Reports of Machine Learning and Inference Laboratory, MLI 91-11, George Mason University, December 1991.

THE AQ FAMILY OF LEARNING PROGRAMS: A Review of Recent Developments and an Exemplary Application

R.S. Michalski, K. Kaufman and J. Wnek
Center for Artificial Intelligence
4400 University Dr.
Fairfax, VA 22030
George Mason University
{michalski,kaufman,wnek}@aic.gmu.edu

Abstract

This paper briefly reviews the family of AQ inductive learning programs, and describes recent new additions to the AQ family of inductive learning programs, such as AQ14-NT, AQ15-GA, AQ15-FCLS, AQ17-DCI, AQ17-HCI and AQ17-1. The AQ family includes a long series of programs based on the AQ general covering algorithm. The first members of the family were developed in the early seventies. The most important new ideas in the development of the series include powerful methods for data-driven (AQ17-DCI) and hypothesis-driven (AQ17-HCI) constructive induction, a method for handling noise in the input data (AQ14-NT), an integration of the AQ method with a genetic algorithm (AQ15-GA), and a method that combines the logic-based approach with some aspects of neural net type learning (AQ15-FCLS). The presented programs are illustrated by an application to the so-called MONKs' problems, and their performance is compared with several other learning programs.

1. INTRODUCTION

There have been a number of learning methods and approaches developed for inductive concept learning from examples. These methods can be divided into symbolic and subsymbolic, based on whether or not their representations of attributes and concepts consist of simple symbolic structures (e.g. logical descriptions) that can be related directly to the objects they represent. Typical symbolic representations include decision trees or rules, while subsymbolic methods include genetic algorithms and neural networks.

Given the many machine learning programs that have been developed for the purpose of concept learning, an important theoretical and practical problem is to determine the areas of best applicability of the various methods. One of the efforts in this direction was an international competition in which attendees at the 1991 European Summer School on Machine Learning were challenged with three problems developed by Thrun and Cheng (Thrun, Mitchell and Cheng, 1991). Given that the School was being held on the grounds of a converted priory, the problems became known as the "MONKs' Problems."

The AQ family of learning programs is a group of symbolic, generally rule-based programs that use the *Star* methodology for learning concepts from examples. This paper summarizes the history of these learning programs, including some programs based on, but not directly related to AQ. We also describe the results from applying recently developed AQ programs to the MONKs' problems. The results of the application of other programs to these problems and a more brief presentation of the results using the AQ programs were documented by Thrun, Mitchell and Cheng (1991). The main body of this paper concludes with descriptions of other comparative studies of machine learning programs.

2. THE AQ ALGORITHM, ITS HISTORY AND ITS IMPLEMENTATIONS

2.1 AQ Algorithm

All of the programs this paper focuses on use AQ as their basic induction algorithm. A brief description of the AQ algorithm follows:

1. Select a seed example from the set of training examples for a given decision class.

2. Using the *extend against* operator (Michalski, 1983), generate a set of alternative most general rules (a star) that cover the seed example, but do not cover any negative examples of the class.

3. Select the "best" rule from the star according to a multi-criteria rule quality function (called LEF - the lexicographical evaluation function), and remove the examples covered by this rule from the set of positive examples yet to be covered. Typical conditions in the LEF include maximizing the number of events of the decision class covered by the rule, minimizing or maximizing the number of conditions in the rule, and minimizing the total "cost" of the rule.

4. If this set of examples yet to be covered is not empty, select a new seed from this set and go to step 2. Otherwise, if another decision class still requires rules to be learned, return to step 1, and perform it for this other decision class.

The stars generated in step 2 above are obtained as follows:

1. The initial partial star is either the entire event space or, in the case of incremental learning (learning from examples and prior rules), the portion of the event space covered by the corresponding rule from the rule set to be enhanced.

2. If the partial star covers no negative examples of the concept, we are finished. Otherwise, repeat steps 3-6 until it no longer covers any negative examples.

3. Select a negative example covered by the partial star.

4. Generate the set of maximally general hypotheses in the event space that cover the seed example of the concept but not the selected negative example (extension against).

5. Intersect the output of the previous set with the current partial star to generate a new partial star.

6. Trim the partial star by only retaining the "best" *maxstar* rules in the star, where *maxstar* is a user-defined parameter.

The set of rules generated by AQ will approach, in polynomial time, the simplicity of the optimally simple rules that could be generated by exhaustive search (an NP-hard undertaking). Michalski proved that the upper bound on the size of a set of disjoint stars in a given example and event space is a lower bound on the number of conjunctive rules necessary to completely and consistently cover a concept's examples (Michalski, 1969b). Hence the difference between the size of a cover generated by AQ and the size of a disjoint set of stars in that example space is the maximum deviation from minimality of the cover.

2.2. A Brief History of AQ Family of Programs

The AQ algorithm was developed by Michalski (1969). AQVAL/1, also known as AQ7, is an implementation of AQ that can express its discovered rules in the variable-valued logical language VL_1 (Michalski, 1973b; Michalski and Larson, 1975). It learns these rules from a set of examples in the form of attribute-value tuples and chooses optimal rules based on user-defined criteria. Rules discovered by AQVAL/1 can be disjoint (i.e., no example in the event space can be covered by more than one rule), intersecting (i.e., events can be covered by multiple rules) or ordered (i.e., the rules have a priority order in which the *i*th rule will take effect only if the previous *i*-1 rules did not fire.)

AQ9 (Cuneo, 1975) uses the AQ methodology to optimize existing VL_1 rules according to some user-defined criterion. It can, for example, generate discriminant rules that will succinctly distinguish between classes of events from an input consisting of characteristic rules learned by AQ7. Rules learned in this manner will often be more useful than discriminant rules learned directly from examples. AQ9 can generate rules that are ordered, intersecting or disjoint.

UNICLASS (Stepp, 1979) uses the AQ algorithm to characterize a single class of examples. In this case, rule generation is not affected by other classes of events, instead an optimal covering is generated based solely on the user-defined criteria and the input positive examples.

The capability for incremental learning was added in AQ11 (Larson, 1976; Michalski, 1983) and later NEWGEM (Reinke, 1984; Mozetic, 1985). With incremental learning, previously generated (whether by machine learning or by manual entry) rules can be combined with new examples as input to the learning module, which will then refine the rules to account for examples that are not consistent with the initial knowledge. This process can be repeated over multiple iterations. These programs also incorporate ATEST, a module that allows rules to be tested for consistency and completeness and expresses the results of these tests in the form of a *confusion matrix* (Reinke, 1984).

AQ15 (Michalski et al, 1986) added *a-rules* and *l-rules*, respectively arithmetic and logic-based background knowledge, into the AQ framework. For example, in a domain which has numerical attributes *length* and *width*, an a-rule may be used to define an attribute *area* as the product of those two initial attributes. Or in a domain in which a person's age and marital status are input attributes, an l-rule can be employed to ensure that all instances of married individuals had ages above a certain level.

3. RECENT PROGRAMS AND EXTENSIONS TO THE AQ ALGORITHM

The AQ-based programs that were applied to the MONKs' problems are the result of efforts to further expand the learning capabilities of AQ in several new directions. The AQ17 programs are able to create new attributes better suited to the input example set (constructive induction). The DCI and HCI modules provide two different approaches to the task of constructive induction - a data-driven approach and a hypothesis-driven approach. AQ15-FCLS treats concepts as flexible entities with a basic core that almost defines them. AQ15-GA combines the abilities of AQ with those of genetic algorithms in order to optimize the discovered knowledge. And AQ14-NT is a version of NEWGEM designed to be able to learn concepts in noisy environments. In Section 3.1, we describe these programs in further detail, and discuss the tasks for which they are best suited. We conclude Section 3 with a listing and description of some of the other machine learning systems that rely heavily on the AQ algorithm.

3.1. Recent AQ Programs

3.1.1. AQ17-DCI (Data-driven constructive induction)

This program, developed by Bloedorn and Michalski, is based on the classical AQ algorithm (Section 2.1), but it includes an algorithm for constructive induction that generates a number of new attributes. The quality of any generated attribute is evaluated according to a special Quality Function for attributes. If the Quality Function exceeds a certain threshold value, then the attribute is selected. A brief description of the algorithm for data-driven constructive induction (Bloedorn and Michalski, 1991) is given below. The program works in two phases.

Phase 1.

1. Identify all numeric-valued attributes.

2. Repeat steps 3 through 5 for each possible combination of these attributes, starting with the pairs of attributes, and extending them if their quality was found acceptable according to the attribute Quality Function (QF).

3. Repeat steps 4 and 5 for each constructive induction operator. The current operators include addition, subtraction, multiplication, integer division and logical comparison of attributes (Bloedorn and Michalski, 1991).

4. Calculate the value resulting from the application of the given constructive induction operator to the given attribute pair.

5. Evaluate the discriminatory power of this newly constructed attribute using the attribute Quality Function, described by Bloedorn and Michalski (1991). If the QF for an attribute is above an assumed threshold, then the attribute is stored, else it is discarded.

6. Repeat steps 4 and 5 for each available global function operator that takes as argument an entire event (example), and calculate various global functions (properties) of it.

The program has a default list of global functions, but allows the user to modify the list to fit the problem at hand. The default list of functions include MAX (the maximum of the values of the numerical attributes in an event), MIN (the minimum value), AVE (the average value), MF (the most-frequent value), LF (least-frequent), and #VarEQ(x), which measures the number of variables (attributes) that take the value x in an example of a given class.

Phase 2.

1. Identify in the data all attributes that are binary-valued (i.e., boolean attributes).

2. Search for pairwise symmetry among the attributes and then for larger symmetry or approximate symmetry groups, based on the ideas described in (Michalski, 1969a; Jensen, 1975).

3. For each candidate symmetry group, create a new attribute that is the arithmetic sum of the attributes in the group.

4. Determine the quality function of the newly created attributes, and select the best attribute.

5. Enhance the dataset with values of this attribute, and induce new decision rules.

The method described above allows the system to express simply symmetric or partially symmetric Boolean functions and k-of-n functions, as well as more complex functions that depend on the presence of a certain number of attribute values in the data. Such functions are among the most difficult functions to express in terms of conventional logic operators.

3.1.2. AQ15-FCLS (Flexible concept learning)

This method, pioneered by Michalski, Zhang, Bergadano and Matwin, combines both symbolic and numeric representations in generating a concept description (Zhang and Michalski, 1991). The program is oriented toward learning flexible concepts, i.e., imprecise and context-dependent. To describe such concepts it creates two-tiered descriptions, which consist of a Basic Concept Representation (BCR) and an Inferential Concept Interpretation (ICI) to handle exceptions. In the program, the BCR is in the form of rules, and the ICI is in the form of a weighted evaluation function which sums up the contributions of individual conditions in a rule, and compares it with a THRESHOLD. The learning program learns both the rules and an appropriate value for the THRESHOLD.

Each rule of a concept description is learned in two steps, the first step is similar to the STAR algorithm in AQ that generates a general rule, and the second step optimizes the rule by specializing it and adjusting the accuracy threshold.

This flexible concept learning methodology has also been incorporated into the POSEIDON system (Bergadano et al, 1990).

3.1.3. AQ17-HCI (Hypothesis-driven constructive induction)

AQ17-HCI (Hypothesis-Driven Constructive Induction), created by Wnek and Michalski, represents a module employed in the AQ17 attribute-based multistrategy constructive learning system. This module implements a new iterative constructive induction capability in which new attributes are generated based on the analysis of the hypotheses produced in the previous iteration (Wnek and Michalski, 1991). Input to the HCI module consists of the example set and a set of rules, in this case generated by the AQ15 program (Michalski et al, 1986). The rules are then evaluated according to a rule quality criterion, and the rules that score the best for each decision class are combined into new attributes. These attributes are incorporated into the set of training examples, and the learning process is repeated. The process continues until a termination criterion is satisfied. The method is a special implementation of the idea of the "survival of the fittest," and therefore can be viewed as a combination of symbolic learning with a form of genetic algorithm-based learning.

A brief description of the HCI algorithm follows:

1. Induce rules for each decision class using a standard AQ algorithm (as implemented in AQ15) from a subset of the available training examples.

2. Identify variables from the original set that are not present in the rules, and classify them as irrelevant.

3. For each decision class, generate a new attribute that represents the disjunction of the highest quality rules (based on their coverage of the initial training examples).

4. Modify the training examples by adding the newly constructed attributes and removing the ones found to be irrelevant.

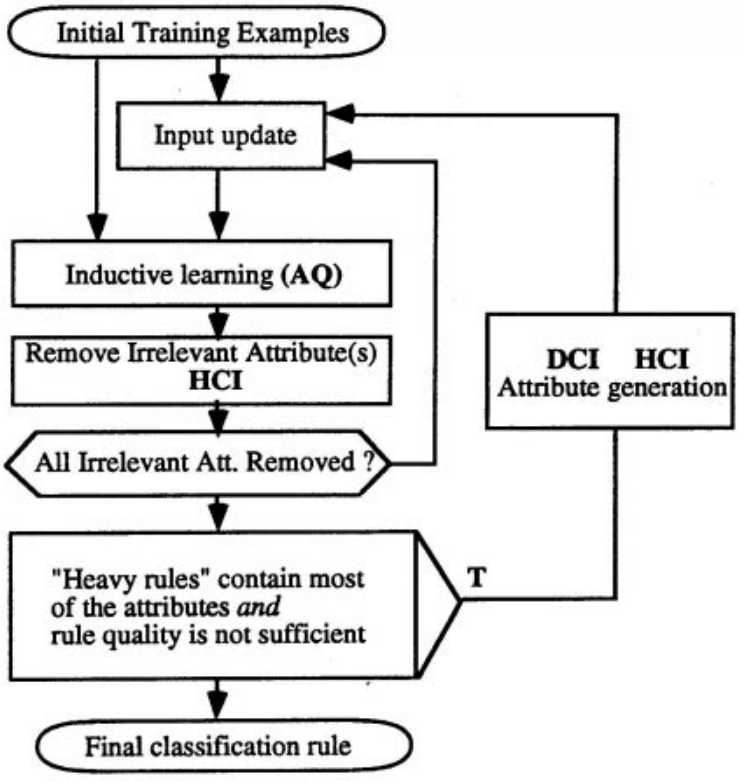
5. Induce rules from this modified training set.

6. Test these rules against the remainder of the training set. If the performance is not satisfactory, return to step 1. Otherwise, extend the initial complete set of training examples with the attributes from the obtained rules. Induce the final set of rules from this set of examples.

In these examples, the inductions performed in steps 1, 5 and 6 each use the learning algorithm implemented in the AQ15 program.

3.1.4. AQ17-1 (Multistrategy Constructive Induction)

AQ17-1 combines the work of Wnek, Bloedorn and Michalski, and combines the two constructive induction strategies implemented earlier in the AQ17-DCI and AQ17-HCI programs (described in Sections 3.1.1 and 3.1.3) into a new AQ-based program. The algorithm for employing these strategies is shown in Figure 1.



AQ - Basic inductive algorithm
DCI - Data-driven Constructive Induction
HCI - Hypothesis-driven Constructive Induction

Figure 1. AQ17 Control Strategy

AQ17-1 works as follows: The initial training example set is divided into a training sample (about 70%) and a tuning sample (about 30%). The training data is fed to a basic inductive learning module (AQ). Using the rules to evaluate the attributes against the tuning sample, the HCI method removes irrelevant attributes from the description of the training data. The process of rule generation and attribute removal is repeated until no more irrelevant attributes are found. If the

quality of the most recently generated rules is sufficient, then the final classification rule is generated from the complete training set, and the algorithm stops. Rule quality criterion is a measure of both accuracy on the tuning sample, and simplicity of the description.

If the rule quality criterion is not satisfied, the structure of the rules is checked. If "heavy rules," i.e. those rules that cover many examples and contain most of the remaining attributes, are present and rule quality is not satisfied, then the constructive attribute generation module is invoked. The newly supplemented attribute set is evaluated, and if rule quality is now sufficient, the final classification rule is produced from the complete training set.

Because of the nature of this algorithm, the rules generated by AQ17-1 will be identical to the rules generated by the individual HCI or DCI programs in cases in which the output from one or both of these programs is satisfactory. For this reason, the rules discovered by AQ17-1 for each of the three MONKs' problems (see Section 5) were the same as the rules generated by one of the component programs.

3.1.5. AQ14-NT (noise-tolerant learning from engineering data)

The program, developed by Pachowicz and Bala, implements an algorithm specially designed for learning from noisy engineering data (Pachowicz and Bala, 1991a and 1991b). The acquisition of concept descriptions (in the form of a set of decision rules) is performed in the following two phases:

- **Phase 1:**
  Concept-driven closed-loop filtration of training data, where a single loop of gradual noise removal from the training dataset is composed of the following four stages:

  1. Induce decision rules from a given dataset using the AQ14 (NEWGEM) inductive learning program.
  2. Truncate concept descriptions by removing "least significant" rules, that is, rules that cover only a small portion of the training data (this step is performed using the so-called TRUNC procedure).
  3. Create a new training dataset that includes only the training examples that are covered by the truncated concept descriptions.
  4. If the size of the dataset falls below an assumed percentage of the training data (that reflects an assumed error rate in the data), then go to Phase 2. Otherwise, return to step 1.

- **Phase 2:**
  Acquire concept descriptions from the improved training dataset using the AQ14 learning program.

