w \in \{0,1\}^k \; [\; f_M(x,w) \in X \;], \\ & \leftrightarrow\; \exists w \in \{0,1\}^k \; [\; f_M(x,w) \in \mathit{Rep}(Z) \;], \\ & \leftrightarrow\; \text{there exists } u\#f \text{ in } \mathit{Rep}(Z) \cap Q(x,z_0,z_1), \\ & \leftrightarrow\; \text{some string exists in } \mathit{Rep}(Z) - \overline{Q(x,z_0,z_1)}, \\ & \leftrightarrow\; X - \overline{Q(x,z_0,z_1)} \ne \emptyset, \\ & \leftrightarrow\; \overline{Q(x,z_0,z_1)} \not\supseteq X. \end{aligned}$$

Thus, a yes/no answer to superset query '$\overline{Q(x,z_0,z_1)} \supseteq X$?' determines whether x is accepted by M^X. This is the idea of simulating M by using superset queries.

Now define a learner S and a bound b that satisfy the lemma. First, define b to be a polynomial such that for every $x \in \Sigma^*$ and every $y_1, y_2 \in \{0,1\}^*$, $b(|\langle x, y_1, y_2\rangle|)$ is larger than the length of strings $u\#f$ in $Q(x,y_1,y_2)$. Clearly, such a polynomial b exists because $|u\#q| \le p_M(|x|) \cdot (\max\{|y_1|,|y_2|\}+1) + q_M(|x|)$. Next we define S. Let T be any b-bounded (Sup;)-teacher for *CIR*. Consider any $r \in R^{\mathrm{rep}}_{cir}$ and any input $\langle x, y_1, y_2\rangle$. Since $r \in R^{\mathrm{rep}}_{cir}$, $\Phi_{cir}(r) = \mathit{Rep}(Y)$ for some $Y \subseteq \{0,1\}^*$; furthermore, since

$y_1, y_2 \in \{0,1\}^* \cap Rep(Y)$, we can assume that $y_1, y_2 \in Y$. Now our S (with teacher $T(r)$ and on input $\langle x, y_1, y_2 \rangle$) executes as follows: S generates the representation of $\overline{Q(x, y_1, y_2)}$ (i.e., a string $r_Q \in R_{cir}$ such that $\Phi_{cir}(r_Q) = \overline{Q(x, y_1, y_2)}$), and asks superset query r_Q (i.e., '$\overline{Q(x, y_1, y_2)} \supseteq \Phi_{cir}(r)$?'). Then S accepts the input if and only if it receives a negative answer to the query. Notice that if a counterexample to the query exists, it is found within the length bound. Hence, S receives a negative answer if and only if $\overline{Q(x, y_1, y_2)} \not\supseteq \Phi_{cir}(r)$. Thus, it follows from the relation established above that $\langle S, T(r) \rangle$ accepts $\langle x, y_1, y_2 \rangle \leftrightarrow M^{\Phi_{cir}(r)}$ accepts x. Therefore, S satisfies the lemma. $\square$

Following an argument similar to the above, we have the following lemma.

Lemma 4.4. For any $\mathrm{NP}^{()}_{\overline{\mathrm{m}}}$-acceptor M, there exist a polynomial time (Sub;)-learner S for *CIR* and a polynomial length bound b such that

$$\begin{aligned}
&\forall T: \ b\text{-bounded (Sub;)-teacher for } \mathit{CIR}, \\
&\forall r \in R^{\mathrm{rep}}_{cir}, \ \forall x \in \Sigma^*, \ \forall y_1, y_2 \in \{0,1\}^* \cap \Phi_{cir}(r) \\
&[\ M^{\Phi_{cir}(r)} \text{ accepts } x \ \leftrightarrow \ \langle S, T(r) \rangle (x, y_1, y_2) \text{ accepts } x\ ].
\end{aligned}$$

Proof. For a given $\mathrm{NP}^{()}_{\overline{\mathrm{m}}}$-machine M and any repetitive oracle set X, let Z and $Q(x, z_0, z_1)$ be the same sets in the proof of Lemma 4.3. Then we have the following relation (let $k = p_M(|x|)$):

$$\begin{aligned}
M^X \text{ accepts } x \ &\leftrightarrow \ \exists w \in \{0,1\}^k \ [\ f_M(x,w) \notin X\ ], \\
&\leftrightarrow \ \exists w \in \{0,1\}^k \ [\ f_M(x,w) \notin Rep(Z)\ ], \\
&\leftrightarrow \ \text{there exists } u\#f \text{ in } Q(x, z_0, z_1) - Rep(Z), \\
&\leftrightarrow \ Q(x, z_0, z_1) \not\subseteq X.
\end{aligned}$$

