Chapter 6

Separating Teaching Assistant Classes

In the previous chapter, we introduced various assistant classes and used these classes to give finer upper bounds on learning some algebraic representation classes. As mentioned earlier, the different assistant classes that we have defined are in exact analogy to standard complexity classes. Furthermore, the previous chapter indicates that some of these teaching assistant classes have some correspondence with exact learnability in Angluin's model. Motivated by the possible finer classification of representation classes, and again in analogy with complexity classes, in this chapter we define and study some more teaching assistant classes. Particularly, we define assistant classes corresponding to the classes UP $\cap$ co-UP, UP and NP $\cap$ co-NP. Using some subclasses of the representation class SYM, we show some separations that are possible among various teaching assistant classes that we consider.

Before going into the proofs, we summarize the containments and separations that we show among various teaching assistant classes in the following figure. For the completeness sake, we also compare various types of queries in Angluin's model. In the last chapter we have shown that the membership queries in Angluin's model are completely characterized by P-assistants in the teaching assistant model. It is shown in [Ang88] that the class k-CNF is FP-learnable with only equivalence queries in Angluin's model. It is also shown there that the representation class monotone-

DNF (class of monotone boolean functions represented by monotone DNF formulas) is FP-learnable with equivalence and membership queries. In [AHK93], it is shown that the class of monotone Read-once formulas (class of monotone boolean functions representable by monotone formulas in which all the literals appear exactly once) are FP-learnable with only membership queries.

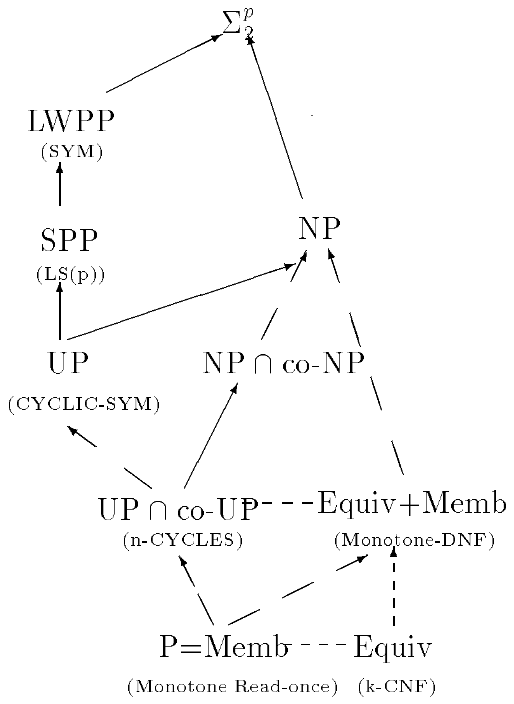

Figure 1. Inclusions and separations of different teaching assistant classes[1].

In the next section we define three more teaching assistant classes analogous to complexity classes NP ∩ co-NP, UP ∩ co-UP and UP. We also prove machine characterizations of some of these classes. These characterizations are used to show some more lower bounds in the teaching assistant model.

[1]In the 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. For each teaching assistant class $\mathcal{C}$ in the picture, a representation class that is FP-learnable with $\mathcal{C}$-assistant is given in brackets under the assistant class. Formal definitions of these classes are given later.

6.1 More Assistant Classes

Let us first define the assistant classes of interest to us in this chapter.

Definition 6.1.1 Let $\mathcal{P}$ be a representation class and $L(\mathcal{P})$ be a teaching assistant.

1. $L(\mathcal{P})$ 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 $\langle n, l, x, c\rangle \notin L(\mathcal{P})$.

2. $L(\mathcal{P})$ is said to be in the class $\mathrm{UP}(\mathcal{P})$ 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 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.

3. $L(\mathcal{P})$ 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 $\langle n, l, x, c\rangle \notin L(\mathcal{P})$.

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

Remark. Since we are interested in the query complexity (in an information-theoretic sense) of teaching assistants, it is some times useful to define relativized teaching assistant classes by allowing Turing machines accepting teaching assistants to have oracle access to a language, say $A \subseteq \Sigma^*$. Notice that this language aids the machine only in computation (we term such an oracle a *computational oracle*) and not in retrieving information about the target concept. Let $\mathcal{P}$ be a representation class. Then the assistant class $\mathcal{C}(\mathcal{P})$ relative to a computational oracle $A \subseteq \Sigma^*$ is denoted by $\mathcal{C}(\mathcal{P})^A$.

The following refinement of Proposition 5.4.3 from the previous chapter is direct from the definitions.