A justification for Phase 1 is that the noise in the data is unlikely to constitute any strong patterns in the data, and therefore will require separate rules to account for it. Thus, the examples covered by the "light rules" are likely to represent noise, and therefore are removed from the dataset. Experiments with AQ14-NT applied to a variety of engineering and computer vision problems have shown that it systematically produces classification rules that both perform better and also are much simpler than those produced by other versions of AQ.

3.1.6. AQ15-GA (AQ15 with attribute selection by a genetic algorithm)

In this approach implemented by Vafaie and De Jong (1991), genetic algorithms are used in conjunction with AQ15. The former are used to explore the space of all subsets of a given attribute set. Each of the selected attribute subsets is evaluated (its fitness measured) by invoking AQ15 and measuring the recognition rate of the rules produced.

The evaluation procedure as shown is divided into three main steps. After an attribute subset is selected, the initial training data, consisting of the entire set of attribute vectors and class assignments corresponding to examples from each of the given classes, is reduced. This is done by removing the values for attributes that were eliminated from the original attribute set. The second step is to apply a classification process (AQ15) to the reduced training data set. The decision rules that AQ15 generates for each of the given classes in the training data are then used for classification. The last step is to use the rules produced by the AQ algorithm in order to evaluate the classification and hence, recognition with respect to the test data.

In order to use genetic algorithms as the search procedure, it is necessary to define a fitness function which properly assesses the decision rules generated by the AQ algorithm. The fitness function takes as an input a set of attribute or attribute definitions, a set of decision rules created by the AQ algorithm, and a collection of testing examples defining the attribute values for each example. The fitness function then views the AQ-generated rules as a form of class description that, when applied to a vector of attribute or attribute values, will evaluate to a number. It is evaluated for every attribute subset by applying the following steps: For every testing example a match score is evaluated for all the classification rules generated by the AQ algorithm, in order to find the rule(s) with the highest or best match. At the end of this process, if there is more than one rule having the highest match score, one rule will be selected based on the chosen conflict resolution process. This rule then represents the classification for the given testing example. If this is the appropriate classification, then the testing example has been recognized correctly. After all the testing examples have been classified, the overall fitness function will be evaluated by adding the weighted sum of the match score of all of the correct recognitions and subtracting the weighted sum of the match score of all of the incorrect recognitions.

3.2. AQ-Derived or Supplementary Programs

The INDUCE family of programs, developed by Michalski and Larson, can, like the AQ programs, learn classification rules from examples. Its input data is, however, stored in the form of predicates describing the events rather than in the form of tables of attributes and values, as is the case with AQ. As a result, INDUCE is well suited for learning structural descriptions of objects in which the relationships between different parts of the objects and existential or universal quantifiers may be very useful in distinguishing between classes (Larson, 1977). Because the "attributes" in the output descriptions can be generated by combining the objects' attributes using various relationships, INDUCE is regarded as a pioneering program in the area of constructive induction.

The inductive learning system RIGEL (**R**easoned **I**nductive **G**enera**l**ization) is a modified version of INDUCE. RIGEL's covering algorithm uses interleaved specialization and generalization steps, rather than the extend against operator used by INDUCE, and works from non-randomly selected seed examples. Additionally, the knowledge representation language in RIGEL is fully capable of describing universal and numerical quantifications (Gamallo, Mana and Saitta, 1991).

SPARC (**S**equential **P**attern **R**ecognition) employs AQ and two other learning models to learn patterns in sequences, as opposed to sets of events (Michalski, Ko and Chen, 1985; Dieterrich and Michalski, 1986; Michalski, Ko and Chen, 1986). As with the structural knowledge used by INDUCE, sequential domains are more complex than those applicable to AQ itself; an event's position in the sequence and relation to its neighbors must be taken into account when learning patterns.

AQPLUS (Forsburg, 1976) reduces the attribute space in a domain by selecting and retaining the most useful attributes. It operates by repeatedly running AQ on different subsets of the attribute set and evaluating the results so as to generate utility values for the different attributes.

EMERALD (Kaufman, Michalski and Schultz, 1989) integrates five inductive learning programs - AQ15, INDUCE, CLUSTER, SPARC and ABACUS - into an environment designed to be a tool

for education and research in machine learning. Users can interact with the programs by challenging them with problems from predefined domains.

AQR is a straightforward reconstruction and implementation of the basic AQ algorithm (Clark and Niblett, 1989). As does AQ, AQR builds rules in disjunctive normal form for each decision class using the Star methodology, as controlled by user-defined criteria.

CN2 is designed to combine the advantages of both ID3 (decision tree) and AQ (decision rule) based algorithms. The CN2 learning algorithm generates a *decision list* - an ordered set of condition-action rules in which the testing and application of the rules occurs in top-to-bottom sequence (Clark and Niblett, 1989). The idea behind such a representation allows the algorithm to cope with noisy data, since the data descriptions are not required to be consistent with all of the input examples. CN2 is a computationally expensive algorithm in that it exhaustively searches possible specializations of the rules it discovers.

CAQ (Whitehall, Lu and Stepp, 1990) was developed to improve performance in engineering domains where continuous numeric values and noisy data may be present. Instead of quantizing subsets of a range of possible values, CAQ's continuous domain knowledge representation incorporates tolerance thresholds, such that any two values closer to one another than the threshold are regarded as equivalent. Furthermore, CAQ uses a statistical method to determine the "best" boundary point between classes.

DLG generates disjunctive descriptions of classes of events much like AQ does (Webb, 1992), but instead of using the Star/extend against method to create rules, DLG uses *least generalization* (Plotkin, 1970). The description covering a seed example is sequentially expanded to cover the other examples of that class, and if the expanded description has a higher quality rating (typically based on the number of positive and negative examples covered) than the basis description, it replaces that description in the sequential generation of the rule. Computationally, this method is simpler than AQ.

INLEN (**I**nference and **L**earning) is an environment for discovering patterns, trends, exceptions and facts in databases (Michalski et al, 1992). The "toolbox" architecture allows users to call upon different learning and discovery programs depending on the task at hand. INLEN uses AQ to discover rules distinguishing different classes in relational tables.

4. An EXPERIMENTAL COMPARISON: THE MONKS' PROBLEMS

4.1 Problem definition

The domain of the MONKs' problems is an abstraction of a robots domain previously used in a comparison of learning methods' performance [Wnek et al, 1990]. The problems were presented in a way such that physical attributes and values were replaced by generic attributes and numerical values, for example, *x1 is 2* instead of *head_shape is square*. In this paper, we will often revert to the latter notation or hybrid notation (such as *head_shape is 2*) when it makes the presentation more understandable. A complete listing of the attributes and values in both formats is shown in Table 1. All examples come from this event space, which spans 6 multiple-valued attributes (it has a total of 432 possible examples). The sizes of the value sets of the attributes, x1, x2, ..., x6, are 3, 3, 2, 3, 4, and 2, respectively.

Attribute	x1	x2	x3	x4	x5	x6
Robot Att.	*head_shape*	*body_shape*	*is_smiling*	*holding*	*jacket_color*	*has_tie*
Value						
1	round	round	yes	sword	red	yes
2	square	square	no	balloon	yellow	no
3	octagon	octagon		flag	green	
4					blue	

Table 1. Attributes and values in the robots domain

Problem 1.

There were 124 training examples, which represented 30% of the total event space (62 positive and 62 negative). The testing examples were the set of all possible examples (216 positive and 216 negative). The goal concept was as follows:

(head_shape = body_shape) or (jacket color = red)

This problem represented a straightforward concept in which the values of two attributes had to be compared, either implicitly or explicitly. A diagrammatic visualization of this problem is shown in Figure 2. The dark area in this figure represents the positive concept, and the white area represents the concept negation, or the set of all possible counterexamples. Positive and negative training examples are represented by + and - respectively.

Problem 2.

There were 169 training examples, which represented 40% of the total event space (105 positive and 64 negative). The testing examples were all possible examples (190 positive and 242 negative). The goal concept was as follows:

Exactly two of the six attributes have their first value

In other words, exactly two of the x_i have values of 1. This concept does not have a simple representation in Disjunctive Normal Form, and therefore poses a problem for many symbolic learning methods.

A diagrammatic visualization of this problem is shown in Figure 3.

Problem 3.

There were 122 training examples, which represented 30% of the total event space (62 positive and 60 negative). The testing examples were the set of all possible examples (204 positive and 228 negative). Noise was inserted into the example set so that 5% of the training examples were misclassified. The goal concept was as follows:

(jacket_color is green and holding is sword) or
(jacket color is not blue and body_shape is not octagon)

The goal of this problem was to test learning programs' adaptation to noisy environments.

A diagrammatic visualization of this problem is shown in Figure 4. The plus in the white area and the minuses in the black area represent the noisy training examples.