Thus, a yes/no answer to subset query '$Q(x, z_0, z_1) \subseteq X$?' determines whether x is accepted by M^X. The rest of the proof is similar. $\square$

Proof of Theorem 4.2. Let M be any $\mathrm{P}^{\mathrm{NP}^{()}_1}$-machine; that is, M consists of a $\mathrm{P}^{()}$-machine M_a that uses a $\mathrm{NP}^{()}_1$-acceptor M_b as an oracle. First we show that M is simulated by some $\mathrm{P}^{()}$-machine M_1 by using two oracle machines, $\mathrm{NP}^{()}_{\mathrm{m}}$-machine M_2 and $\mathrm{NP}^{()}_{\overline{\mathrm{m}}}$-machine M_3.

From our assumption on $\mathrm{NP}^{()}_1$-machines (see Section 2.2), we have some polynomial p, polynomial time computable function f, and polynomial time computable predicate E such that

$$\forall X, \forall x\ [\ M_b^X \text{ accepts } x \ \leftrightarrow \ \exists w \in \{0,1\}^{p(|x|)}\ [\ E(x, w, \chi_X(f(x,w)))\ ]\ ].$$

Then, by modifying M_b, we can construct $\mathrm{NP}^{()}_{\mathrm{m}}$-machine M_2 and $\mathrm{NP}^{()}_{\overline{\mathrm{m}}}$-machine M_3 that have the following properties for the above p, f, and E:

for every oracle set X and every input x,
M_2^X accepts $x \;\leftrightarrow\; \exists w \in \{0,1\}^{p(|x|)}$ [$E(x,w,1) \;\wedge\; f(x,w) \in X$], and
M_3^X accepts $x \;\leftrightarrow\; \exists w \in \{0,1\}^{p(|x|)}$ [$E(x,w,0) \;\wedge\; f(x,w) \notin X$].

Hence, for every oracle set X and every input x,

$$M_b^X \text{ accepts } x \;\leftrightarrow\; M_2^X \text{ accepts } x \;\vee\; M_3^X \text{ accepts } x.$$

Thus, clearly some $\mathrm{P}^{()}$-machine M_1 simulates M by using M_2 and M_3 as oracles.

From Lemma 4.3 and 4.4, we have some S_2 and b_2 satisfying Lemma 4.3 for M_2, and S_3 and b_3 satisfying Lemma 4.4 for M_3. Using them, we define S and b that satisfy the theorem for M.

First consider S. For a given input $\langle n,m,y_1,y_2\rangle$, S simulates $M_1(0^n,0^m)$ as follows. For each time M_1 queries x to M_2, S executes S_2 on input $\langle x,y_1,y_2\rangle$ thereby obtaining the answer to the query. Similarly, S gets answers to queries to M_3 by using S_3. Then it is clear from the above discussion that S simulates M correctly (if the length bound b is appropriately defined). Furthermore, since M_1, S_2, and S_3 are polynomial time, the simulation halts within some polynomial time.

Next we define b. Consider again the simulation of $M_1(0^n,0^m)$ by S, where S is given $\langle n,m,y_1,y_2\rangle$ as an input. We may assume that $|y_1|,|y_2| \le m$, and that M_1 never asks a query longer than $p(n+m)$ for some polynomial p. In order to receive correct answers from any b-bounded teacher during the simulation, it suffices that b satisfies the following inequalities:

for every query x asked to M_2 during the simulation, $b_2(|\langle x,y_1,y_2\rangle|) \le b(n+m)$, and
for every query x asked to M_3 during the simulation, $b_3(|\langle x,y_1,y_2\rangle|) \le b(n+m)$.