Proposition 6.1.2 *For any representation class $\mathcal{P}$*

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

6.1.1 Learning with $\mathrm{NP} \cap \text{co-NP}$ and $\mathrm{UP} \cap \text{co-UP}$ assistants

Machine characterizations of learners which use assistants from NP$\cap$co-NP and UP$\cap$ co-UP assistant classes will be handy in many of our proofs, especially for proving lower bounds. We next show these characterizations. For this purpose, we extend the deterministic learners in Angluin's model to nondeterministic learners. We consider two types of nondeterministic learners; NPSV learners and UPSV learners. Let $\mathcal{P}$ be a representation class. A NPSV (UPSV) learner is a nondeterministic oracle Turing machine transducer which on input $\langle 0^n, 0^l \rangle$ and $c \in \mathcal{P}$ as target concept, has to output a representation $r \in R$ such that $\mu_n(r) = c$ (if $size(c) \leq l$) for the concept c, on at least one (respectively, *exactly* one) of its paths. Moreover, the learner should output the *same* representation on all the paths on which it is outputting. Also, the running time of the transducer must be bounded by a polynomial. As in the case of teaching assistant classes, we also allow these learners to have access to a computational oracle whenever necessary.

In the next theorem, we show machine characterizations of the above-defined assistant classes. The proof of the theorem is an adaptation of the proof that a function $f \in \mathrm{FP}^{\mathrm{NP}\cap\text{co-NP}}$ if and only if f is NPSV computable (implicitly in [BLS84]).

Theorem 6.1.3 *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 very similarly.

Let $\mathcal{P} = \langle R, \{\mu_n\}_{n\geq 1}, t\rangle$ be any representation class and A be a computational oracle. Let β be an FP-learner for $\mathcal{P}$ using a 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 $size(c) \leq l$. 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 $\mathcal{P} = \langle R, \{\mu_n\}_{n\geq 1}, t\rangle$ be any representation class that is NPSV-learnable using 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 exactly $q(n+l)$ 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 using 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 $\mathcal{P}$ using teaching assistant $L(\mathcal{P})$. Let $c \in \mathcal{P}_n$ be the target concept and l such that $size(c) \leq l$.

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 $y \leftarrow y\overline{b}$;
 end-if
 $i \leftarrow i + 1$;
end-while
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∩co-NP-assistants and UP∩co-UP-assistants? We show in the next theorem that information-theoretically they have the same power. More precisely, we show that UP ∩ co-UP-assistants with computational oracles in NP are as powerful as NP ∩ co-NP-assistants. Thus, separating these assistant classes in the polynomial-time learnability sense is as hard as separating P and NP.

Computational difficulty of separating UP∩co-UP and NP∩co-NP assistants

Here we prove the following theorem.

Theorem 6.1.4 *Let $\mathcal{P}$ be any representation class. If $\mathcal{P}$ is* FP*-learnable with an* (NP ∩ co-NP)*-assistant, then there is language $B \in$* NP *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 using NP ∩ co-NP-assistants and UP ∩ co-UP-assistants proved in Theorem 6.1.3. Let $\mathcal{P} = \langle R, \{\mu_n\}_{n\geq 1}, t\rangle$ be any representation class which is FP-learnable using NP ∩ co-NP-assistant. Then by Theorem 6.1.3, there is an NPSV-learner α, which learns $\mathcal{P}$ using only membership queries. 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, outputs a string on $\rho\}$. We first show that B is in NP. Consider a nondeterministic oracle machine N which on input $\langle 0^n, 0^l, x, y\rangle$ first guesses 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. 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 computes the answer to this query using the representation y (part of input) of $\mu_n(y)$. Hence $B \in$ NP.

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 6.1.3, it follows that $\mathcal{P}$ is FP-learnable using a $(\mathrm{UP} \cap \text{co-UP})^B$-assistant with the help of an NP oracle. ∎

The next corollary is immediate.

Corollary 6.1.5 *If there exists a representation class $\mathcal{P}$ that is* FP-*learnable with an* NP ∩ co-NP-*assistant, and not* FP-*learnable with a* UP ∩ co-UP-*assistant, then* P ≠ NP.

6.2 Learning Subclasses of SYM

In this section we show more results on the learnability of algebraic representation classes n-CYCLES and CYCLIC-SYM. These are subclasses of SYM $= \langle R, \{\mu_n\}_{n\geq 1}, t\rangle$. By restricting elements in R to satisfy some pre-specified property, we define these subclass of SYM as follows. $\mathcal{P}' = \langle R', \{\mu'_n\}_{n\geq 1}, t'\rangle$ be a subclass of SYM. $\mathcal{P}'$ is the class CYCLIC-SYM if R' consists of those sets of permutations which generate a *cyclic* group. $\mathcal{P}'$ is the class n-CYCLES if R' consists of a single *n-cycle* (please recall the definition of an *n-cycle* from the previous chapter). It is clear that n-CYCLES⊆CYCLIC-SYM⊆SYM. Since SYM is honest, it follows that all these classes are honest.

Next we show upper and lower bounds on learning these subclasses in teaching assistant model. These results are relatively easier to prove. The purpose of these results is to illustrate the separations that are possible among various teaching assistant classes that we have defined.

First we show upper and lower bounds on learning n-CYCLES.

Learning n-CYCLES

We show the following theorem.