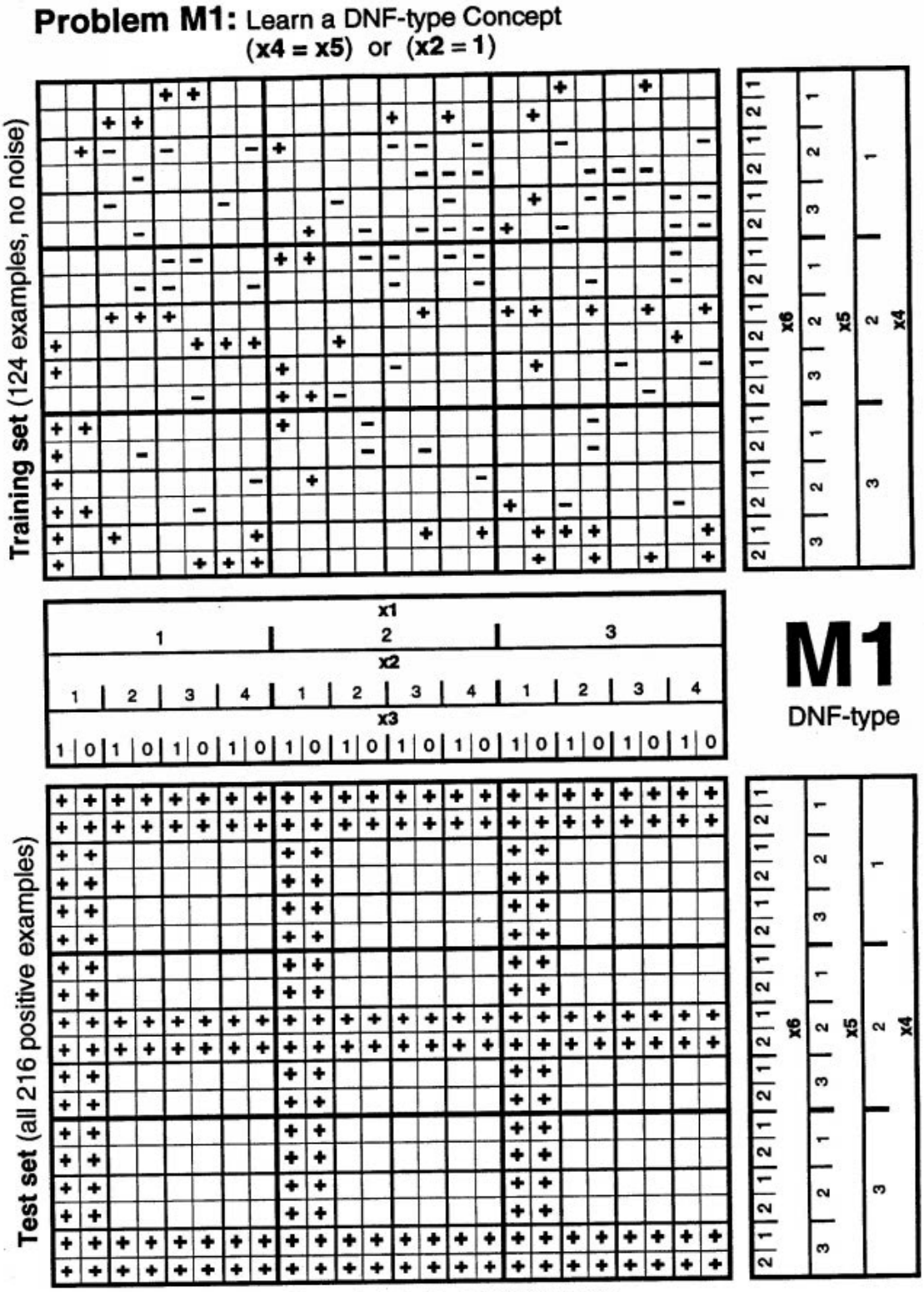


Figure 2. The first MONKs' Problem

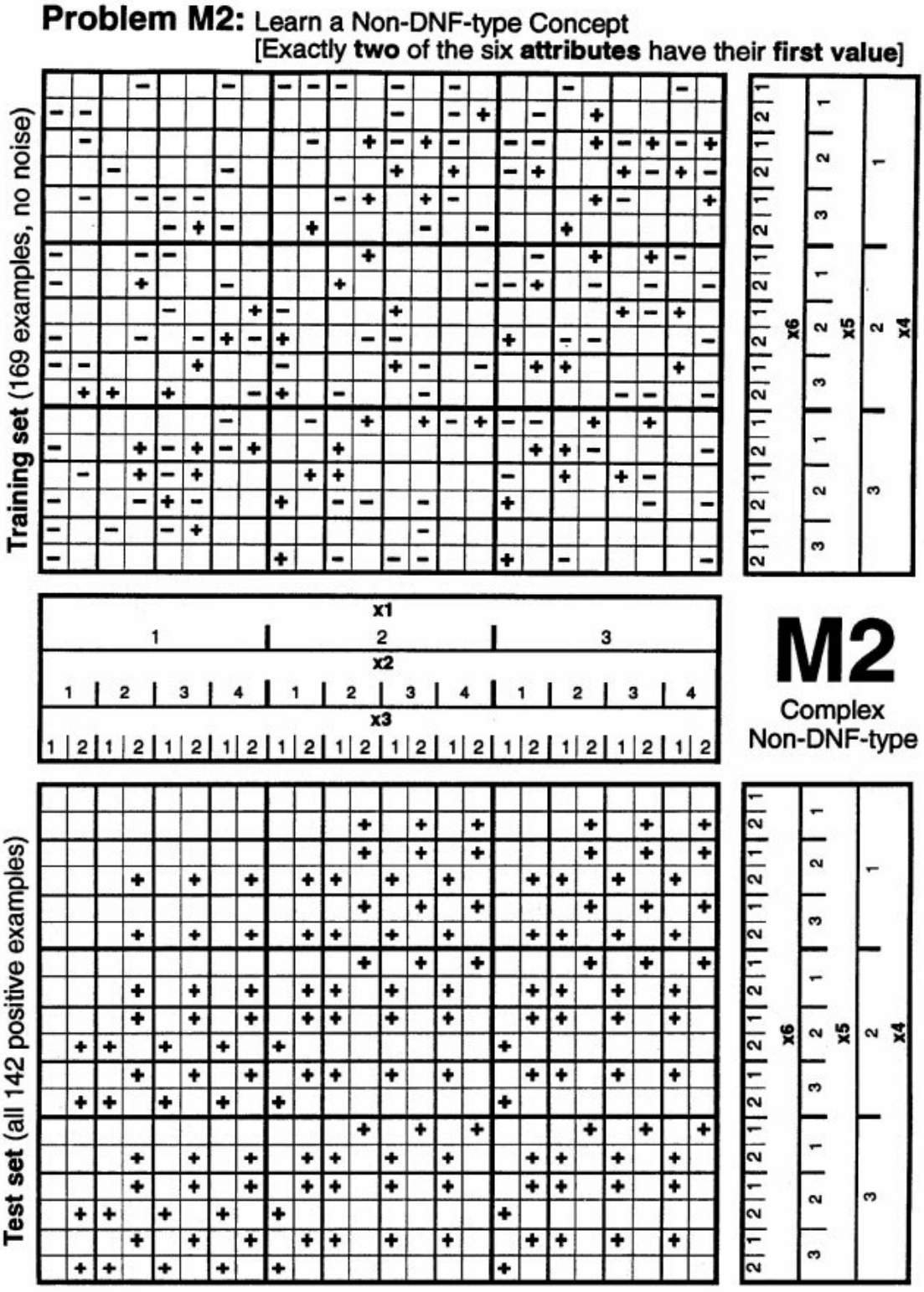


Figure 3. The second MONKs' Problem

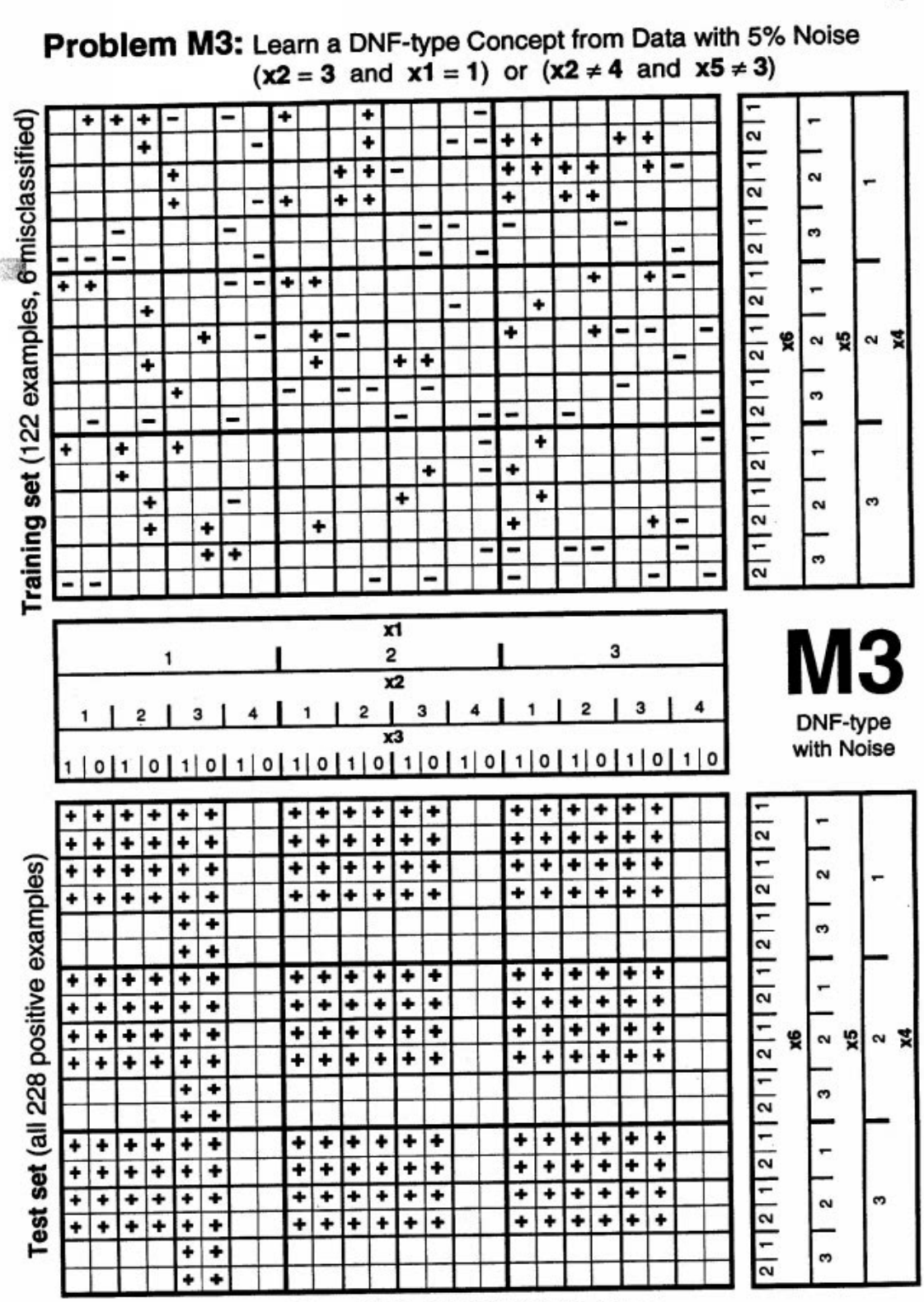


Figure 4. The third MONKs' Problem

5. SUMMARY OF RESULTS

Below is a listing of the rules obtained by the various AQ programs (AQ17-DCI, AQ17-HCI, AQ17-1, AQ15-GA, AQ15-FCLS or AQ14-NT), and the results of testing them on the testing examples. A listing of all the data, both training and testing sets, as provided by the creators of the problems, is in the Appendix.

Rules generated by different programs were tested using the ATEST program that computes a confusion matrix (Reinke, 1984). ATEST computes the so-called *consonance degree* between an unknown example and the rules for each decision class. The output from this program includes numerical evaluations of the accuracy of the rules based on the percentage of the testing examples correctly classified (by choosing the rule that best fits the example), and the percentage of examples precisely matched by the correct decision rule. These percentages are output by ATEST as OVERALL % CORRECT-FLEX-MATCH and OVERALL % CORRECT-100% MATCH, respectively.

Details of the different programs, and of the AQ algorithm underlying these programs are given in Sections 3.1 and 2.1 respectively. It should be noted that results are not always presented for each of these programs as applied to each of the three problems. As indicated above, these programs derive from the same basic method, each adding features appropriate to specific types of problems. The different programs derived basically the same rule for the first problem; the ones shown here are the ones whose knowledge representation schema allowed for the most elegant presentation of the output. We felt that for the sake of brevity and emphasis on the matching of the programs' different features with the types of problems to be solved, we should present only the results of the programs better suited for the given type of problem. For example, we felt that there was no reason to apply AQ14-NT, a program with special features to cope with noisy data to Problem 2, a problem in which data were without noise, and the testing events were 100% correctly classified by the rules obtained by other programs. For the same reason, we do not emphasize the results of the application of the data-driven constructive induction program AQ17-DCI to Problem 3; its algorithm is strictly data-driven, and as such is less suitable for learning from noisy data than other AQ programs.

The results of applying the six AQ programs to the three MONKs' problems are summarized in Table 2. The percentages represent the correct classification percentages as calculated by ATEST.

PROGRAM	#1	#2	#3
AQ17-DCI	100%	100%	97.2%
AQ17-HCI	100%	93.1%	100%
AQ15-FCLS	100%	92.6%	97.2%
AQ14-NT	100%	N/A	100%
AQ15-GA	100%	86.8%	100%
AQ17-HCI_DCI	100%	100%	100%

Table 2. Summary of AQ programs' correct classification percentages

5.1. Results for the 1st problem

5.1.1. Rules obtained by AQ17-DCI and AQ17-1

These are the rules obtained by AQ17-DCI, a version of the AQ program that employs data-driven constructive induction. The results include 1 rule for Class 0 (that represents positive examples of the concept), and 2 rules for Class 1 (that represents the negative examples):

Class 0:

Rule 1 [jacket_color ≠ red] & [head_shape ≠ body_shape] (total:62, unique:62)

Class 1:

Rule 1 [head_shape = body_shape] (total:41, unique:33)
Rule 2 [jacket_color = red] (total:29, unique:21)

In the above rules, expressions in [] denote individual conditions in a rule, "total" means the total number of training examples of the given class covered by the rule, and "unique" means the number of training examples covered by that rule only, and not by any other rules.

There is only one rule for Class 0, and there are two rules for Class 1. Thus, if either of the Class 1 rules is matched by a given instance, that instance is then classified as belonging to Class 1. A set of such rules is logically equivalent to a disjunction of conjunctions. The syntax of the rules is defined formally according to the variable-valued logic calculus VL_1 (Michalski, 1973a). Individual rules correspond to "complexes" in VL_1.

These rules were also found by AQ17-1. While the combined AQ17 algorithm (Section 3.1.4) initially constructs hypothesis-based rules, the rules discovered by this method (see Section 5.1.2), although 100% accurate, were found to be too complex. The program therefore attempted data-driven constructive induction and obtained the rules shown above.

The results of applying the rules to the testing examples were:

RESULTS	
OVERALL % CORRECT FLEX MATCH:	100.00
OVERALL % CORRECT 100% MATCH:	100.00

where:
% FLEX MATCH means the percentage of the correctly classified examples within the total set of testing examples, using a flexible matching function (see Reinke, 1984), and % 100% MATCH means that the percentage of correctly classified examples that matched the rules exactly.

The number of testing events satisfying individual rules in the correct class description is given in the table below:

	RULES	
	R1	R2
CLASS 0	215	
CLASS 1	144	108

5.1.2. Rules obtained by AQ17-HCI

These are the rules obtained by AQ17-HCI, a version of the AQ program that employs hypothesis-driven constructive induction (see Section 3.1.3. The results include one rule for Class 0 that represents positive examples of the concept, and one rule for Class 1 that represents the negative examples:

Class 0:

Rule 1 [Neg = false] (total:62, unique:62)

Class 1:

Rule 1 [Pos = false] (total:62, unique:62)

where Neg and Pos are attributes constructed from the original ones, or intermediate ones, as defined below (these rules, as one can check, are logically equivalent to the AQ17-DCI generated rules)

c01 <:: [head_shape = round] & [body_shape ≠ round] & [jacket_color ≠ red]
c05 <:: [head_shape = square] & [body_shape ≠ square] & [jacket_color ≠ red]
c08 <:: [head_shape = octagon] & [body_shape ≠ octagon] & [jacket_color ≠ red]
c10 <:: [head_shape = round] & [body_shape = round]
c12 <:: [jacket_color = red]
c13 <:: [head_shape = square] & [body_shape = square]
c15 <:: [head_shape = octagon] & [body_shape = octagon]
Pos <:: [c10 = false] & [c12 = false] & [c13 = false] & [c15 = false]
Neg <:: [c01 = false] & [c05 = false] & [c08 = false]

TEST RESULTS - SUMMARY
OVERALL % CORRECT FLEX MATCH: 100.00
OVERALL % CORRECT 100% MATCH: 100.00

Number of testing events satisfying individual rules in the correct class description:

	RULES R 1
CLASS 0	215
CLASS 1	216

Other programs either were either not used on this problem, or generated similar results.

5.2. Results for the 2nd problem

5.2.1. Rules obtained by AQ17-DCI and AQ17-1

The rules below were obtained by AQ17-DCI, which is capable of generating all kinds of new attributes from the original attributes, and by AQ17-1, which incorporates the DCI method. For the problem at hand, the program found that a new attribute that expresses the number of variables in the training examples that have some specific value is highly relevant to this problem. Such an attribute is assigned by the program the name #VarEQ(x), which means "the number of variables with value of rank *x* (in their domain)" in an example. The lowest value in the domain has rank 1, the next lowest has rank 2, etc. In this case, the relevant attribute was #VarEQ(1) - the number of variables taking on their first, or lowest value. Based on this attribute, the program constructed appropriate decision rules. There were two one-condition rules for Class 0, representing the positive examples of the concept, and one rule for Class 1 that represents the negative examples. The rule for Class 1 is logically equivalent to the negation of the union (disjunction) of the rules for Class 0.

Class 0:

Rule 1 [#VarEQ(1) ≥ 3]
Rule 2 [#VarEQ(1) ≤ 1]

Class 1:

Rule 1 [#VarEQ(1) = 2]

The results of applying the rules to the testing examples were:

RESULTS	
OVERALL % CORRECT FLEX MATCH:	100.00
OVERALL % CORRECT 100% MATCH:	100.00

4.2.2. Rules obtained by AQ17-HCI

There are 4 top level rules for Class 0 (positive examples), and 6 top level rules for Class 1 (negative examples):

Class 0:

Rule 1 [Pos73 = true] (total:90, unique:49)
Rule 2 [c14 = false] & [c26 = false] & [c53 = false] & [c67 = false] & [c72 = false] & [Neg74 = false] (total:38, unique:6)
Rule 3 [holding = balloon or flag] & [c6 = false] & [c20 = false] & [Neg74 = false] (total:22, unique:5)
Rule 4 [head_shape = square] & [has_tie = false] & [c44 = false] & [c50 = false] & [Neg74 = false] (total:6, unique:2)

Class 1:

Rule 1 [Neg74 = true] (total:43, unique:30)
Rule 2 [jacket_color ≠ red] & [has_tie = true] & [c60 = true] & [Pos73 = false] (total:17, unique:4)
Rule 3 [head_shape ≠ round] & [body_shape = square] & [c28 = false] & [Pos73 = false] (total:16, unique:7)
Rule 4 [body_shape = octagon] & [c48 = true] & [c66 = true] (total:4, unique:2)
Rule 5 [jacket_color = green] & [c43 = true] & [c52 = false] & [c53 = false] & [c55 = false] & [c69 = true] & [Pos73 = false] (total:4, unique:2)
Rule 6 [body_shape = octagon] & [c9 = false] & [c10 = true] & [c23 = true] & [c32 = true] (total:3, unique:1)

Attributes "c_i, i=1..72" "Pos73," and "Neg74" were constructed during the learning process. The following is a partial listing of the constructed attributes that were relevant to the discovered rules; (Thrun, Mitchell and Cheng, 1991) contains the complete listing.

c2 <:: [jacket_color = red or blue]
c4 <:: [body_shape ≠ round] & [is_smiling = false]
c5 <:: [head_shape ≠ round] & [is_smiling = false]
c6 <:: [head_shape ≠ round] & [body_shape ≠ round]
c7 <:: [holding ≠ flag] & [jacket_color ≠ yellow]
c9 <:: [head_shape ≠ square] & [jacket_color ≠ red]
c10 <:: [holding ≠ flag] & [jacket_color ≠ red]
...
c70 <:: [jacket_color ≠ red] & [c18 = false]
c72 <:: [jacket_color ≠ blue] & [c37 = true]

Pos73 <:: [c4 = false] & [c16 = false] & [c33 = false] & [c39 = false] & [c40 = false] or
[c15 = false] & [c43 = false] & [c47 = false] & [c68 = false] or
[body_shape ≠ octagon] & [c21 = false] & [c41 = true] & [c44 = false] &
[c65 = true] & [c67 = false] or
[c33 = true] & [c60 = true]

Neg74 <:: [c4 = false] & [c42 = true] & [c56 = false] & [c65 = true] & [c68 = true] or
[c2 = false] & [c4 = false] & [c16 = false] & [c17 = true] & [c26 = true] or
[is_smiling = false] & [holding ≠ sword] & [c14 = false] & [c41 = true] &
[c43 = true] & [c59 = false] & [c69 = false] & [c70 = false] or
[has_tie = false] & [c5 = true] & [c44 = false] & [c61 = false]

TEST RESULTS - SUMMARY	
OVERALL % CORRECT FLEX MATCH:	93.06
OVERALL % CORRECT 100% MATCH:	86.57

The above summary of the results shows that the rules generated by AQ17-HCI approximate quite well the concept in Problem 2 although they use only logical operators. This result is quite interesting because concepts such as the one in Problem 2 are among the most difficult to learn using solely logic-based inductive learners (classical rule learning or decision tree learning programs). This result demonstrates the power of hypothesis-driven constructive induction.

Number of testing events satisfying individual complexes in the correct class description:

	RULES					
	R 1	R 2	R 3	R 4	R 5	R 6
CLASS 0	232	84	54	12		
CLASS 1	77	44	32	10	5	4

5.2.3. Rules obtained by AQ17-FCLS

These are the rules obtained by AQ17-FCLS, a version of the AQ program that learns flexible concepts by generating rules that permit partial matching. The threshold parameter indicates the minimum percentage of the individual conditions in the rule that must be satisfied for the rule to apply. The results include two rules for Class 0 that represent positive examples of the concept, and 18 rules for Class 1 that represent the negative examples. The discovered rules fully encompass Class 0, but they failed to get a complete grasp of the concept of Class 1:

Class 0:

Rule 1 [head_shape = round] & [body_shape = round] & [is_smiling = true] & [holding = sword] & [jacket_color = red] & [has_tie = true]
with THRESHOLD = 50 %
(Total positive examples covered: 64)

This rule says that three or more variables must have rank equal to 1.

Rule 2 [head_shape ≠ round] & [body_shape ≠ round] & [is_smiling = false] & [holding ≠ sword] & [jacket_color ≠ red] & [has_tie = false]
with THRESHOLD = 83 % (5/6)
(Total positive examples covered: 41)

This rule says that five or six out of six variables must not have their first values, or equivalently, that at most one variable may have its first value. Thus the disjunction of these two rules above indicates that the number of variables which have their first value cannot be equal to 2.

These rules classified correctly 100% of the examples of Class 0.

Class 1:

Since the current program does not have the ability to express the negation of the above two rules for Class 0, to program generated many "lightweight" rules to cover all examples of Class 1. The overall performance using the flexible match was not 100% because in some cases when an example matched equally well the rules for both classes, an incorrect class was chosen. In the next version of the program, we plan to include the missing negation operator.

Rule 1 [is_smiling = true] & [holding ≠ sword] & [jacket_color = yellow] & [has_tie = false]
with THRESHOLD = 100 %
(Total positive examples covered: 8)

Rule 2 [head_shape ≠ round] & [body_shape ≠ round] & [is_smiling = true] & [holding ≠ sword] & [jacket_color ≠ red] & [has_tie = true]
with THRESHOLD = 100 %
(Total positive examples covered: 9)

Rule 3 [head_shape ≠ round] & [body_shape ≠ round] & [is_smiling = false] & [holding ≠ sword] & [jacket_color = yellow] & [has_tie = false]
with THRESHOLD = 100 %
(Total positive examples covered: 7)

Rule 4 [head_shape = octagon] & [body_shape = round] & [is_smiling = true] & [holding = sword] & [jacket_color = green] & [has_tie = false]
with THRESHOLD = 83 %
(Total positive examples covered: 5)

...