Now define $b(n+m)$ to be any polynomial bounding above $\max\{b_2(d_{\mathrm{pair}}(p(n+m)+2m)), b_3(d_{\mathrm{pair}}(p(n+m)+2m))\}$, where d_{pair} is the constant for our pairing function (see Section 2). Then for every query x asked to M_2 during the simulation, $b_2(|\langle x,y_1,y_2\rangle|)$ $\le b_2(d_{\mathrm{pair}}(p(n+m)+m+m)) \le b(n+m)$. Similarly, for every query x asked to M_3 during the simulation, $b_3(|\langle x,y_1,y_2\rangle|) \le b_3(d_{\mathrm{pair}}(p(n+m)+m+m)) \le b(n+m)$. Hence, b satisfies the above inequalities. $\square$

Next consider more general case, i.e., the case where learners are not given any element of a target set in advance. Here we have the following simple strategy: use two superset min-counterexample-queries to get two necessary elements of a target set, and then with these two elements, achieve the simulation discussed above. We essentially follow this strategy, but we use some technique to avoid min-counterexample queries.

The following proposition, which is of interest by itself, claims that any superset (resp., subset) min-counterexample query is simulated by some superset (resp., subset) yes/no-queries.

Proposition 4.5. For any $r \in R_{cir}$, consider the process of learning r. Let b be the length bound in the process.

(1) For any superset query r' (i.e., '$\Phi_{cir}(r') \supseteq \Phi_{cir}(r)$?'), one can obtain the lexicographically smallest counterexample (if it exists in $\Sigma^{\leq b}$) by using at most b superset yes/no-queries.

(2) For any subset query r' (i.e., '$\Phi_{cir}(r') \subseteq \Phi_{cir}(r)$?'), one can obtain the lexicographically smallest counterexample (if it exists in $\Sigma^{\leq b}$) by using at most b subset yes/no-queries.

Remark.

(1) Part (1) (resp., (2)) holds for any representation class $C = (R, \Phi)$ satisfying the following (a) and (b) (resp., (a) and (c)). Clearly, *DFA* and *CIR* satisfy (a), (b), and (c), and *NFA* and *CFG* satisfy (a) and (b).

  (a) A representation $r \in R$ of $\mathit{Upper}(x)$ is polynomial time computable from $x \in \Sigma^*$. Where $\mathit{Upper}(x)$ is the set of strings that are lexicographically larger than or equal to x.

  (b) A representation $r \in R$ of $\Phi(r_1) \cup \Phi(r_2)$ is polynomial time computable from r_1 and r_2.

  (c) A representation $r \in R$ of $\Phi(r_1) - \Phi(r_2)$ is polynomial time computable from r_1 and r_2.

(2) As an immediate corollary, we have the following relation: *CIR* is (Mem;Sub[min], Sup[min])-learnable if and only if it is (Sub,Sup;)-learnable.

Proof. We prove part (1); the proof of part (2) is similar and thus omitted.

We consider the process of learning r, and let b be the length bound in the process. Let r' be a superset query to which we want to compute the lexicographically smallest counterexample. We may assume that some counterexample exists in $\Sigma^{\leq b}$. (Otherwise, we get 'yes' to the superset yes/no-query r'.) Let y_0 be the lexicographically smallest counter example; that is, y_0 is the smallest element in $\Phi_{cir}(r)^{\leq m} - \Phi_{cir}(r')^{\leq m}$.

For any $y \in \Sigma^*$, let r_y be a representation of $\mathit{Upper}(y) \cup \Phi_{cir}(r')$. Then, for every $y_1 \le y_0$ and $y_2 > y_0$, we have $\Phi_{cir}(r_{y_1}) \supseteq \Phi_{cir}(r)$ and $\Phi_{cir}(r_{y_2}) \not\supseteq \Phi_{cir}(r)$; hence, the answer to superset queries r_{y_1} and r_{y_2} is respectively 'yes' and 'no'. In particular, 'yes' to r_λ, and 'no' to $r_{0^{b+1}}$ (where λ denotes the null string). Thus, our task is to obtain y such that superset queries r_y and $r_{y'}$ receive respectively 'yes' and 'no' (where y' is the next string under the lexicographic order). Such y is computed by the following binary search:

begin
  $y_1 := \lambda$; $y_2 := 0^{b+1}$;
  while $(y_1)' < y_2$ **do**
    $y :=$ the half point between y_1 and y_2;
    ask superset query r_y;   (i.e., '$\Phi_{cir}(r_y) \supseteq \Phi_{cir}(r)$?')
    if $\Phi_{cir}(r_y) \supseteq \Phi_{cir}(r)$ **then** $y_1 := y$ **else** $y_2 := y$
  end-while
  output(y_1)
end. □

Now we are ready to prove the following lower bound.

Theorem 4.6. $\mathrm{P}^{\mathrm{NP}_1^{()}}$ is a lower bound of the power of (Sub,Sup;)-learning systems for *CIR* on R_{cir}^{rep}.