Theorem 6.2.1 *The class n-CYCLES is not FP-learnable with P-assistants. It is FP-learnable with a $\mathrm{UP} \cap \mathrm{co\text{-}UP}$-assistant.*

Proof. Notice that FP-learning with membership query in Angluin's model is same as FP-learning with a P-assistant. Now, the proof of the first part of the theorem is already given as the proof of the Theorem 5.3.2 given in the previous chapter, namely the class SYM is not FP-learnable with membership queries.

For the proof of the second part, we need the following elementary fact about the number of generators of a group generated by an n-cycle. To be more precise, 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 $G = \langle\sigma\rangle$.

Now, 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'. From Theorem 6.1.3, it follows that n-CYCLES is FP-learnable with a $\mathrm{UP} \cap \mathrm{co\text{-}UP}$-assistant ∎

We now turn to the learnability of the class CYCLIC-SYM.

Learning CYCLIC-SYM

We show that while CYCLIC-SYM is not learnable with NP $\cap$ co-NP-assistants, it is FP-learnable with a UP-assistant. We first prove the lower bound.

Theorem 6.2.2 *The class* CYCLIC-SYM *is not* FP-*learnable with* NP $\cap$ co-NP-*assistants.*

Proof. To show this lower bound, notice that it suffices to show that CYCLIC-SYM is not NPSV-learnable with membership queries. Suppose α is an NPSV-learner for CYCLIC-SYM. Let s be a polynomial such that $s(n)$ upper bounds the number of membership queries asked by $\alpha(0^n)$. Choose a prime p such that $(p-2)! > s(p)$. 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 $(q_1, q_2 \ldots, q_{s(p)})$ 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 $(p-2)! > s(p)$, Proposition 5.3.1 implies that there is a p-cycle g such that $\{q_i \mid 1 \leq i \leq s(p)\} - \{e\} \cap \langle g \rangle = \emptyset$. Now consider the same computation path ρ for $\alpha(0^p)$ for target group $\langle g \rangle$. Notice that the answers to queries $(q_1, q_2 \ldots, q_{s(p)}, e)$ 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. $\blacksquare$

Now we give an upper bound on learning CYCLIC-SYM. For the proof of this theorem we require the following simple group-theoretic results. See the appendix for a proof.

Proposition 6.2.3 *Let G be a cyclic subgroup of S_n.*

1. *If G is of order p^k, for prime p, then $p^k \leq n$.*
2. *Let $|G|$ has the 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 $k_i \leq e_i$.*

Theorem 6.2.4 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 witnesses L(CYCLIC-SYM) in UP(CYCLIC − SYM).

ACCEPT-L(CYCLIC-SYM)$(0^n, p, k, x, G)$

```
 1  K ← p^k − p^(k−1);
 2  if p^k > n or p not a prime
 3     then reject
 4  end-if
 5  Guess K distinct strings x_1, ..., x_K in lexicographical order
 6  for i = 1 to K
 7  do if x_i does not encode a permutation of S_n
 8          then reject
 9          else  Compute the order o_i of the element encoded by x_i
10                if o_i ≠ p^k
11                   then reject
12                end-if
13       end-if
14  end-for
15  if ∃y such that x_1 = xy
16     then accept
17     else  reject
18  end-if
```

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. *line* 2 can be obviously done in time polynomial in n. Since $p^k \leq n$ the nondeterministic step in *line*-5 can also be done in time polynomial in n. Computing the order of $g \in S_n$, in *line*-9 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 decomposition. Now, from

Proposition 6.2.3, 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 CyclicLearner for CYCLIC-SYM, which uses the assistant L(CYCLIC-SYM). Let G be the target concept. We first give an intuitive idea of how CyclicLearner it works. Let $p_1^{e_1} \dots p_l^{e_l}$ 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. CyclicLearner exploits precisely this property: using the 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.

CyclicLearner(0^n)

```
σ ← e;
for primes p ≤ n
do Compute the largest index d such that p^d ≤ n;
    while ⟨0^n, p, d, λ, G⟩ ∉ L(CYCLIC-SYM)
    do d ← d − 1;
    end-while
    (* Using L(CYCLIC-SYM) prefix search for g_p of order p^d *)
    z ← λ;
    while ⟨0^n, p, d, z0, G⟩ or ⟨0^n, p, d, z1, G⟩ ∈ L(CYCLIC-SYM)
    do if ⟨0^n, p, d, z0, G⟩ ∈ L(CYCLIC-SYM)
           then z ← z0;
           else z ← z1;
    end-while
    g_p ← z;
    σ ← σg_p;
end-for
output σ.
```

6.3 Summary

The main focus of this chapter was on separating some of the teaching assistant classes that we defined in this and the previous chapters of the thesis. We introduced subclasses of the representation class SYM for showing some of the separations. (The separations shown are depicted in Figure 1.) The main purpose of defining these subclasses was to illustrate the possibility of separations among the teaching assistant classes. However, it will be interesting to see whether there are other representation classes studied in the literature which witness these separations.