Rule 18 [head_shape ≠ round] & [body_shape = round] & [is_smiling = false] & [holding ≠ sword] & [jacket_color ≠ red] & [has_tie = true]

with THRESHOLD = 100 %
(Total positive examples covered: 3)

TEST RESULTS - SUMMARY	
The percentage of correctly classified testing events:	92.6%
The percentage of correctly classified testing events in Class 0:	100.0%
The percentage of correctly classified testing events in Class 1:	85.2%
The total number of rules in the descriptions:	2 for Class 0 18 for Class 1
The total number of conditions in the descriptions:	110

5.3. Results for the 3rd problem

5.3.1. Rules obtained by AQ17-HCI and AQ17-1

Below are the rules obtained by the hypothesis-driven constructive induction method, both as a part of AQ17-1 and as a stand-alone learning program:

Class 0:

Rule 1 [Pos1 = true] (total:49, unique:49)
Rule 2 [body_shape ≠ round] & [holding ≠ sword] & [jacket_color = green] (total:11, unique:11)
Rule 3 [body_shape = round] & [holding = sword] & [jacket_color = green] (total:1, unique:1)
Rule 4 [body_shape = square] & [holding = balloon] & [jacket_color = yellow] (total:1, unique:1)

Class 1:

Rule 1 [Neg2 = true] (total:57, unique:57)
Rule 2 [body_shape = octagon] & [holding = sword] & [jacket_color = green or blue] (total:3, unique:3)

where Pos1 and Neg2 are attributes constructed from the original ones (Wnek & Michalski, 1991)

Pos1 <:: [jacket_color = blue] or
[body_shape = octagon] & [jacket_color ≠ green]

Neg2 <:: [body_shape ≠ octagon] & [jacket_color ≠ blue]

TEST RESULTS - SUMMARY
OVERALL % CORRECT FLEX MATCH: 100.00
OVERALL % CORRECT 100% MATCH: 86.11

Since this problem involves noisy data, the flexible match accuracy figures should always be used. The results using a 100% match are shown just for comparison.

Number of testing events satisfying individual rules in the correct class description:

	RULES			
	R 1	R 2	R 3	R 4
CLASS 0	180	24	0	0
CLASS 1	216	12		

Rules 3 and 4 for Class 0 were each exceptions, covering a single example of the class. As can be seen from the above chart, in each case that example was a noisy event, which was also covered by the wider ranging rules of Class 1. AQ's evaluation capabilities were able to select the proper rule to apply in a flexible match.

5.3.2. Rules obtained by AQ14-NT

These are the rules obtained by AQ14-NT, a version of the AQ program that employs a noise-filtration technique. The results include one rule for Class 0 that represents positive examples of the concept, and one rule for Class 1 that represents negative examples.

After only two loops of concept-driven filtration of training dataset (with truncation parameter equal to 10%) and repeated learning, we received the following set of rules:

Class 0:

Rule 1 [jacket_color = blue]
Rule 2 [body_shape = octagon] & [holding ≠ sword]
Rule 3 [body_shape = octagon] & [jacket_color = red or yellow]

Class 1:

Rule 1 [body_shape ≠ octagon] & [jacket_color ≠ blue]
Rule 2 [holding = sword] & [jacket_color = green]

These rules recognized all test data correctly, i.e., on the 100% level.

The program was able to filter out the events representing noise in the data and achieve such a high degree of recognition.

5.3.3. Rules obtained by AQ17-FCLS

These are the rules obtained by AQ17-FCLS. The results include two rules for Class 0 that represent positive examples of the concept, and one rule for Class 1 that represents the negative examples. The threshold parameter indicates the minimum percentage of selectors in the rule that must be true for the rule to apply. This set of rules is intentionally incomplete and inconsistent with the training set since it was generated with a 10% error tolerance. This produced better results than other tolerances that were tried.

Class 0:

Rule 1 [head_shape ≠ round] & [body_shape = octagon] & [jacket_color = blue]

with THRESHOLD = 67 %
(Total positive examples covered: 42)

Rule 2 [head_shape = round] & [body_shape = octagon] & [jacket_color = blue]
with THRESHOLD = 67 %
(Total positive examples covered: 26)

Class 1:

Rule 1 [body_shape = round or square] & [jacket_color ≠ blue]
with THRESHOLD = 100 %
(Total positive examples covered: 57)

TEST RESULTS - SUMMARY	
The percentage of correctly classified testing events:	97.2%
The percentage of correctly classified testing events in Class 0:	100.0%
The percentage of correctly classified testing events in Class 1:	94.7%
The total number of rules in the descriptions:	2 for Class 0 1 for Class 1
The total number of conditions in the descriptions:	8

5.3.4. Rules obtained by AQ15-GA

Below are the rules obtained by AQ15-GA, a program that uses a genetic algorithm in conjunction with the AQ rule-generation algorithm. The first rule is for the positive examples of the concept, Class 0, and the second for the negative examples, Class 1. A genetic algorithm determined that 3 attributes (body_shape, holding, and jacket_color) were the most meaningful. Using these, the rules discovered were as follows:

Class 0:

Rule 1 [jacket_color = blue]
Rule 2 [body_shape = octagon] & [jacket_color = red or yellow]
Rule 3 [body_shape ≠ round] & [holding ≠ sword] & [jacket_color = green]
Rule 4 [body_shape = round] & [holding = sword] & [jacket_color = green]
Rule 5 [body_shape = square] & [holding = balloon] & [jacket_color = yellow]

Class 1:

Rule 1 [body_shape ≠ octagon] & [jacket_color ≠ blue]
Rule 2 [body_shape = octagon] & [holding = sword] &
[jacket_color = green or blue]

Results on testing the rules on testing events using program ATEST:

TEST RESULTS - SUMMARY
OVERALL % CORRECT FLEX MATCH: 100.00
OVERALL % CORRECT 100% MATCH: 100.00

6. COMPARISON OF AQ WITH OTHER PROGRAMS

In the Thrun study (Thrun, Mitchell & Cheng, 1991), the AQ programs performed very favorably in comparison with the other programs examined (Table 3). Programs and algorithms that were tested on these problems included Assistant Professional (Cestnik, Kononenko & Bratko); mFOIL (Dzeroski); ID5R, IDL, ID5R-hat and TDIDT (Van de Velde); ID3 (with and without windowing), ID5R, AQR, CN2 and CLASSWEB (Kreuziger, Hamman and Wenzel); PRISM (Keller); ECOBWEB (Reich & Fisher); Backpropagation (Thrun) and Cascade Correlation (Fahlman).

RESEARCH GROUP AFFILIATION AND LEARNING PROGRAMS APPLIED	#1 (DNF-type)	#2 (non-DNF)	#3 (noisy DNF)
Jozef Stefan Institute, Ljubljana, Slovenia			
Assistant Professional	100%	81.3%	100%
mFOIL	100%	69.2%	100%
Vrije Universiteit Brussel, Brussels, Belgium			
ID5R	81.7%	61.8%	
IDL	97.2%	66.2%	
ID5R-hat	90.3%	65.7%	
TDIDT	75.7 %	66.7%	
Institute for Real-Time Computer Control Systems and Robotics and University of Karlsruhe, Karlsruhe, Germany			
ID3	98.6%	67.9%	94.4%
ID3, no windowing	83.2%	69.1%	95.6%
ID5R	79.7%	69.2%	95.2%
AQR	95.9%	79.7%	87.0%
CN2	100%	69.0%	89.1%
CLASSWEB 0.10	71.8%	64.8%	80.8%
CLASSWEB 0.15	65.7%	61.6%	85.4%
CLASSWEB 0.20	63.0%	57.2%	75.2%
University of Zurich, Zurich, Switzerland			
PRISM	86.3%	72.7%	90.3%
Carnegie Mellon University, Pittsburgh, Pennsylvania and Vanderbilt University, Nashville, Tennessee			
ECOBWEB leaf prediction	71.8%	67.4%	68.2%
ECOBWEB l.p. & information utility	82.7%	71.3%	68.0%
Carnegie Mellon University, Pittsburgh, Pennsylvania			
Backpropagation	100%	100%	93.1%
Backpropagation with weight decay	100%	100%	97.2%
Cascade Correlation	100%	100%	97.2%
Center for Artificial Intelligence, George Mason University, Fairfax, Virginia			
AQ17-DCI	100%	100%	94.2%
AQ17-HCI	100%	93.1%	100%
AQ17-1	100%	100%	100%
AQ15-FCLS		92.6%	97.2%
AQ14-NT			100%
AQ15-GA	100%	86.8%	100%

Table 3. Classification Results Reported by Thrun (updated to show recent AQ17 results)

The results shown in Table 3 provide a partial update to the results reported by Thrun. They include an improved result from AQ17-DCI for problem 3, and also the AQ17-1 results that were achieved after the initial study was completed. They do not, however, reflect later results obtained by other researchers. While the results do not guarantee either that all the programs were employed with optimal parameter settings or that these results can be extended to other concept learning problems, it is interesting to note that in the initial study, only the rule-based AQ programs were able to come up with 100% classification rates on both the k-of-n concept and the noisy concept. In addition, the lower bounds on the classification rates of the AQ programs were very respectable in comparison with the other programs, indicating that these programs can generate reasonable results, even when the optimal tool is not being used.

An earlier study by Wnek, Sarma, Wahab and Michalski (1990) compared the standard version of AQ15, a backpropagation neural net, the CFS classifier system and the C4.5 decision tree learning program by testing their ability to learn five concepts created by human subjects. Their results indicated that the symbolic algorithms were better suited toward these symbolic-oriented problems, both in terms of error rate and knowledge complexity.

Spears and Gordon (1991) compared NEWGEM (a predecessor to AQ15), C4.5, and GABIL, a genetic algorithm-based system, on an artificial disjuncts-and-conjuncts domain and on breast cancer case data. They found comparable error rates among the systems and proposed a multistrategy system that would improve the GABIL results by incorporating elements of other systems in the form of genetic operators.

Bergadano, Matwin, Michalski and Zhang (1990) compared three AQ-based systems with ASSISTANT and with an exemplar-based method on two real world domains: congressional voting and labor negotiations. The AQ programs generated simpler and more accurate rules, particularly when flexible concept learning (such as in AQ15-FCLS) was employed.

7. CONCLUSIONS

We have described a family of inductive learning programs that are based on the AQ learning algorithm. Recently developed programs in this family are capable of using constructive induction techniques, learning flexible concepts, filtering out noisy data, and taking multistrategy approaches which can use the abilities of different learning methods to their advantage.

When tested on the MONKs' problems, the AQ-based programs were able to generate rules that closely approximated or exactly matched the target concepts. By choosing the proper program, users could generate a simple and accurate concept representation.

Researchers have been turning more frequently to comparisons of different learning paradigms. No single set of test problems will generate results that will hold true universally. Nonetheless, by performing different comparisons, and by establishing diverse sets of benchmark problems, we can begin to discover details of the suitability of different learning methods to different types of problems.

As the testing of the AQ programs shows, they are very powerful, and ready for use in various practical applications. By combining some of the specialized techniques and modules, more powerful and versatile modules can be created. For example, the integration of the DCI and HCI modules of AQ17 into a single unit resulted in a program capable of adapting to the challenges presented by both the second and the third MONKs' problem. An important topic of this ongoing research (and of many other multistrategy projects) is the development of a unified control system that can integrate smoothly the capabilities of the different techniques so that they can be used most efficiently.

Acknowledgments

This research was done in the Artificial Intelligence Center of George Mason University. The activities of the Center are supported in part by the by the National Science Foundation under grant No. IRI-9020266., in part by the Defense Advanced Research Projects Agency under the grant administered by the Office of Naval Research, No. N00014-91-J-1854, and in part by the Office of Naval Research under grant No. N00014-91-J-1351.

References

Bala, J.W. and Pachowicz, P.W, "Recognizing Noisy Patterns of Texture via Iterative Optimization and Matching of their Rule Description", *Reports of Machine Learning and Inference Laboratory*, MLI 90-18, George Mason University, Fairfax, VA 1990.

Bergadano, F, Matwin, S, Michalski, R.S. and Zhang, J, "Learning Two-Tiered Descriptions of Flexible Concepts: The Poseidon System," *Reports of Machine Learning and Inference Laboratory*, MLI 90-10, George Mason University, Fairfax, VA 1990.

Bloedorn, E., and Michalski, R.S, "Data-driven Constructive Induction in AQ17-DCI: A Method and Experiments," *Reports of Machine Learning and Inference Laboratory*, GMU, 1991.

Bloedorn, E., Michalski, R.S. and Wnek, J, "AQ17 - A Multistrategy Constructive Learning System," *Reports of Machine Learning and Inference Laboratory*, GMU, 1991.

Clark, P. and Niblett, T, "The CN2 Inductive Algorithm," *Machine Learning*, Vol. 3, pp. 261-283, 1989.

Cuneo, R.P, "Selected Problems of Minimalization of Variable-Valued Logic Formulas," Master's Thesis, Department of Computer Science, University of Illinois, Urbana, 1975.

Dietterich, T. and Michalski, R.S., "Learning to Predict Sequences," Chapter in *Machine Learning: An Artificial Intelligence Approach Vol. II*, R. S. Michalski, J. Carbonell and T. Mitchell (Eds.), Morgan Kaufmann Publishers, Los Altos, CA, pp. 63-106, 1986.

Forsburg, S, "AQPLUS: An Adaptive Random Search Method for Selecting a Best Set of Attributes From a Large Set of Candidates" Internal Report, Department of Computer Science, University of Illinois, Urbana, 1976.

Gamello, R, Mana, F. and Saitta, L, "RIGEL: An Inductive Learning System," *Machine Learning*, Vol. 6, pp. 7-35, 1991.

Jensen, G.M, "SYM-1: A Program that Detects Symmetry of Variable-Valued Logic Functions," Report No. 729, Department of Computer Science, University of Illinois, Urbana, May 1975.

Kaufman, K, Michalski, R.S. and Schultz, A, "EMERALD 1: An Integrated System of Machine Learning Programs for Education and Research, User's Guide," *Reports of Machine Learning and Inference Laboratory*, MLI 89-7, George Mason University, Fairfax VA, 1989.

Larson, J, "A Multi-Step Formation of Variable-Valued Logic Hypotheses," *Proceedings of the 1976 International Symposium on Multiple-Valued Logic*, Logan UT, May 25-28, 1976.

Larson, J, "INDUCE-1: An Interactive Inductive Inference Program in VL_{21} Logic System," Report No. 876, Department of Computer Science, University of Illinois, Urbana, 1977.

Michalski, R.S, "Recognition of Total or Partial Symmetry in a Completely or Incompletely Specified Switching Function," *Proceedings of the IV Congress of the International Federation on Automatic Control (IFAC), Vol. 27*, pp. 109-129, Warsaw, June 16-21, 1969.

Michalski, R.S, "On the Quasi-Minimal Solution of the Covering Problem," *Proceedings of the V International Symposium on Information Processing* (FCIP 69), Vol. A3 (Switching Circuits), Bled, Yugoslavia, pp. 125-128, 1969.

Michalski, R.S, "Discovering Classification Rules Using Variable-Valued Logic System VL1," *Proceedings of the Third International Joint Conference on Artificial Intelligence*, pp. 162-172, Stanford, California, August 20-23, 1973.

Michalski, R.S, "AQVAL/1 -- Computer Implementation of a Variable-Valued Logic System VL1 and Examples of its Application to Pattern Recognition," Implemented AQVAL Programs," *Proceedings of the First International Joint Conference on Pattern Recognition*, pp. 162-172, Washington DC, pp. 3-17, October 30 - November 1, 1973.

Michalski, R.S, "Toward Computer-Aided Induction: A Brief Review of Currently Implemented AQVAL Programs," Report No. UIUCDCS-R-77-874, Department of Computer Science, University of Illinois, Urbana, May, 1977.

Michalski, R.S, "A Theory and Methodology of Inductive Learning,"*Machine Learning: An Artificial Intelligence Approach*, R. S. Michalski, J. Carbonell and T. Mitchell (Eds.), pp. 83-134, Morgan Kaufmann, 1983.

Michalski, R.S., Ko, H. and Chen, K., "SPARC/E(V.2), An Eleusis Rule Generator and Game Player," ISG 85-11, UIUCDCS-F-85-941, Department of Computer Science, University of Illinois, Urbana, IL, February 1985.

Michalski, R.S., Ko, H. and Chen, K., "Qualitative Process Prediction: A Method and Program SPARC/G," *Expert Systems*, Guetler, C. (Ed.), Academic Press Inc., London, 1986.

Michalski, R.S. and Larson, J.B, "AQVAL/1 (AQ7) User's Guide and Program Description", Report No. 731, Department of Computer Science, University of Illinois, Urbana, 1975.

Michalski, R.S. and Larson, J.B, "Selection of Most Representative Training Examples and Incremental Generation of VL1 Hypotheses: the underlying methodology and the description of programs ESEL and AQ11," Report No. 867, Dept. of CS, University of Illinois, Urbana, 1978.

Michalski, R.S., Mozetic, I., Hong, J., and Lavrac, N., "The Multipurpose Incremental Learning System AQ15 and its Testing Application to Three Medical Domains," Proceedings AAAI, Philadelphia, August 11-15, 1986.

Mozetic, I., "NEWGEM: Program for Learning from Examples, Program Documentation and User's Guide", Report No. UIUCDCS-F-85-949, Department of Computer Science, University of Illinois at Urbana-Champaign, 1985.

Pachowicz, P.W. and J. Bala, "Improving Recognition Effectiveness of Noisy Texture Concepts through Optimization of Their Descriptions", *Proceedings of the 8th International Workshop on Machine Learning*, Evanston IL, pp. 625-629, 1991a.

Pachowicz, P.W. and Bala, J, "Advancing Texture Recognition through Machine Learning and Concept Optimization", *Reports of Machine Learning and Inference Laboratory*, Center for Artificial Intelligence, George Mason University, 1991b (also submitted to IEEE PAMI).

Pagallo, G. and Haussler, D, "Boolean Feature Discovery in Empirical Learning," *Machine Learning*, Vol. 5, pp. 71-99, 1990.