Remark. That is, for every $\mathrm{P}^{\mathrm{NP}_1^{()}}$-machine M, there exist a polynomial time (Sub,Sup;)-learner S for *CIR* and a polynomial length bound b' such that S with a b'-bounded teacher simulates M. Obviously, the same lower bound holds for any query-answer type stronger than (Sub,Sup;), e.g., (Mem;Sub[min],Sup[min]).

Proof. As explained above, it suffices to show the way to get two elements $y_1, y_2 \in \{0,1\}^{\le m}$ of a target set by using superset min-counterexample-queries. The following learner S achieves this task.

(Let $\langle n, m\rangle$ and $\Phi_{cir}(r)$ be the given input and target set.)

(1) S generates r_1 such that $\Phi_{cir}(r_1) = \emptyset$ and asks a superset query r_1 (i.e., '$\emptyset \supseteq \Phi_{cir}(r)$?'). If any counterexample exists, then a teacher gives the lexicographically smallest counterexample y_1, which clearly satisfies $y_1 \in \{0,1\}^{\le m} \cap \Phi_{cir}(r)$. Otherwise, S can output r_1 as a m-representation for r.

(2) S generates r_2 such that $\Phi_{cir}(r_2) = \mathit{Rep}(\{y_1\})$, asks a superset query r_2 (i.e., '$\mathit{Rep}(\{y_1\}) \supseteq \Phi_{cir}(r)$?'). The rest is the same as (1). □

Now we use the lower bound result to relate the learnability of R_{cir}^{rep} to the complexity of representation finding problems for sets in P/poly. Here the following theorem, which corresponds to Theorem 3.4, states one general relationship between them.

Theorem 4.7. Let $C = (R, \Phi)$ be any one of the following representation classes:- *DFA*, *NFA*, *CFG*, and *CIR*. For any query-answer type α, let $\mathcal{M}^{()}$ be a query machine type that is a lower bound of the power of α-learning systems for C. Then if C is not α-learnable, then some $B \in \mathrm{P/poly}[C]$ exists such that no $\mathcal{M}^{()}$-machine solves FIND-REP($B|C$). (In order to simplify our argument, we assume that the machine type $\mathcal{M}^{()}$ is one of those defined in Preliminaries that use $\mathrm{P}^{()}$-machines as base machines.)

Remark. $C = (R, \Phi)$ can be any representation class that satisfies the following conditions with some constant d_{a}, some polynomials p_{a} and p_{b}, and some polynomial time deterministic transducers M_{a} and M_{c}.

(a) $\forall r_1, r_2 \in R\ [\ \Phi(r_1) \subseteq \Sigma^{\le l} \wedge \Phi(r_2) \subseteq \Sigma^{\ge l}$
$$\to\ \exists r_3 \in R\ [\ |r_3| \le |r_1| 2^{d_{\mathrm{a}} l} + p_{\mathrm{a}}(|r_2|)\ \wedge\ \Phi(r_3) = \Phi(r_1) \cup \Phi(r_2)\ ]\ ].$$
Furthermore, $M_{\mathrm{a}}(r_1, r_2)$ yields r_3.

(b) $\forall r_1 \in R,\ \forall l_1, l_2 \ge 0,\ \exists r_2 \in R\ [\ |r_2| \le p_{\mathrm{b}}(|r_1| + l_1 + l_2)\ \wedge\ \Phi(r_2) = 0^{l_1}\#\Phi(r_1)^{\le l_2}\ ]$.

(c) For any given $r_1 \in R$ that is a $l+m+1$-representation of $X \cup 0^l\#Y$ for some $X \subseteq \Sigma^{\le l}$ and Y, $M_{\mathrm{c}}(r_1)$ outputs some m-representation r_2 of Y. (That is, $\Phi(r_1)^{\le l+m+1} = X \cup 0^l\#Y^{\le m}$ and $\Phi(r_2)^{\le m} = Y^{\le m}$.)

It is not hard to show that *DFA*, *NFA*, *CFG*, and *CIR* have the properties.

Proof. We assume that FIND-REP(B) is $\mathcal{M}^{()}$-solvable for every $B \in \mathrm{P/poly}[C]$, and show that C is α-learnable. (Throughout this proof, by FIND-REP(B) we mean FIND-REP($B|C$).)

First we show the following fact.

Fact. Let p be any polynomial. There exist some finite set X and some $\mathcal{M}^{()}$-machine M such that for every finite set $Z \in \mathrm{P}/p(m)[C]$, M solves FIND-REP($X \cup Z$).

Proof of Fact. We assume that the fact does not hold, and construct some $\tilde{Z} \in \mathrm{P/poly}[C]$ whose representation finding problem is not $\mathcal{M}^{()}$-solvable, which contradicts our assumption.

Thus, suppose that for every finite set X and every $\mathcal{M}^{()}$-machine M, there is some finite set $Z \in \mathrm{P}/p(m)[C]$ such that M cannot solve FIND-REP($X \cup Z$). From this assumption, we construct, by the following standard stage construction, a sequence $Z_1, Z_2, \ldots$ in $\mathrm{P}/p(m)[C]$ such that no $\mathcal{M}^{()}$-machine can solve FIND-REP($\cup_{i\ge1} Z_i$).

First we observe that our assumption indeed implies the following: for every finite set X, every $\mathcal{M}^{()}$-machine M, and every $s \geq 0$, there is some finite set $Z \in \mathrm{P}/p(m)[C]$ such that $Z \subseteq \Sigma^{\geq s}$ and M cannot solve FIND-REP$(X \cup Z)$. Because, if otherwise, there are some finite set X, $\mathcal{M}^{()}$-machine M, and constant s such that M solves FIND-REP$(X \cup Z)$ for every finite $Z \subseteq \Sigma^{\geq s}$; but from this, we can define some $\mathcal{M}^{()}$-machine M' that solves FIND-REP$(X \cup Z)$ for *every* finite set Z, contradicting our first assumption. Here M' is executed as follows: Since X is a finite set, we may assume that M' knows X. Thus, M' can get a description r_1 of $X \cup Z^{\geq s}$ by executing M relative to $X \cup Z^{\geq s}$; then it gets a description r_2 of $X \cup Z^{\leq s}$ by just asking exhaustive queries; and finally computes a description of $\Phi(r_1) \cup \Phi(r_2)$ by using M_{a} in condition (a).

We construct $Z_1, Z_2, \ldots$ by stages. Let $\{M_i\}_{i \geq 1}$ be an enumeration of $\mathcal{M}^{()}$-machines, and let s_i and t_i be the length of the shortest and longest element of Z_i respectively. Consider any stage $k \geq 1$ of the construction, and let $X = \cup_{1 \leq i \leq k-1} Z_i$. Then from the assumption, we can find some finite set Z_k such that M_k does not solve FIND-REP$(X \cup Z_k)$. Furthermore, from the observation above, we may assume that s_k, the length of the shortest element of Z_k, is large enough so that $\log\log s_k \geq t_{k-1}$ and $M_{k-1}(0^{t_k})$ asks no query of length $\geq s_k$. Then it is almost immediate from this construction that no $\mathcal{M}^{()}$-machine solves FIND-REP$(\cup_{i \geq 1} Z_i)$. We define $\tilde{Z} = \cup_{i \geq 1} Z_i)$.

Now, for leading the contradiction, it suffices to show that $\tilde{Z} \in \mathrm{P/poly}[C]$. Consider any $m \geq 0$. Then there is some k such that $t_k \leq \log\log m$, and $Z^{\leq m} = \cup_{1 \leq i \leq k} Z_i \cup Z_{k+1}^{\leq m}$. Hence, from condition (a), we have some $r \in R$ such that $Z^{\leq m} = \Phi(r)^{\leq m}$, and

$$\begin{aligned} |r| &\leq (\cdots((|r_1|2^{d_{\mathrm{a}} t_1} + p_{\mathrm{a}}(|r_2|))2^{d_{\mathrm{a}} t_2} + p_{\mathrm{a}}(|r_3|))\cdots)2^{d_{\mathrm{a}} t_k} + p_{\mathrm{a}}(|r'|) \\ &\leq 2^{k d_{\mathrm{a}} t_k}\left(\textstyle\sum_{i=1}^{k} p_{\mathrm{a}}(|r_i|)\right) + p_{\mathrm{a}}(|r'|), \end{aligned}$$

where r_i and r' are the shortest representations for Z_i and $Z_{k+1}^{\leq m}$ respectively. Note here that each $Z_i \in \mathrm{P}/p(m)[C]$; hence, $|r_i| \leq p(t_i)$ and $|r'| \leq p(m)$. Thus, $|r| \leq (2^{k d_{\mathrm{a}} t_k} k + 1) p_{\mathrm{a}}(p(m))$, which is less than $q(m)$ for some polynomial q. (Recall that $k \leq t_k \leq \log\log m$.) Therefore, $Z^{\leq m}$ has a representation of length $\leq q(m)$. That is, $Z \in \mathrm{P/poly}[C]$. $\square$ (Fact)