Plotkin, G.D, "A Note on Inductive Generalization," In Meltzer, B. and Michie, D. (Eds.) *Machine Intelligence 5*, Edinburgh University Press, Edinburgh, pp. 153-163, 1970.

Reinke, R.E., "Knowledge Acquisition and Refinement Tools for the ADVISE Meta-expert System," Master's Thesis, University of Illinois, 1984.

Spears, W.M. and Gordon, D.F, "Adaptive Strategy Selection for Concept Learning," *Proceedings of the First International Workshop on Multistrategy Learning*, Harpers Ferry WV, pp. 231-246, November 1991.

Stepp, R, "The Uniclass Inductive Program AQ7UN1: Program Implementation and User's Guide," Report No. 949, Department of CS, University of Illinois, Urbana, July, 1979.

Thrun, S.B, Mitchell, T. and Cheng, J. (Eds.), "The MONKs' Problems: A Performance Comparison of Different Learning Algorithms," Technical Report CMU-CS-91-197, Carnegie Mellon University, October 1991.

Vafaie, H. and De Jong, K, "Improving the Performance of a Rule induction System Using Genetic Algorithms," *Proceedings of the First International Workshop on Multistrategy Learning*, Harper's Ferry WV, pp. 305-315, 1991.

Webb, G, "Learning Disjunctive Class Descriptions by Least Generalization," Deakin University School of Computing and Mathematics Computing Series Report 5/92, Deakin University, Geelong 3217, Australia, 1992.

Whitehall, B, Lu, S. and Stepp, R.E, "CAQ: A Machine Learning Tool for Engineering," *International Journal for Artificial Intelligence in Engineering*, Vol. 5, No. 4, 1990.

Wnek, J. and Michalski, R.S., "Hypothesis-driven Constructive Induction in AQ17: A Method and Experiments," Twelfth International Joint Conference on Artificial Intelligence, Sydney Australia, August 1991.

Wnek, J., Sarma, J., Wahab, A. and Michalski, R.S., "Comparing Learning Paradigms via Diagrammatic Visualization," *Methodologies for Intelligent Systems, 5*, Ras, Z.W., Zemankova, M. and Emrich, M.L. (Eds.), North Holland, pp. 428-437, 1990.

Zhang, J. and Michalski, R.S., "Combining Symbolic and Numeric Representations in Learning Flexible Concepts: the FCLS System", in preparation.

APPENDIX: INPUT DATA FOR THE THREE MONKS' PROBLEMS

Below are the training and testing data sets that were presented to us for the MONKs' Problems. In each of the problems, the testing set consisted of the entire event space, and as such was a superset of the training set. This has an effect on some of the numeric results of the ATEST analyses of the rules discovered by the various programs.

The examples are presented in the following format:

$$\#: x_1\ x_2\ x_3\ x_4\ x_5\ x_6 \rightarrow C$$

where # is the position of the example in the lexicographic ordering of the event space, x_i represents the value of the ith variable, and C is the class to which the example is assigned.

Training Data Set #1

The training set of examples for the first problem:
(head_shape = body_shape) or (jacket color = red)

5: 1 1 1 1 3 1 -> 1
6: 1 1 1 1 3 2 -> 1
19: 1 1 1 3 2 1 -> 1
22: 1 1 1 3 3 2 -> 1
27: 1 1 2 1 2 1 -> 1
28: 1 1 2 1 2 2 -> 1
37: 1 1 2 2 3 1 -> 1
39: 1 1 2 2 4 1 -> 1
42: 1 1 2 3 1 2 -> 1
50: 1 2 1 1 1 2 -> 1
51: 1 2 1 1 2 1 -> 0
53: 1 2 1 1 3 1 -> 0
56: 1 2 1 1 4 2 -> 0
57: 1 2 1 2 1 1 -> 1
61: 1 2 1 2 3 1 -> 0
62: 1 2 1 2 3 2 -> 0
64: 1 2 1 2 4 2 -> 0
67: 1 2 1 3 2 1 -> 0
72: 1 2 1 3 4 2 -> 0
76: 1 2 2 1 2 2 -> 0
86: 1 2 2 2 3 2 -> 0
87: 1 2 2 2 4 1 -> 0
88: 1 2 2 2 4 2 -> 0
92: 1 2 2 3 2 2 -> 0
93: 1 2 2 3 3 1 -> 0
94: 1 2 2 3 3 2 -> 0
99: 1 3 1 1 2 1 -> 0
103: 1 3 1 1 4 1 -> 0
107: 1 3 1 2 2 1 -> 0
111: 1 3 1 2 4 1 -> 0
114: 1 3 1 3 1 2 -> 1
116: 1 3 1 3 2 2 -> 0
117: 1 3 1 3 3 1 -> 0
119: 1 3 1 3 4 1 -> 0
120: 1 3 1 3 4 2 -> 0
124: 1 3 2 1 2 2 -> 0
130: 1 3 2 2 1 2 -> 1
132: 1 3 2 2 2 2 -> 0
134: 1 3 2 2 3 2 -> 0
135: 1 3 2 2 4 1 -> 0
136: 1 3 2 2 4 2 -> 0
137: 1 3 2 3 1 1 -> 1
139: 1 3 2 3 2 1 -> 0
143: 1 3 2 3 4 1 -> 0
144: 1 3 2 3 4 2 -> 0
149: 2 1 1 1 3 1 -> 0
150: 2 1 1 1 3 2 -> 0
153: 2 1 1 2 1 1 -> 1
154: 2 1 1 2 1 2 -> 1
156: 2 1 1 2 2 2 -> 0
157: 2 1 1 2 3 1 -> 0
159: 2 1 1 2 4 1 -> 0
160: 2 1 1 2 4 2 -> 0
167: 2 1 1 3 4 1 -> 0
172: 2 1 2 1 2 2 -> 0
173: 2 1 2 1 3 1 -> 0
176: 2 1 2 1 4 2 -> 0
181: 2 1 2 2 3 1 -> 0
184: 2 1 2 2 4 2 -> 0
188: 2 1 2 3 2 2 -> 0
191: 2 1 2 3 4 1 -> 0
195: 2 2 1 1 2 1 -> 1
196: 2 2 1 1 2 2 -> 1
197: 2 2 1 1 3 1 -> 1
206: 2 2 1 2 3 2 -> 1
209: 2 2 1 3 1 1 -> 1
210: 2 2 1 3 1 2 -> 1
212: 2 2 1 3 2 2 -> 1
214: 2 2 1 3 3 2 -> 1
216: 2 2 1 3 4 2 -> 1
217: 2 2 2 1 1 1 -> 1
222: 2 2 2 1 3 2 -> 1
223: 2 2 2 1 4 1 -> 1
224: 2 2 2 1 4 2 -> 1
227: 2 2 2 2 2 1 -> 1
239: 2 2 2 3 4 1 -> 1
241: 2 3 1 1 1 1 -> 1
249: 2 3 1 2 1 1 -> 1
253: 2 3 1 2 3 1 -> 0
258: 2 3 1 3 1 2 -> 1
261: 2 3 1 3 3 1 -> 0
264: 2 3 1 3 4 2 -> 0
270: 2 3 2 1 3 2 -> 0
273: 2 3 2 2 1 1 -> 1
274: 2 3 2 2 1 2 -> 1
275: 2 3 2 2 2 1 -> 0
286: 2 3 2 3 3 2 -> 0
289: 3 1 1 1 1 1 -> 1
290: 3 1 1 1 1 2 -> 1
297: 3 1 1 2 1 1 -> 1
300: 3 1 1 2 2 2 -> 0
308: 3 1 1 3 2 2 -> 0
313: 3 1 2 1 1 1 -> 1
316: 3 1 2 1 2 2 -> 0
324: 3 1 2 2 2 2 -> 0
326: 3 1 2 2 3 2 -> 0
332: 3 1 2 3 2 2 -> 0
337: 3 2 1 1 1 1 -> 1
344: 3 2 1 1 4 2 -> 0
346: 3 2 1 2 1 2 -> 1
352: 3 2 1 2 4 2 -> 0
361: 3 2 2 1 1 1 -> 1
362: 3 2 2 1 1 2 -> 1
366: 3 2 2 1 3 2 -> 0
377: 3 2 2 3 1 1 -> 1
379: 3 2 2 3 2 1 -> 0
383: 3 2 2 3 4 1 -> 0
385: 3 3 1 1 1 1 -> 1
387: 3 3 1 1 2 1 -> 1
392: 3 3 1 1 4 2 -> 1
398: 3 3 1 2 3 2 -> 1
400: 3 3 1 2 4 2 -> 1
402: 3 3 1 3 1 2 -> 1
403: 3 3 1 3 2 1 -> 1
404: 3 3 1 3 2 2 -> 1
408: 3 3 1 3 4 2 -> 1
409: 3 3 2 1 1 1 -> 1
414: 3 3 2 1 3 2 -> 1
415: 3 3 2 1 4 1 -> 1
416: 3 3 2 1 4 2 -> 1
426: 3 3 2 3 1 2 -> 1
428: 3 3 2 3 2 2 -> 1
430: 3 3 2 3 3 2 -> 1
432: 3 3 2 3 4 2 -> 1

Testing Data Set #1

1: 1 1 1 1 1 1 -> 1
2: 1 1 1 1 1 2 -> 1
3: 1 1 1 1 2 1 -> 1
4: 1 1 1 1 2 2 -> 1
5: 1 1 1 1 3 1 -> 1
6: 1 1 1 1 3 2 -> 1
7: 1 1 1 1 4 1 -> 1
8: 1 1 1 1 4 2 -> 1
9: 1 1 1 2 1 1 -> 1
10: 1 1 1 2 1 2 -> 1
11: 1 1 1 2 2 1 -> 1
12: 1 1 1 2 2 2 -> 1
13: 1 1 1 2 3 1 -> 1
14: 1 1 1 2 3 2 -> 1
15: 1 1 1 2 4 1 -> 1
16: 1 1 1 2 4 2 -> 1
17: 1 1 1 3 1 1 -> 1
18: 1 1 1 3 1 2 -> 1
19: 1 1 1 3 2 1 -> 1
20: 1 1 1 3 2 2 -> 1
21: 1 1 1 3 3 1 -> 1
22: 1 1 1 3 3 2 -> 1
23: 1 1 1 3 4 1 -> 1
24: 1 1 1 3 4 2 -> 1
25: 1 1 2 1 1 1 -> 1
26: 1 1 2 1 1 2 -> 1
27: 1 1 2 1 2 1 -> 1
28: 1 1 2 1 2 2 -> 1
29: 1 1 2 1 3 1 -> 1
30: 1 1 2 1 3 2 -> 1
31: 1 1 2 1 4 1 -> 1
32: 1 1 2 1 4 2 -> 1
33: 1 1 2 2 1 1 -> 1
34: 1 1 2 2 1 2 -> 1
35: 1 1 2 2 2 1 -> 1
36: 1 1 2 2 2 2 -> 1
37: 1 1 2 2 3 1 -> 1
38: 1 1 2 2 3 2 -> 1
39: 1 1 2 2 4 1 -> 1
40: 1 1 2 2 4 2 -> 1
41: 1 1 2 3 1 1 -> 1
42: 1 1 2 3 1 2 -> 1
43: 1 1 2 3 2 1 -> 1
44: 1 1 2 3 2 2 -> 1
45: 1 1 2 3 3 1 -> 1
46: 1 1 2 3 3 2 -> 1
47: 1 1 2 3 4 1 -> 1
48: 1 1 2 3 4 2 -> 1