Apply this fact for the polynomial $p_{\mathrm{b}}(2m)$. Then we have some finite set X and some $\mathcal{M}^{()}$-machine M_1 such that M solves FIND-REP$(X \cup Z)$ for every finite set $Z \in \mathrm{P}/p_{\mathrm{b}}(2m)[C]$. Let l_X be the length of the longest string in X. We use M_1 to define M_2 that executes as follows:

The execution of M_2 on input $\langle 0^n, 0^m\rangle$ relative to Y:

begin

$l := l_X + m + n;$

$r_1 := M_1^{X\cup Z}(0^{l+m+1})$ (where $Z = 0^l\#Y^{\le m}$);

$r_2 := M_c(r_1);$

output r_2

end.

Clearly, M_2 is a $\mathcal{M}^{()}$-machine. In the following, we show that M_2 learns C correctly. That is, for any $r \in R$, $n \ge |r|$, and $m \ge 0$, consider the execution of M_2 on $\langle 0^n, 0^m\rangle$ relative to $\Phi(r)$, and show that M_2 yields some m-representation for r. In the execution, M_1 is executed on input 0^{l+m+1} relative to $X \cup Z$, where $Z = 0^l\#\Phi(r)^{\le m}$. Now we estimate the descriptive complexity of Z. From (b), we have some (exact) representation r_Z of Z such that $|r_Z| \le p_b(|r| + l + m) \le p_b(l + m + n) \le p_b(2l)$. Hence, for any $m' > l$, $Z^{\le m'}$ has a m'-representation, namely, r_Z, of length $\le p_b(2m')$. (Notice that r_Z is a m'-representation for any m'.) On the other hand, if $m' \le l$, then $Z^{\le m'}$ is empty, and we may assume that empty set has a representation of length $\le p_b(0)$. Thus, Z is a finite set in $\mathrm{P}/p_b(2m)[C]$. Then it follows from our assumption on M_1 that M_1 solves FIND-REP($X \cup Z$). Hence, r_1 $(= M_1^{X\cup Z}(0^{l+m+1}))$ is a $l + m + 1$-representation for $X \cup Z$; i.e., $\Phi(r_1)^{\le l+m+1} = (X \cup Z)^{\le l+m+1}$. Note that $X \cup Z$ $(= X \cup 0^l\#\Phi(r)^{\le m}) \subseteq \Sigma^{\le l+m+1}$; hence, $\Phi(r_1)^{\le l+m+1} = X \cup 0^l\#\Phi(r)^{\le m}$. Now from (c), r_2 $(= M_c(r_1))$ satisfies that $\Phi(r_2)^{\le m} = \Phi(r)^{\le m}$; that is, r_2 is a m-representation for r.

Finally, since $\mathcal{M}^{()}$ is a lower bound of the power of α-learning systems for C, the computation by M_2 can be simulated by some α-learning system. Therefore, C is α-learnable. $\square$

As a corollary to Theorem 4.6 and Theorem 4.7, we can prove the converse of Corollary 3.5 (2) for some representation *subclass* of *CIR*.

Corollary 4.8. If R_{cir}^{rep} is not (Sub,Sup;)-learnable, then some $B \in \mathrm{P/poly}$ exists such that no $\mathrm{P}^{\mathrm{NP}_1^{()}}$-machine solves FIND-REP(B).

Remark. The corollary holds for any query-answer type stronger than (Sub,Sup;).

In this section, we have discussed the problem of learning (some subclass of) *CIR*. Some of our results hold for the other representation classes such as *NFA* and *CFG*. As a final remark of this section, we explain them considering *NFA*. (The same results hold for *CFG*.)

Let R_{nfa}^{rep} be the set of representations in R_{nfa} denoting repetitive sets.

Theorem 4.9. $\mathrm{P}^{\mathrm{NP}_{\mathrm{m}}^{()}}$ is a lower bound of the power of (Sup;)-learning systems for *NFA* on R_{nfa}^{rep}.

Note that this theorem corresponds to Theorem 4.6, and that it is provable in the same way by using the following lemma and the one corresponding to Proposition 4.5.

Lemma 4.10. For any $\mathrm{NP}_{\mathrm{m}}^{()}$-acceptor M, there exist a polynomial time (Sup;)-learner S for *NFA* and a polynomial length bound b such that

$$\begin{array}{l} \forall T:\ b\text{-bounded (Sup;)-teacher for } \mathit{NFA}, \\ \forall r \in R_{cir}^{\mathrm{rep}},\ \forall x \in \Sigma^*,\ \forall y_1, y_2 \in \{0,1\}^* \cap \Phi_{cir}(r) \\ [\ M^{\Phi_{cir}(r)} \text{ accepts } x \ \leftrightarrow\ \langle S, T(r)\rangle(x, y_1, y_2) \text{ accepts } x\ ]. \end{array}$$

Proof Outline. The proof is essentially the same as the one for Lemma 4.3. Here we explain only the point that is particular in this proof.

Let M be any $\mathrm{NP}_{\mathrm{m}}^{()}$-machine that is to be simulated. Recall that the key idea is that one can simulate M by using superset queries; let us see how the simulation works in this context. We use the same symbols as the proof of Lemma 4.3: M is specified by f_M and p_M. $X \subseteq \Sigma^*$ is any oracle set, where $X = \mathit{Rep}(Z)$ for some $Z \subseteq \{0,1\}^*$. z_0 and z_1 are any elements of Z. And x is any input string for M.

In the previous proof, the superset query '$\overline{Q(x, z_0, z_1)} \supseteq X$?' is used to determine whether M^X accepts x. Note that in order to ask this superset query, a learner has to present a representation of $\overline{Q(x, z_0, z_1)}$. Thus, in our case, $\overline{Q(x, z_0, z_1)}$ must have a nfa representation that is polynomial time computable from x, z_0, and z_1. However, it is impossible in general because the size of nfa for $\overline{Q(x, z_0, z_1)}$ is not polynomially bounded. Here we introduce a new set $R(x, z_0, z_1)$, which is similar to $\overline{Q(x, z_0, z_1)}$, such that some polynomial time algorithm computes an nfa representation for $\overline{R(x, z_0, z_1)}$ from x, z_0, and z_1.

Define $R(x, z_0, z_1)$ to be $\{\ id_1\#\cdots\#id_k\#f_M(x, w) : w \in \{0,1\}^{p_M(|x|)}\ \}$, where id_1, ..., id_k is the sequence of M's instantaneous descriptions on nondeterministic computation path w for input x. We assume that each instantaneous description is encoded in $\mathit{Rep}(Z)$ by using, e.g., $z_0\#z_1$ and $z_1\#z_0$ as alphabet symbols. Thus, $id\#f$ $(= id_1\#\cdots\#id_k\#f_M(x, w))$ is in $\mathit{Rep}(Z)$ if and only if $f_M(x, w) \in X$ $(= \mathit{Rep}(Z))$. Hence, by the same argument as before, we can prove that M^X accepts x if and only if $\overline{R(x, z_0, z_1)} \not\supseteq X$.

Now consider the nfa acceptor of $\overline{R(x, z_0, z_1)}$. Note that a given $v \in \Sigma^*$ is not in $R(x, z_0, z_1)$ if and only if one of the following three conditions holds: (i) v is not of the form $id\#f$, (ii) $v = id\#f$ but id does not denote any consistent computation, or (iii) $v = id\#f$ and id denotes some computation but f is not queried on the computation. It is not hard to construct an nfa of polynomial size that accepts v if one of (i) $-$ (iii) holds. Thus, some polynomial time learner ask superset query '$\overline{R(x, z_0, z_1)} \subseteq X$?'. The detail is left to the reader. $\square$

Notice that there is no polynomial bound on the size of nfa that accepts the above $R(x, z_0, z_1)$. Thus, the lemma corresponding to Lemma 4.4 is left open.

5. Concluding Remarks

We discuss some of the related open problems. In this paper, some close relationship has been shown between polynomial time query learnability and difficulty of representation finding problems. Thus, one obvious question is complexity of representation finding problems. To be more precise, take *CIR* for example of representation classes. Then our question is to prove/disprove the following:

$(*)$ Some $A \in$ P/poly exists such that no $\mathrm{P}^{\mathrm{NP}_1^{()}}$-machine solves FIND-REP(A).