49: 1 2 1 1 1 1 -> 1
50: 1 2 1 1 1 2 -> 1
51: 1 2 1 1 2 1 -> 0
52: 1 2 1 1 2 2 -> 0
53: 1 2 1 1 3 1 -> 0
54: 1 2 1 1 3 2 -> 0
55: 1 2 1 1 4 1 -> 0
56: 1 2 1 1 4 2 -> 0
57: 1 2 1 2 1 1 -> 1
58: 1 2 1 2 1 2 -> 1
59: 1 2 1 2 2 1 -> 0
60: 1 2 1 2 2 2 -> 0
61: 1 2 1 2 3 1 -> 0
62: 1 2 1 2 3 2 -> 0
63: 1 2 1 2 4 1 -> 0
64: 1 2 1 2 4 2 -> 0
65: 1 2 1 3 1 1 -> 1
66: 1 2 1 3 1 2 -> 1
67: 1 2 1 3 2 1 -> 0
68: 1 2 1 3 2 2 -> 0
69: 1 2 1 3 3 1 -> 0
70: 1 2 1 3 3 2 -> 0
71: 1 2 1 3 4 1 -> 0
72: 1 2 1 3 4 2 -> 0
73: 1 2 2 1 1 1 -> 1
74: 1 2 2 1 1 2 -> 1
75: 1 2 2 1 2 1 -> 0
76: 1 2 2 1 2 2 -> 0
77: 1 2 2 1 3 1 -> 0
78: 1 2 2 1 3 2 -> 0
79: 1 2 2 1 4 1 -> 0
80: 1 2 2 1 4 2 -> 0
81: 1 2 2 2 1 1 -> 1
82: 1 2 2 2 1 2 -> 1
83: 1 2 2 2 2 1 -> 0
84: 1 2 2 2 2 2 -> 0
85: 1 2 2 2 3 1 -> 0
86: 1 2 2 2 3 2 -> 0
87: 1 2 2 2 4 1 -> 0
88: 1 2 2 2 4 2 -> 0
89: 1 2 2 3 1 1 -> 1
90: 1 2 2 3 1 2 -> 1
91: 1 2 2 3 2 1 -> 0
92: 1 2 2 3 2 2 -> 0
93: 1 2 2 3 3 1 -> 0
94: 1 2 2 3 3 2 -> 0
95: 1 2 2 3 4 1 -> 0
96: 1 2 2 3 4 2 -> 0
97: 1 3 1 1 1 1 -> 1
98: 1 3 1 1 1 2 -> 1
99: 1 3 1 1 2 1 -> 0
100: 1 3 1 1 2 2 -> 0
101: 1 3 1 1 3 1 -> 0
102: 1 3 1 1 3 2 -> 0
103: 1 3 1 1 4 1 -> 0
104: 1 3 1 1 4 2 -> 0
105: 1 3 1 2 1 1 -> 1
106: 1 3 1 2 1 2 -> 1
107: 1 3 1 2 2 1 -> 0
108: 1 3 1 2 2 2 -> 0
109: 1 3 1 2 3 1 -> 0
110: 1 3 1 2 3 2 -> 0
111: 1 3 1 2 4 1 -> 0
112: 1 3 1 2 4 2 -> 0
113: 1 3 1 3 1 1 -> 1
114: 1 3 1 3 1 2 -> 1
115: 1 3 1 3 2 1 -> 0
116: 1 3 1 3 2 2 -> 0
117: 1 3 1 3 3 1 -> 0
118: 1 3 1 3 3 2 -> 0
119: 1 3 1 3 4 1 -> 0
120: 1 3 1 3 4 2 -> 0
121: 1 3 2 1 1 1 -> 1
122: 1 3 2 1 1 2 -> 1
123: 1 3 2 1 2 1 -> 0
124: 1 3 2 1 2 2 -> 0
125: 1 3 2 1 3 1 -> 0
126: 1 3 2 1 3 2 -> 0
127: 1 3 2 1 4 1 -> 0
128: 1 3 2 1 4 2 -> 0
129: 1 3 2 2 1 1 -> 1
130: 1 3 2 2 1 2 -> 1
131: 1 3 2 2 2 1 -> 0
132: 1 3 2 2 2 2 -> 0
133: 1 3 2 2 3 1 -> 0
134: 1 3 2 2 3 2 -> 0
135: 1 3 2 2 4 1 -> 0
136: 1 3 2 2 4 2 -> 0
137: 1 3 2 3 1 1 -> 1
138: 1 3 2 3 1 2 -> 1
139: 1 3 2 3 2 1 -> 0
140: 1 3 2 3 2 2 -> 0
141: 1 3 2 3 3 1 -> 0
142: 1 3 2 3 3 2 -> 0
143: 1 3 2 3 4 1 -> 0
144: 1 3 2 3 4 2 -> 0
145: 2 1 1 1 1 1 -> 1
146: 2 1 1 1 1 2 -> 1
147: 2 1 1 1 2 1 -> 0
148: 2 1 1 1 2 2 -> 0
149: 2 1 1 1 3 1 -> 0
150: 2 1 1 1 3 2 -> 0
151: 2 1 1 1 4 1 -> 0
152: 2 1 1 1 4 2 -> 0
153: 2 1 1 2 1 1 -> 1
154: 2 1 1 2 1 2 -> 1
155: 2 1 1 2 2 1 -> 0
156: 2 1 1 2 2 2 -> 0
157: 2 1 1 2 3 1 -> 0
158: 2 1 1 2 3 2 -> 0
159: 2 1 1 2 4 1 -> 0
160: 2 1 1 2 4 2 -> 0
161: 2 1 1 3 1 1 -> 1
162: 2 1 1 3 1 2 -> 1
163: 2 1 1 3 2 1 -> 0
164: 2 1 1 3 2 2 -> 0
165: 2 1 1 3 3 1 -> 0
166: 2 1 1 3 3 2 -> 0
167: 2 1 1 3 4 1 -> 0
168: 2 1 1 3 4 2 -> 0
169: 2 1 2 1 1 1 -> 1
170: 2 1 2 1 1 2 -> 1
171: 2 1 2 1 2 1 -> 0
172: 2 1 2 1 2 2 -> 0
173: 2 1 2 1 3 1 -> 0
174: 2 1 2 1 3 2 -> 0
175: 2 1 2 1 4 1 -> 0
176: 2 1 2 1 4 2 -> 0
177: 2 1 2 2 1 1 -> 1
178: 2 1 2 2 1 2 -> 1
179: 2 1 2 2 2 1 -> 0
180: 2 1 2 2 2 2 -> 0
181: 2 1 2 2 3 1 -> 0
182: 2 1 2 2 3 2 -> 0
183: 2 1 2 2 4 1 -> 0
184: 2 1 2 2 4 2 -> 0
185: 2 1 2 3 1 1 -> 1
186: 2 1 2 3 1 2 -> 1
187: 2 1 2 3 2 1 -> 0
188: 2 1 2 3 2 2 -> 0
189: 2 1 2 3 3 1 -> 0
190: 2 1 2 3 3 2 -> 0
191: 2 1 2 3 4 1 -> 0
192: 2 1 2 3 4 2 -> 0
193: 2 2 1 1 1 1 -> 1
194: 2 2 1 1 1 2 -> 1
195: 2 2 1 1 2 1 -> 1
196: 2 2 1 1 2 2 -> 1
197: 2 2 1 1 3 1 -> 1
198: 2 2 1 1 3 2 -> 1
199: 2 2 1 1 4 1 -> 1
200: 2 2 1 1 4 2 -> 1
201: 2 2 1 2 1 1 -> 1
202: 2 2 1 2 1 2 -> 1
203: 2 2 1 2 2 1 -> 1
204: 2 2 1 2 2 2 -> 1
205: 2 2 1 2 3 1 -> 1
206: 2 2 1 2 3 2 -> 1
207: 2 2 1 2 4 1 -> 1
208: 2 2 1 2 4 2 -> 1
209: 2 2 1 3 1 1 -> 1
210: 2 2 1 3 1 2 -> 1
211: 2 2 1 3 2 1 -> 1
212: 2 2 1 3 2 2 -> 1
213: 2 2 1 3 3 1 -> 1
214: 2 2 1 3 3 2 -> 1
215: 2 2 1 3 4 1 -> 1
216: 2 2 1 3 4 2 -> 1
217: 2 2 2 1 1 1 -> 1
218: 2 2 2 1 1 2 -> 1
219: 2 2 2 1 2 1 -> 1
220: 2 2 2 1 2 2 -> 1
221: 2 2 2 1 3 1 -> 1
222: 2 2 2 1 3 2 -> 1
223: 2 2 2 1 4 1 -> 1
224: 2 2 2 1 4 2 -> 1
225: 2 2 2 2 1 1 -> 1
226: 2 2 2 2 1 2 -> 1
227: 2 2 2 2 2 1 -> 1
228: 2 2 2 2 2 2 -> 1
229: 2 2 2 2 3 1 -> 1
230: 2 2 2 2 3 2 -> 1
231: 2 2 2 2 4 1 -> 1
232: 2 2 2 2 4 2 -> 1
233: 2 2 2 3 1 1 -> 1
234: 2 2 2 3 1 2 -> 1
235: 2 2 2 3 2 1 -> 1
236: 2 2 2 3 2 2 -> 1
237: 2 2 2 3 3 1 -> 1
238: 2 2 2 3 3 2 -> 1
239: 2 2 2 3 4 1 -> 1
240: 2 2 2 3 4 2 -> 1
241: 2 3 1 1 1 1 -> 1
242: 2 3 1 1 1 2 -> 1
243: 2 3 1 1 2 1 -> 0
244: 2 3 1 1 2 2 -> 0
245: 2 3 1 1 3 1 -> 0
246: 2 3 1 1 3 2 -> 0
247: 2 3 1 1 4 1 -> 0
248: 2 3 1 1 4 2 -> 0
249: 2 3 1 2 1 1 -> 1
250: 2 3 1 2 1 2 -> 1
251: 2 3 1 2 2 1 -> 0
252: 2 3 1 2 2 2 -> 0
253: 2 3 1 2 3 1 -> 0
254: 2 3 1 2 3 2 -> 0
255: 2 3 1 2 4 1 -> 0
256: 2 3 1 2 4 2 -> 0
257: 2 3 1 3 1 1 -> 1
258: 2 3 1 3 1 2 -> 1
259: 2 3 1 3 2 1 -> 0
260: 2 3 1 3 2 2 -> 0
261: 2 3 1 3 3 1 -> 0
262: 2 3 1 3 3 2 -> 0
263: 2 3 1 3 4 1 -> 0
264: 2 3 1 3 4 2 -> 0
265: 2 3 2 1 1 1 -> 1
266: 2 3 2 1 1 2 -> 1
267: 2 3 2 1 2 1 -> 0
268: 2 3 2 1 2 2 -> 0
269: 2 3 2 1 3 1 -> 0
270: 2 3 2 1 3 2 -> 0
271: 2 3 2 1 4 1 -> 0
272: 2 3 2 1 4 2 -> 0
273: 2 3 2 2 1 1 -> 1
274: 2 3 2 2 1 2 -> 1
275: 2 3 2 2 2 1 -> 0
276: 2 3 2 2 2 2 -> 0
277: 2 3 2 2 3 1 -> 0
278: 2 3 2 2 3 2 -> 0
279: 2 3 2 2 4 1 -> 0
280: 2 3 2 2 4 2 -> 0
281: 2 3 2 3 1 1 -> 1
282: 2 3 2 3 1 2 -> 1
283: 2 3 2 3 2 1 -> 0
284: 2 3 2 3 2 2 -> 0
285: 2 3 2 3 3 1 -> 0
286: 2 3 2 3 3 2 -> 0
287: 2 3 2 3 4 1 -> 0
288: 2 3 2 3 4 2 -> 0
289: 3 1 1 1 1 1 -> 1
290: 3 1 1 1 1 2 -> 1
291: 3 1 1 1 2 1 -> 0
292: 3 1 1 1 2 2 -> 0
293: 3 1 1 1 3 1 -> 0
294: 3 1 1 1 3 2 -> 0
295: 3 1 1 1 4 1 -> 0
296: 3 1 1 1 4 2 -> 0
297: 3 1 1 2 1 1 -> 1
298: 3 1 1 2 1 2 -> 1
299: 3 1 1 2 2 1 -> 0
300: 3 1 1 2 2 2 -> 0
301: 3 1 1 2 3 1 -> 0
302: 3 1 1 2 3 2 -> 0
303: 3 1 1 2 4 1 -> 0
304: 3 1 1 2 4 2 -> 0

305: 3 1 1 3 1 1 -> 1
306: 3 1 1 3 1 2 -> 1
307: 3 1 1 3 2 1 -> 0
308: 3 1 1 3 2 2 -> 0
309: 3 1 1 3 3 1 -> 0
310: 3 1 1 3 3 2 -> 0
311: 3 1 1 3 4 1 -> 0
312: 3 1 1 3 4 2 -> 0
313: 3 1 2 1 1 1 -> 1
314: 3 1 2 1 1 2 -> 1
315: 3 1 2 1 2 1 -> 0
316: 3 1 2 1 2 2 -> 0
317: 3 1 2 1 3 1 -> 0
318: 3 1 2 1 3 2 -> 0
319: 3 1 2 1 4 1 -> 0
320: 3 1 2 1 4 2 -> 0
321: 3 1 2 2 1 1 -> 1
322: 3 1 2 2 1 2 -> 1
323: 3 1 2 2 2 1 -> 0
324: 3 1 2 2 2 2 -> 0
325: 3 1 2 2 3 1 -> 0
326: 3 1 2 2 3 2 -> 0
327: 3 1 2 2 4 1 -> 0
328: 3 1 2 2 4 2 -> 0
329: 3 1 2 3 1 1 -> 1
330: 3 1 2 3 1 2 -> 1
331: 3 1 2 3 2 1 -> 0
332: 3 1 2 3 2 2 -> 0
333: 3 1 2 3 3 1 -> 0
334: 3 1 2 3 3 2 -> 0
335: 3 1 2 3 4 1 -> 0
336: 3 1 2 3 4 2 -> 0
337: 3 2 1 1 1 1 -> 1
338: 3 2 1 1 1 2 -> 1
339: 3 2 1 1 2 1 -> 0
340: 3 2 1 1 2 2 -> 0
341: 3 2 1 1 3 1 -> 0
342: 3 2 1 1 3 2 -> 0
343: 3 2 1 1 4 1 -> 0
344: 3 2 1 1 4 2 -> 0
345: 3 2 1 2 1 1 -> 1
346: 3 2 1 2 1 2 -> 1
347: 3 2 1 2 2 1 -> 0
348: 3 2 1 2 2 2 -> 0
349: 3 2 1 2 3 1 -> 0
350: 3 2 1 2 3 2 -> 0
351: 3 2 1 2 4 1 -> 0
352: 3 2 1 2 4 2 -> 0
353: 3 2 1 3 1 1 -> 1
354: 3 2 1 3 1 2 -> 1
355: 3 2 1 3 2 1 -> 0
356: 3 2 1 3 2 2 -> 0
357: 3 2 1 3 3 1 -> 0
358: 3 2 1 3 3 2 -> 0
359: 3 2 1 3 4 1 -> 0
360: 3 2 1 3 4 2 -> 0
361: 3 2 2 1 1 1 -> 1
362: 3 2 2 1 1 2 -> 1
363: 3 2 2 1 2 1 -> 0
364: 3 2 2 1 2 2 -> 0
365: 3 2 2 1 3 1 -> 0
366: 3 2 2 1 3 2 -> 0
367: 3 2 2 1 4 1 -> 0
368: 3 2 2 1 4 2 -> 0
369: 3 2 2 2 1 1 -> 1
370: 3 2 2 2 1 2 -> 1
371: 3 2 2 2 2 1 -> 0
372: 3 2 2 2 2 2 -> 0
373: 3 2 2 2 3 1 -> 0
374: 3 2 2 2 3 2 -> 0
375: 3 2 2 2 4 1 -> 0
376: 3 2 2 2 4 2 -> 0
377: 3 2 2 3 1 1 -> 1
378: 3 2 2 3 1 2 -> 1
379: 3 2 2 3 2 1 -> 0
380: 3 2 2 3 2 2 -> 0
381: 3 2 2 3 3 1 -> 0
382: 3 2 2 3 3 2 -> 0
383: 3 2 2 3 4 1 -> 0
384: 3 2 2 3 4 2 -> 0
385: 3 3 1 1 1 1 -> 1
386: 3 3 1 1 1 2 -> 1
387: 3 3 1 1 2 1 -> 1
388: 3 3 1 1 2 2 -> 1
389: 3 3 1 1 3 1 -> 1
390: 3 3 1 1 3 2 -> 1
391: 3 3 1 1 4 1 -> 1
392: 3 3 1 1 4 2 -> 1
393: 3 3 1 2 1 1 -> 1
394: 3 3 1 2 1 2 -> 1
395: 3 3 1 2 2 1 -> 1
396: 3 3 1 2 2 2 -> 1
397: 3 3 1 2 3 1 -> 1
398: 3 3 1 2 3 2 -> 1
399: 3 3 1 2 4 1 -> 1
400: 3 3 1 2 4 2 -> 1
401: 3 3 1 3 1 1 -> 1
402: 3 3 1 3 1 2 -> 1
403: 3 3 1 3 2 1 -> 1
404: 3 3 1 3 2 2 -> 1
405: 3 3 1 3 3 1 -> 1
406: 3 3 1 3 3 2 -> 1
407: 3 3 1 3 4 1 -> 1
408: 3 3 1 3 4 2 -> 1
409: 3 3 2 1 1 1 -> 1
410: 3 3 2 1 1 2 -> 1
411: 3 3 2 1 2 1 -> 1
412: 3 3 2 1 2 2 -> 1
413: 3 3 2 1 3 1 -> 1
414: 3 3 2 1 3 2 -> 1
415: 3 3 2 1 4 1 -> 1
416: 3 3 2 1 4 2 -> 1
417: 3 3 2 2 1 1 -> 1
418: 3 3 2 2 1 2 -> 1
419: 3 3 2 2 2 1 -> 1
420: 3 3 2 2 2 2 -> 1
421: 3 3 2 2 3 1 -> 1
422: 3 3 2 2 3 2 -> 1
423: 3 3 2 2 4 1 -> 1
424: 3 3 2 2 4 2 -> 1
425: 3 3 2 3 1 1 -> 1
426: 3 3 2 3 1 2 -> 1
427: 3 3 2 3 2 1 -> 1
428: 3 3 2 3 2 2 -> 1
429: 3 3 2 3 3 1 -> 1
430: 3 3 2 3 3 2 -> 1
431: 3 3 2 3 4 1 -> 1
432: 3 3 2 3 4 2 -> 1

Training Data Set #2

The training set of examples for the second problem:
Exactly two of the six attributes have their first value

4: 1 1 1 1 2 2 -> 0
7: 1 1 1 1 4 1 -> 0
9: 1 1 1 2 1 1 -> 0
10: 1 1 1 2 1 2 -> 0
11: 1 1 1 2 2 1 -> 0
13: 1 1 1 2 3 1 -> 0
15: 1 1 1 2 4 1 -> 0
19: 1 1 1 3 2 1 -> 0
23: 1 1 1 3 4 1 -> 0
25: 1 1 2 1 1 1 -> 0
26: 1 1 2 1 1 2 -> 0
37: 1 1 2 2 3 1 -> 0
39: 1 1 2 2 4 1 -> 0
40: 1 1 2 2 4 2 -> 1
42: 1 1 2 3 1 2 -> 0
44: 1 1 2 3 2 2 -> 1
50: 1 2 1 1 1 2 -> 0
58: 1 2 1 2 1 2 -> 0
60: 1 2 1 2 2 2 -> 1
61: 1 2 1 2 3 1 -> 0
62: 1 2 1 2 3 2 -> 1
63: 1 2 1 2 4 1 -> 0
65: 1 2 1 3 1 1 -> 0
66: 1 2 1 3 1 2 -> 0
68: 1 2 1 3 2 2 -> 1
69: 1 2 1 3 3 1 -> 0
70: 1 2 1 3 3 2 -> 1
71: 1 2 1 3 4 1 -> 0
72: 1 2 1 3 4 2 -> 1
75: 1 2 2 1 2 1 -> 0
79: 1 2 2 1 4 1 -> 0
85: 1 2 2 2 3 1 -> 1
87: 1 2 2 2 4 1 -> 1
89: 1 2 2 3 1 1 -> 0
90: 1 2 2 3 1 2 -> 1
93: 1 2 2 3 3 1 -> 1
94: 1 2 2 3 3 2 -> 0
95: 1 2 2 3 4 1 -> 1
96: 1 2 2 3 4 2 -> 0
98: 1 3 1 1 1 2 -> 0
100: 1 3 1 1 2 2 -> 0
101: 1 3 1 1 3 1 -> 0
102: 1 3 1 1 3 2 -> 0
107: 1 3 1 2 2 1 -> 0
108: 1 3 1 2 2 2 -> 1
110: 1 3 1 2 3 2 -> 1
111: 1 3 1 2 4 1 -> 0
116: 1 3 1 3 2 2 -> 1
117: 1 3 1 3 3 1 -> 0
120: 1 3 1 3 4 2 -> 1
125: 1 3 2 1 3 1 -> 0
126: 1 3 2 1 3 2 -> 1
127: 1 3 2 1 4 1 -> 0
130: 1 3 2 2 1 2 -> 1
134: 1 3 2 2 3 2 -> 0
136: 1 3 2 2 4 2 -> 0
139: 1 3 2 3 2 1 -> 1
145: 2 1 1 1 1 1 -> 0
148: 2 1 1 1 2 2 -> 0
149: 2 1 1 1 3 1 -> 0
156: 2 1 1 2 2 2 -> 1
162: 2 1 1 3 1 2 -> 0
164: 2 1 1 3 2 2 -> 1
166: 2 1 1 3 3 2 -> 1
167: 2 1 1 3 4 1 -> 0
169: 2 1 2 1 1 1 -> 0
172: 2 1 2 1 2 2 -> 1
175: 2 1 2 1 4 1 -> 0
179: 2 1 2 2 2 1 -> 1
184: 2 1 2 2 4 2 -> 0
185: 2 1 2 3 1 1 -> 0
186: 2 1 2 3 1 2 -> 1
188: 2 1 2 3 2 2 -> 0
190: 2 1 2 3 3 2 -> 0
192: 2 1 2 3 4 2 -> 0
197: 2 2 1 1 3 1 -> 0
200: 2 2 1 1 4 2 -> 1
201: 2 2 1 2 1 1 -> 0
205: 2 2 1 2 3 1 -> 1
213: 2 2 1 3 3 1 -> 1
214: 2 2 1 3 3 2 -> 0
215: 2 2 1 3 4 1 -> 1
217: 2 2 2 1 1 1 -> 0
220: 2 2 2 1 2 2 -> 0
222: 2 2 2 1 3 2 -> 0
223: 2 2 2 1 4 1 -> 1
224: 2 2 2 1 4 2 -> 0
225: 2 2 2 2 1 1 -> 1
228: 2 2 2 2 2 2 -> 0
229: 2 2 2 2 3 1 -> 0
233: 2 2 2 3 1 1 -> 1
235: 2 2 2 3 2 1 -> 0
236: 2 2 2 3 2 2 -> 0
240: 2 2 2 3 4 2 -> 0
241: 2 3 1 1 1 1 -> 0
242: 2 3 1 1 1 2 -> 0
246: 2 3 1 1 3 2 -> 1
249: 2 3 1 2 1 1 -> 0
253: 2 3 1 2 3 1 -> 1
254: 2 3 1 2 3 2 -> 0
256: 2 3 1 2 4 2 -> 0
258: 2 3 1 3 1 2 -> 1
259: 2 3 1 3 2 1 -> 1
263: 2 3 1 3 4 1 -> 1
266: 2 3 2 1 1 2 -> 1
267: 2 3 2 1 2 1 -> 1
269: 2 3 2 1 3 1 -> 1
272: 2 3 2 1 4 2 -> 0

273: 2 3 2 2 1 1 -> 1
275: 2 3 2 2 2 1 -> 0
278: 2 3 2 2 3 2 -> 0
285: 2 3 2 3 3 1 -> 0
286: 2 3 2 3 3 2 -> 0
288: 2 3 2 3 4 2 -> 0
295: 3 1 1 1 4 1 -> 0
298: 3 1 1 2 1 2 -> 0
300: 3 1 1 2 2 2 -> 1
302: 3 1 1 2 3 2 -> 1
303: 3 1 1 2 4 1 -> 0
304: 3 1 1 2 4 2 -> 1
305: 3 1 1 3 1 1 -> 0
306: 3 1 1 3 1 2 -> 0
308: 3 1 1 3 2 2 -> 1
310: 3 1 1 3 3 2 -> 1
313: 3 1 2 1 1 1 -> 0
316: 3 1 2 1 2 2 -> 1
317: 3 1 2 1 3 1 -> 0
318: 3 1 2 1 3 2 -> 1
319: 3 1 2 1 4 1 -> 0
320: 3 1 2 1 4 2 -> 1
323: 3 1 2 2 2 1 -> 1
330: 3 1 2 3 1 2 -> 1
331: 3 1 2 3 2 1 -> 1
332: 3 1 2 3 2 2 -> 0
336: 3 1 2 3 4 2 -> 0
338: 3 2 1 1 1 2 -> 0
340: 3 2 1 1 2 2 -> 1
341: 3 2 1 1 3 1 -> 0
342: 3 2 1 1 3 2 -> 1
346: 3 2 1 2 1 2 -> 1
347: 3 2 1 2 2 1 -> 1
353: 3 2 1 3 1 1 -> 0
355: 3 2 1 3 2 1 -> 1
357: 3 2 1 3 3 1 -> 1
358: 3 2 1 3 3 2 -> 0
361: 3 2 2 1 1 1 -> 0
364: 3 2 2 1 2 2 -> 0
365: 3 2 2 1 3 1 -> 1
366: 3 2 2 1 3 2 -> 0
369: 3 2 2 2 1 1 -> 1
371: 3 2 2 2 2 1 -> 0
372: 3 2 2 2 2 2 -> 0
374: 3 2 2 2 3 2 -> 0
377: 3 2 2 3 1 1 -> 1
382: 3 2 2 3 3 2 -> 0
384: 3 2 2 3 4 2 -> 0
385: 3 3 1 1 1 1 -> 0
387: 3 3 1 1 2 1 -> 0
389: 3 3 1 1 3 1 -> 0
390: 3 3 1 1 3 2 -> 1
398: 3 3 1 2 3 2 -> 0
409: 3 3 2 1 1 1 -> 0
417: 3 3 2 2 1 1 -> 1
419: 3 3 2 2 2 1 -> 0
421: 3 3 2 2 3 1 -> 0
422: 3 3 2 2 3 2 -> 0
425: 3 3 2 3 1 1 -> 1
427: 3 3 2 3 2 1 -> 0
432: 3 3 2 3 4 2 -> 0

Testing Data Set #2

1: 1 1 1 1 1 1 -> 0
2: 1 1 1 1 1 2 -> 0
3: 1 1 1 1 2 1 -> 0
4: 1 1 1 1 2 2 -> 0
5: 1 1 1 1 3 1 -> 0
6: 1 1 1 1 3 2 -> 0
7: 1 1 1 1 4 1 -> 0
8: 1 1 1 1 4 2 -> 0
9: 1 1 1 2 1 1 -> 0
10: 1 1 1 2 1 2 -> 0
11: 1 1 1 2 2 1 -> 0
12: 1 1 1 2 2 2 -> 0
13: 1 1 1 2 3 1 -> 0
14: 1 1 1 2 3 2 -> 0
15: 1 1 1 2 4 1 -> 0
16: 1 1 1 2 4 2 -> 0
17: 1 1 1 3 1 1 -> 0
18: 1 1 1 3 1 2 -> 0
19: 1 1 1 3 2 1 -> 0
20: 1 1 1 3 2 2 -> 0
21: 1 1 1 3 3 1 -> 0
22: 1 1 1 3 3 2 -> 0
23: 1 1 1 3 4 1 -> 0
24: 1 1 1 3 4 2 -> 0
25: 1 1 2 1 1 1 -> 0
26: 1 1 2 1 1 2 -> 0
27: 1 1 2 1 2 1 -> 0
28: 1 1 2 1 2 2 -> 0
29: 1 1 2 1 3 1 -> 0
30: 1 1 2 1 3 2 -> 0
31: 1 1 2 1 4 1 -> 0
32: 1 1 2 1 4 2 -> 0
33: 1 1 2 2 1 1 -> 0
34: 1 1 2 2 1 2 -> 0
35: 1 1 2 2 2 1 -> 0
36: 1 1 2 2 2 2 -> 1
37: 1 1 2 2 3 1 -> 0
38: 1 1 2 2 3 2 -> 1
39: 1 1 2 2 4 1 -> 0
40: 1 1 2 2 4 2 -> 1
41: 1 1 2 3 1 1 -> 0
42: 1 1 2 3 1 2 -> 0
43: 1 1 2 3 2 1 -> 0
44: 1 1 2 3 2 2 -> 1
45: 1 1 2 3 3 1 -> 0
46: 1 1 2 3 3 2 -> 1
47: 1 1 2 3 4 1 -> 0
48: 1 1 2 3 4 2 -> 1
49: 1 2 1 1 1 1 -> 0
50: 1 2 1 1 1 2 -> 0
51: 1 2 1 1 2 1 -> 0
52: 1 2 1 1 2 2 -> 0
53: 1 2 1 1 3 1 -> 0
54: 1 2 1 1 3 2 -> 0
55: 1 2 1 1 4 1 -> 0
56: 1 2 1 1 4 2 -> 0
57: 1 2 1 2 1 1 -> 0
58: 1 2 1 2 1 2 -> 0
59: 1 2 1 2 2 1 -> 0
60: 1 2 1 2 2 2 -> 1
61: 1 2 1 2 3 1 -> 0
62: 1 2 1 2 3 2 -> 1
63: 1 2 1 2 4 1 -> 0
64: 1 2 1 2 4 2 -> 1
65: 1 2 1 3 1 1 -> 0
66: 1 2 1 3 1 2 -> 0
67: 1 2 1 3 2 1 -> 0
68: 1 2 1 3 2 2 -> 1
69: 1 2 1 3 3 1 -> 0
70: 1 2 1 3 3 2 -> 1
71: 1 2 1 3 4 1 -> 0
72: 1 2 1 3 4 2 -> 1
73: 1 2 2 1 1 1 -> 0
74: 1 2 2 1 1 2 -> 0
75: 1 2 2 1 2 1 -> 0
76: 1 2 2 1 2 2 -> 1
77: 1 2 2 1 3 1 -> 0
78: 1 2 2 1 3 2 -> 1
79: 1 2 2 1 4 1 -> 0
80: 1 2 2 1 4 2 -> 1
81: 1 2 2 2 1 1 -> 0
82: 1 2 2 2 1 2 -> 1
83: 1 2 2 2 2 1 -> 1
84: 1 2 2 2 2 2 -> 0
85: 1 2 2 2 3 1 -> 1
86: 1 2 2 2 3 2 -> 0
87: 1 2 2 2 4 1 -> 1
88: 1 2 2 2 4 2 -> 0
89: 1 2 2 3 1 1 -> 0
90: 1 2 2 3 1 2 -> 1
91: 1 2 2 3 2 1 -> 1
92: 1 2 2 3 2 2 -> 0
93: 1 2 2 3 3 1 -> 1
94: 1 2 2 3 3 2 -> 0
95: 1 2 2 3 4 1 -> 1
96: 1 2 2 3 4 2 -> 0
97: 1 3 1 1 1 1 -> 0
98: 1 3 1 1 1 2 -> 0
99: 1 3 1 1 2 1 -> 0
100: 1 3 1 1 2 2 -> 0
101: 1 3 1 1 3 1 -> 0
102: 1 3 1 1 3 2 -> 0
103: 1 3 1 1 4 1 -> 0
104: 1 3 1 1 4 2 -> 0
105: 1 3 1 2 1 1 -> 0
106: 1 3 1 2 1 2 -> 0
107: 1 3 1 2 2 1 -> 0
108: 1 3 1 2 2 2 -> 1
109: 1 3 1 2 3 1 -> 0
110: 1 3 1 2 3 2 -> 1
111: 1 3 1 2 4 1 -> 0
112: 1 3 1 2 4 2 -> 1
113: 1 3 1 3 1 1 -> 0
114: 1 3 1 3 1 2 -> 0
115: 1 3 1 3 2 1 -> 0
116: 1 3 1 3 2 2 -> 1
117: 1 3 1 3 3 1 -> 0
118: 1 3 1 3 3 2 -> 1
119: 1 3 1 3 4 1 -> 0
120: 1 3 1 3 4 2 -> 1
121: 1 3 2 1 1 1 -> 0
122: 1 3 2 1 1 2 -> 0
123: 1 3 2 1 2 1 -> 0
124: 1 3 2 1 2 2 -> 1
125: 1 3 2 1 3 1 -> 0
126: 1 3 2 1 3 2 -> 1
127: 1 3 2 1 4 1 -> 0
128: 1 3 2 1 4 2 -> 1
129: 1 3 2 2 1 1 -> 0
130: 1 3 2 2 1 2 -> 1
131: 1 3 2 2 2 1 -> 1
132: 1 3 2 2 2 2 -> 0
133: 1 3 2 2 3 1 -> 1
134: 1 3 2 2 3 2 -> 0
135: 1 3 2 2 4 1 -> 1
136: 1 3 2 2 4 2 -> 0
137: 1 3 2 3 1 1 -> 0
138: 1 3 2 3 1 2 -> 1
139: 1 3 2 3 2 1 -> 1
140: 1 3 2 3 2 2 -> 0
141: 1 3 2 3 3 1 -> 1
142: 1 3 2 3 3 2 -> 0
143: 1 3 2 3 4 1 -> 1
144: 1 3 2 3 4 2 -> 0
145: 2 1 1 1 1 1 -> 0
146: 2 1 1 1 1 2 -> 0
147: 2 1 1 1 2 1 -> 0
148: 2 1 1 1 2 2 -> 0
149: 2 1 1 1 3 1 -> 0
150: 2 1 1 1 3 2 -> 0
151: 2 1 1 1 4 1 -> 0
152: 2 1 1 1 4 2 -> 0
153: 2 1 1 2 1 1 -> 0
154: 2 1 1 2 1 2 -> 0
155: 2 1 1 2 2 1 -> 0
156: 2 1 1 2 2 2 -> 1
157: 2 1 1 2 3 1 -> 0
158: 2 1 1 2 3 2 -> 1
159: 2 1 1 2 4 1 -> 0
160: 2 1 1 2 4 2 -> 1
161: 2 1 1 3 1 1 -> 0
162: 2 1 1 3 1 2 -> 0
163: 2 1 1 3 2 1 -> 0
164: 2 1 1 3 2 2 -> 1
165: 2 1 1 3 3 1 -> 0
166: 2 1 1 3 3 2 -> 1
167: 2 1 1 3 4 1 -> 0
168: 2 1 1 3 4 2 -> 1
169: 2 1 2 1 1 1 -> 0
170: 2 1 2 1 1 2 -> 0
171: 2 1 2 1 2 1 -> 0
172: 2 1 2 1 2 2 -> 1
173: 2 1 2 1 3 1 -> 0
174: 2 1 2 1 3 2 -> 1
175: 2 1 2 1 4 1 -> 0
176: 2 1 2 1 4 2 -> 1
177: 2 1 2 2 1 1 -> 0
178: 2 1 2 2 1 2 -> 1
179: 2 1 2 2 2 1 -> 1
180: 2 1 2 2 2 2 -> 0
181: 2 1 2 2 3 1 -> 1
182: 2 1 2 2 3 2 -> 0
183: 2 1 2 2 4 1 -> 1
184: 2 1 2 2 4 2 -> 0