Gavaldà and Watanabe [6] studied complexity of small descriptions in general. As one of their results, they essentially proved the following lower bound: There exists some $A_0 \in$ P/poly such that no NPSV$^{()}$-machine solves FIND-REP(A_0), where a NPSV$^{()}$-machine is a polynomial time nondeterministic "single valued" transducer.

However, more recent study indicates that improving this lower bound to prove $(*)$ is difficult. First note the following straightforward observation on the complexity of FIND-REP(A).

Proposition 5.1. For every $L \in$ P/poly, FIND-REP(L) is solved by some P$^{()}$-machine by using PREF-REP(L) as an oracle, and PREF-REP(L) is in $\Sigma_{2,1}^{\mathrm{P},L}$.

Thus, $(*)$ implies $\Delta_{2,1}^{\mathrm{P},A} \subsetneq \Sigma_{2,1}^{\mathrm{P},A}$ (because PREF-REP(A) $\in \Sigma_{2,1}^{\mathrm{P},A} - \Delta_{2,1}^{\mathrm{P},A}$). That is, $(*)$ implies separation between Δ_2^{P} and Σ_2^{P} — separation at the second level in the polynomial hierarchy — by some P/poly oracle. For any higher level (i.e., the kth level for any $k \geq 3$), it is well-known (see, e.g., [Sch85]) that separation by some P/poly oracle implies the real separation. Recently, Gavaldà [5] proved that this is also true for the second level. That is, $(*)$ implies $\Delta_2^{\mathrm{P}} \subsetneq \Sigma_2^{\mathrm{P}}$. Thus, our problem is harder than the P $\neq$ NP problem. Therefore, for a more reasonable question, we ask whether $\Delta_2^{\mathrm{P}} \subsetneq \Sigma_2^{\mathrm{P}}$ (i.e., $\Delta_{2,1}^{\mathrm{P},A} \subsetneq \Sigma_{2,1}^{\mathrm{P},A}$ for some $A \in \mathrm{P/poly}$) implies $(*)$.

Acknowledgments

The authors thank Prof. Balcázarand Prof. Díaz for their comments on early version of this paper, and anonymous referees for pointing out them some technical errors through their careful reading.

References

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

[2] D. Angluin, "Negative results for equivalence queries". Machine Learning 5 (1990), 121-150.

[3] D. Angluin and M. Kharitonov, When won't membership queries help?, in "Proc. 23rd Annual ACM Sympos. on Theory of Comput." ACM, New York (1991), 444-454.

[4] J. Balcázar, J. Díaz, and J. Gabarró, "Structural Complexity I", EATCS Monographs on Theoretical Computer Science, Springer-Verlag (1988).

[5] R. Gavaldà, Bounding the complexity of advice functions, in "Proc. 7th Structure in Complexity Theory Conference", IEEE (1992), 249-254.

[6] R. Gavaldà and O. Watanabe, On the computational complexity of small descriptions, in "Proc. 6th Structure in Complexity Theory Conference", IEEE (1991), 89-101.

[7] O. Goldreich, S. Goldwasser, and S. Micali, How to construct random functions, J. ACM 33 (1986), 792-807.

[8] R. Karp and R. Lipton, Some connections between nonuniform and uniform complexity classes, in "Proc. 12th Annual ACM Sympos. on Theory of Comput.", ACM, New York (1980), 302-309.

[9] M. Kearns and L. Valiant, Cryptographic limitations on learning Boolean formulae and finite automata, in "Proc. 21st Annual ACM Sympos. on Theory of Comput.", ACM, New York (1989), 433-444.

[10] L. Pitt and M. Warmuth, Prediction preserving reducibility, J. Comput. Syst. Sci. 41 (1990), 430-467.

[11] U. Schöning, "Complexity and Structure", Lecture Notes in Computer Science 211, Springer-Verlag (1985).

[12] L. Stockmeyer, The polynomial-time hierarchy, Theoret. Comput. Sci. 3 (1977), 1-22.

[13] L. Valiant, A theory of the learnable, C. ACM 27 (1984), 1134-1142.

[14] M. Warmuth, Towards representation independence in PAC-learning, in "Proc. AII-89 Workshop on Analogical and Inductive Inference", Lecture Notes in AI 397, Springer-Verlag (1989), 78-103.

[15] O. Watanabe, A formal study of learning via queries, in "Proc. 17th International Colloquium on Automata, Languages and Programming", Lecture Notes in Computer Science 443, Springer-Verlag (1990), 137-152.

[16] O. Watanabe, A framework for polynomial time query learnability, in this issue.