185: 2 1 2 3 1 1 -> 0
186: 2 1 2 3 1 2 -> 1
187: 2 1 2 3 2 1 -> 1
188: 2 1 2 3 2 2 -> 0
189: 2 1 2 3 3 1 -> 1
190: 2 1 2 3 3 2 -> 0
191: 2 1 2 3 4 1 -> 1
192: 2 1 2 3 4 2 -> 0
193: 2 2 1 1 1 1 -> 0
194: 2 2 1 1 1 2 -> 0
195: 2 2 1 1 2 1 -> 0
196: 2 2 1 1 2 2 -> 1
197: 2 2 1 1 3 1 -> 0
198: 2 2 1 1 3 2 -> 1
199: 2 2 1 1 4 1 -> 0
200: 2 2 1 1 4 2 -> 1
201: 2 2 1 2 1 1 -> 0
202: 2 2 1 2 1 2 -> 1
203: 2 2 1 2 2 1 -> 1
204: 2 2 1 2 2 2 -> 0
205: 2 2 1 2 3 1 -> 1
206: 2 2 1 2 3 2 -> 0
207: 2 2 1 2 4 1 -> 1
208: 2 2 1 2 4 2 -> 0
209: 2 2 1 3 1 1 -> 0
210: 2 2 1 3 1 2 -> 1
211: 2 2 1 3 2 1 -> 1
212: 2 2 1 3 2 2 -> 0
213: 2 2 1 3 3 1 -> 1
214: 2 2 1 3 3 2 -> 0
215: 2 2 1 3 4 1 -> 1
216: 2 2 1 3 4 2 -> 0
217: 2 2 2 1 1 1 -> 0
218: 2 2 2 1 1 2 -> 1
219: 2 2 2 1 2 1 -> 1
220: 2 2 2 1 2 2 -> 0
221: 2 2 2 1 3 1 -> 1
222: 2 2 2 1 3 2 -> 0
223: 2 2 2 1 4 1 -> 1
224: 2 2 2 1 4 2 -> 0
225: 2 2 2 2 1 1 -> 1
226: 2 2 2 2 1 2 -> 0
227: 2 2 2 2 2 1 -> 0
228: 2 2 2 2 2 2 -> 0
229: 2 2 2 2 3 1 -> 0
230: 2 2 2 2 3 2 -> 0
231: 2 2 2 2 4 1 -> 0
232: 2 2 2 2 4 2 -> 0
233: 2 2 2 3 1 1 -> 1
234: 2 2 2 3 1 2 -> 0
235: 2 2 2 3 2 1 -> 0
236: 2 2 2 3 2 2 -> 0
237: 2 2 2 3 3 1 -> 0
238: 2 2 2 3 3 2 -> 0
239: 2 2 2 3 4 1 -> 0
240: 2 2 2 3 4 2 -> 0
241: 2 3 1 1 1 1 -> 0
242: 2 3 1 1 1 2 -> 0
243: 2 3 1 1 2 1 -> 0
244: 2 3 1 1 2 2 -> 1
245: 2 3 1 1 3 1 -> 0
246: 2 3 1 1 3 2 -> 1
247: 2 3 1 1 4 1 -> 0
248: 2 3 1 1 4 2 -> 1
249: 2 3 1 2 1 1 -> 0
250: 2 3 1 2 1 2 -> 1
251: 2 3 1 2 2 1 -> 1
252: 2 3 1 2 2 2 -> 0
253: 2 3 1 2 3 1 -> 1
254: 2 3 1 2 3 2 -> 0
255: 2 3 1 2 4 1 -> 1
256: 2 3 1 2 4 2 -> 0
257: 2 3 1 3 1 1 -> 0
258: 2 3 1 3 1 2 -> 1
259: 2 3 1 3 2 1 -> 1
260: 2 3 1 3 2 2 -> 0
261: 2 3 1 3 3 1 -> 1
262: 2 3 1 3 3 2 -> 0
263: 2 3 1 3 4 1 -> 1
264: 2 3 1 3 4 2 -> 0
265: 2 3 2 1 1 1 -> 0
266: 2 3 2 1 1 2 -> 1
267: 2 3 2 1 2 1 -> 1
268: 2 3 2 1 2 2 -> 0
269: 2 3 2 1 3 1 -> 1
270: 2 3 2 1 3 2 -> 0
271: 2 3 2 1 4 1 -> 1
272: 2 3 2 1 4 2 -> 0
273: 2 3 2 2 1 1 -> 1
274: 2 3 2 2 1 2 -> 0
275: 2 3 2 2 2 1 -> 0
276: 2 3 2 2 2 2 -> 0
277: 2 3 2 2 3 1 -> 0
278: 2 3 2 2 3 2 -> 0
279: 2 3 2 2 4 1 -> 0
280: 2 3 2 2 4 2 -> 0
281: 2 3 2 3 1 1 -> 1
282: 2 3 2 3 1 2 -> 0
283: 2 3 2 3 2 1 -> 0
284: 2 3 2 3 2 2 -> 0
285: 2 3 2 3 3 1 -> 0
286: 2 3 2 3 3 2 -> 0
287: 2 3 2 3 4 1 -> 0
288: 2 3 2 3 4 2 -> 0
289: 3 1 1 1 1 1 -> 0
290: 3 1 1 1 1 2 -> 0
291: 3 1 1 1 2 1 -> 0
292: 3 1 1 1 2 2 -> 0
293: 3 1 1 1 3 1 -> 0
294: 3 1 1 1 3 2 -> 0
295: 3 1 1 1 4 1 -> 0
296: 3 1 1 1 4 2 -> 0
297: 3 1 1 2 1 1 -> 0
298: 3 1 1 2 1 2 -> 0
299: 3 1 1 2 2 1 -> 0
300: 3 1 1 2 2 2 -> 1
301: 3 1 1 2 3 1 -> 0
302: 3 1 1 2 3 2 -> 1
303: 3 1 1 2 4 1 -> 0
304: 3 1 1 2 4 2 -> 1
305: 3 1 1 3 1 1 -> 0
306: 3 1 1 3 1 2 -> 0
307: 3 1 1 3 2 1 -> 0
308: 3 1 1 3 2 2 -> 1
309: 3 1 1 3 3 1 -> 0
310: 3 1 1 3 3 2 -> 1
311: 3 1 1 3 4 1 -> 0
312: 3 1 1 3 4 2 -> 1
313: 3 1 2 1 1 1 -> 0
314: 3 1 2 1 1 2 -> 0
315: 3 1 2 1 2 1 -> 0
316: 3 1 2 1 2 2 -> 1
317: 3 1 2 1 3 1 -> 0
318: 3 1 2 1 3 2 -> 1
319: 3 1 2 1 4 1 -> 0
320: 3 1 2 1 4 2 -> 1
321: 3 1 2 2 1 1 -> 0
322: 3 1 2 2 1 2 -> 1
323: 3 1 2 2 2 1 -> 1
324: 3 1 2 2 2 2 -> 0
325: 3 1 2 2 3 1 -> 1
326: 3 1 2 2 3 2 -> 0
327: 3 1 2 2 4 1 -> 1
328: 3 1 2 2 4 2 -> 0
329: 3 1 2 3 1 1 -> 0
330: 3 1 2 3 1 2 -> 1
331: 3 1 2 3 2 1 -> 1
332: 3 1 2 3 2 2 -> 0
333: 3 1 2 3 3 1 -> 1
334: 3 1 2 3 3 2 -> 0
335: 3 1 2 3 4 1 -> 1
336: 3 1 2 3 4 2 -> 0
337: 3 2 1 1 1 1 -> 0
338: 3 2 1 1 1 2 -> 0
339: 3 2 1 1 2 1 -> 0
340: 3 2 1 1 2 2 -> 1
341: 3 2 1 1 3 1 -> 0
342: 3 2 1 1 3 2 -> 1
343: 3 2 1 1 4 1 -> 0
344: 3 2 1 1 4 2 -> 1
345: 3 2 1 2 1 1 -> 0
346: 3 2 1 2 1 2 -> 1
347: 3 2 1 2 2 1 -> 1
348: 3 2 1 2 2 2 -> 0
349: 3 2 1 2 3 1 -> 1
350: 3 2 1 2 3 2 -> 0
351: 3 2 1 2 4 1 -> 1
352: 3 2 1 2 4 2 -> 0
353: 3 2 1 3 1 1 -> 0
354: 3 2 1 3 1 2 -> 1
355: 3 2 1 3 2 1 -> 1
356: 3 2 1 3 2 2 -> 0
357: 3 2 1 3 3 1 -> 1
358: 3 2 1 3 3 2 -> 0
359: 3 2 1 3 4 1 -> 1
360: 3 2 1 3 4 2 -> 0
361: 3 2 2 1 1 1 -> 0
362: 3 2 2 1 1 2 -> 1
363: 3 2 2 1 2 1 -> 1
364: 3 2 2 1 2 2 -> 0
365: 3 2 2 1 3 1 -> 1
366: 3 2 2 1 3 2 -> 0
367: 3 2 2 1 4 1 -> 1
368: 3 2 2 1 4 2 -> 0
369: 3 2 2 2 1 1 -> 1
370: 3 2 2 2 1 2 -> 0
371: 3 2 2 2 2 1 -> 0
372: 3 2 2 2 2 2 -> 0
373: 3 2 2 2 3 1 -> 0
374: 3 2 2 2 3 2 -> 0
375: 3 2 2 2 4 1 -> 0
376: 3 2 2 2 4 2 -> 0
377: 3 2 2 3 1 1 -> 1
378: 3 2 2 3 1 2 -> 0
379: 3 2 2 3 2 1 -> 0
380: 3 2 2 3 2 2 -> 0
381: 3 2 2 3 3 1 -> 0
382: 3 2 2 3 3 2 -> 0
383: 3 2 2 3 4 1 -> 0
384: 3 2 2 3 4 2 -> 0
385: 3 3 1 1 1 1 -> 0
386: 3 3 1 1 1 2 -> 0
387: 3 3 1 1 2 1 -> 0
388: 3 3 1 1 2 2 -> 1
389: 3 3 1 1 3 1 -> 0
390: 3 3 1 1 3 2 -> 1
391: 3 3 1 1 4 1 -> 0
392: 3 3 1 1 4 2 -> 1
393: 3 3 1 2 1 1 -> 0
394: 3 3 1 2 1 2 -> 1
395: 3 3 1 2 2 1 -> 1
396: 3 3 1 2 2 2 -> 0
397: 3 3 1 2 3 1 -> 1
398: 3 3 1 2 3 2 -> 0
399: 3 3 1 2 4 1 -> 1
400: 3 3 1 2 4 2 -> 0
401: 3 3 1 3 1 1 -> 0
402: 3 3 1 3 1 2 -> 1
403: 3 3 1 3 2 1 -> 1
404: 3 3 1 3 2 2 -> 0
405: 3 3 1 3 3 1 -> 1
406: 3 3 1 3 3 2 -> 0
407: 3 3 1 3 4 1 -> 1
408: 3 3 1 3 4 2 -> 0
409: 3 3 2 1 1 1 -> 0
410: 3 3 2 1 1 2 -> 1
411: 3 3 2 1 2 1 -> 1
412: 3 3 2 1 2 2 -> 0
413: 3 3 2 1 3 1 -> 1
414: 3 3 2 1 3 2 -> 0
415: 3 3 2 1 4 1 -> 1
416: 3 3 2 1 4 2 -> 0
417: 3 3 2 2 1 1 -> 1
418: 3 3 2 2 1 2 -> 0
419: 3 3 2 2 2 1 -> 0
420: 3 3 2 2 2 2 -> 0
421: 3 3 2 2 3 1 -> 0
422: 3 3 2 2 3 2 -> 0
423: 3 3 2 2 4 1 -> 0
424: 3 3 2 2 4 2 -> 0
425: 3 3 2 3 1 1 -> 1
426: 3 3 2 3 1 2 -> 0
427: 3 3 2 3 2 1 -> 0
428: 3 3 2 3 2 2 -> 0
429: 3 3 2 3 3 1 -> 0
430: 3 3 2 3 3 2 -> 0
431: 3 3 2 3 4 1 -> 0
432: 3 3 2 3 4 2 -> 0

Training Data Set #3

The training set of examples for the third problem with noisy data:
(jacket_color is green and holding is sword) or
(jacket color is not blue and body_shape is not octagon)

2: 1 1 1 1 1 2 -> 1
3: 1 1 1 1 2 1 -> 1
4: 1 1 1 1 2 2 -> 1
5: 1 1 1 1 3 1 -> 0
7: 1 1 1 1 4 1 -> 0
9: 1 1 1 2 1 1 -> 1
12: 1 1 1 2 2 2 -> 1
16: 1 1 1 2 4 2 -> 0
28: 1 1 2 1 2 2 -> 1
32: 1 1 2 1 4 2 -> 0
36: 1 1 2 2 2 2 -> 1
39: 1 1 2 2 4 1 -> 0
40: 1 1 2 2 4 2 -> 0
41: 1 1 2 3 1 1 -> 1
42: 1 1 2 3 1 2 -> 1
45: 1 1 2 3 3 1 -> 1
46: 1 1 2 3 3 2 -> 1
53: 1 2 1 1 3 1 -> 1
59: 1 2 1 2 2 1 -> 1
60: 1 2 1 2 2 2 -> 1
61: 1 2 1 2 3 1 -> 0
65: 1 2 1 3 1 1 -> 1
66: 1 2 1 3 1 2 -> 1
67: 1 2 1 3 2 1 -> 1
68: 1 2 1 3 2 2 -> 1
70: 1 2 1 3 3 2 -> 1
71: 1 2 1 3 4 1 -> 0
77: 1 2 2 1 3 1 -> 1
80: 1 2 2 1 4 2 -> 0
81: 1 2 2 2 1 1 -> 1
83: 1 2 2 2 2 1 -> 1
84: 1 2 2 2 2 2 -> 1
89: 1 2 2 3 1 1 -> 1
91: 1 2 2 3 2 1 -> 1
92: 1 2 2 3 2 2 -> 1
99: 1 3 1 1 2 1 -> 0
103: 1 3 1 1 4 1 -> 0
110: 1 3 1 2 3 2 -> 0
111: 1 3 1 2 4 1 -> 0
113: 1 3 1 3 1 1 -> 0
117: 1 3 1 3 3 1 -> 0
121: 1 3 2 1 1 1 -> 0
122: 1 3 2 1 1 2 -> 0
123: 1 3 2 1 2 1 -> 0
128: 1 3 2 1 4 2 -> 0
134: 1 3 2 2 3 2 -> 0
136: 1 3 2 2 4 2 -> 0
143: 1 3 2 3 4 1 -> 0
145: 2 1 1 1 1 1 -> 1
146: 2 1 1 1 1 2 -> 1
151: 2 1 1 1 4 1 -> 0
152: 2 1 1 1 4 2 -> 0
153: 2 1 1 2 1 1 -> 1
154: 2 1 1 2 1 2 -> 1
164: 2 1 1 3 2 2 -> 1
166: 2 1 1 3 3 2 -> 1
167: 2 1 1 3 4 1 -> 0
172: 2 1 2 1 2 2 -> 1
183: 2 1 2 2 4 1 -> 0
186: 2 1 2 3 1 2 -> 1
198: 2 2 1 1 3 2 -> 1
200: 2 2 1 1 4 2 -> 0
202: 2 2 1 2 1 2 -> 1
203: 2 2 1 2 2 1 -> 0
209: 2 2 1 3 1 1 -> 1
212: 2 2 1 3 2 2 -> 1
213: 2 2 1 3 3 1 -> 0
214: 2 2 1 3 3 2 -> 0
216: 2 2 1 3 4 2 -> 0
220: 2 2 2 1 2 2 -> 1
226: 2 2 2 2 1 2 -> 1
229: 2 2 2 2 3 1 -> 1
230: 2 2 2 2 3 2 -> 1
239: 2 2 2 3 4 1 -> 0
245: 2 3 1 1 3 1 -> 1
249: 2 3 1 2 1 1 -> 0
251: 2 3 1 2 2 1 -> 0
252: 2 3 1 2 2 2 -> 0
254: 2 3 1 2 3 2 -> 0
261: 2 3 1 3 3 1 -> 0
266: 2 3 2 1 1 2 -> 0
268: 2 3 2 1 2 2 -> 0
271: 2 3 2 1 4 2 -> 0
277: 2 3 2 2 3 1 -> 0
280: 2 3 2 2 4 2 -> 0
281: 2 3 2 3 1 1 -> 0
283: 2 3 2 3 2 1 -> 0
288: 2 3 2 3 4 2 -> 0
289: 3 1 1 1 1 1 -> 1
291: 3 1 1 1 2 1 -> 1
293: 3 1 1 1 3 1 -> 1
304: 3 1 1 2 4 2 -> 0
306: 3 1 1 3 1 2 -> 1
312: 3 1 1 3 4 2 -> 0
315: 3 1 2 1 2 1 -> 1
326: 3 1 2 2 3 2 -> 1
328: 3 1 2 2 4 2 -> 0
329: 3 1 2 3 1 1 -> 1
340: 3 2 1 1 2 2 -> 1
343: 3 2 1 1 4 1 -> 0
349: 3 2 1 2 3 1 -> 1
354: 3 2 1 3 1 2 -> 1
364: 3 2 2 1 2 2 -> 1
366: 3 2 2 1 3 2 -> 1
370: 3 2 2 2 1 2 -> 1
377: 3 2 2 3 1 1 -> 1
382: 3 2 2 3 3 2 -> 1
383: 3 2 2 3 4 1 -> 0
390: 3 3 1 1 3 2 -> 1
391: 3 3 1 1 4 1 -> 1
400: 3 3 1 2 4 2 -> 0
401: 3 3 1 3 1 1 -> 0
403: 3 3 1 3 2 1 -> 0
404: 3 3 1 3 2 2 -> 0
407: 3 3 1 3 4 1 -> 0
409: 3 3 2 1 1 1 -> 0
410: 3 3 2 1 1 2 -> 0
420: 3 3 2 2 2 2 -> 0
422: 3 3 2 2 3 2 -> 0
425: 3 3 2 3 1 1 -> 0
430: 3 3 2 3 3 2 -> 0
432: 3 3 2 3 4 2 -> 0

Testing Data Set #3

1: 1 1 1 1 1 1 -> 1
2: 1 1 1 1 1 2 -> 1
3: 1 1 1 1 2 1 -> 1
4: 1 1 1 1 2 2 -> 1
5: 1 1 1 1 3 1 -> 1
6: 1 1 1 1 3 2 -> 1
7: 1 1 1 1 4 1 -> 0
8: 1 1 1 1 4 2 -> 0
9: 1 1 1 2 1 1 -> 1
10: 1 1 1 2 1 2 -> 1
11: 1 1 1 2 2 1 -> 1
12: 1 1 1 2 2 2 -> 1
13: 1 1 1 2 3 1 -> 1
14: 1 1 1 2 3 2 -> 1
15: 1 1 1 2 4 1 -> 0
16: 1 1 1 2 4 2 -> 0
17: 1 1 1 3 1 1 -> 1
18: 1 1 1 3 1 2 -> 1
19: 1 1 1 3 2 1 -> 1
20: 1 1 1 3 2 2 -> 1
21: 1 1 1 3 3 1 -> 1
22: 1 1 1 3 3 2 -> 1
23: 1 1 1 3 4 1 -> 0
24: 1 1 1 3 4 2 -> 0
25: 1 1 2 1 1 1 -> 1
26: 1 1 2 1 1 2 -> 1
27: 1 1 2 1 2 1 -> 1
28: 1 1 2 1 2 2 -> 1
29: 1 1 2 1 3 1 -> 1
30: 1 1 2 1 3 2 -> 1
31: 1 1 2 1 4 1 -> 0
32: 1 1 2 1 4 2 -> 0
33: 1 1 2 2 1 1 -> 1
34: 1 1 2 2 1 2 -> 1
35: 1 1 2 2 2 1 -> 1
36: 1 1 2 2 2 2 -> 1
37: 1 1 2 2 3 1 -> 1
38: 1 1 2 2 3 2 -> 1
39: 1 1 2 2 4 1 -> 0
40: 1 1 2 2 4 2 -> 0
41: 1 1 2 3 1 1 -> 1
42: 1 1 2 3 1 2 -> 1
43: 1 1 2 3 2 1 -> 1
44: 1 1 2 3 2 2 -> 1
45: 1 1 2 3 3 1 -> 1
46: 1 1 2 3 3 2 -> 1
47: 1 1 2 3 4 1 -> 0
48: 1 1 2 3 4 2 -> 0
49: 1 2 1 1 1 1 -> 1
50: 1 2 1 1 1 2 -> 1
51: 1 2 1 1 2 1 -> 1
52: 1 2 1 1 2 2 -> 1
53: 1 2 1 1 3 1 -> 1
54: 1 2 1 1 3 2 -> 1
55: 1 2 1 1 4 1 -> 0
56: 1 2 1 1 4 2 -> 0
57: 1 2 1 2 1 1 -> 1
58: 1 2 1 2 1 2 -> 1
59: 1 2 1 2 2 1 -> 1
60: 1 2 1 2 2 2 -> 1
61: 1 2 1 2 3 1 -> 1
62: 1 2 1 2 3 2 -> 1
63: 1 2 1 2 4 1 -> 0
64: 1 2 1 2 4 2 -> 0
65: 1 2 1 3 1 1 -> 1
66: 1 2 1 3 1 2 -> 1
67: 1 2 1 3 2 1 -> 1
68: 1 2 1 3 2 2 -> 1
69: 1 2 1 3 3 1 -> 1
70: 1 2 1 3 3 2 -> 1
71: 1 2 1 3 4 1 -> 0
72: 1 2 1 3 4 2 -> 0
73: 1 2 2 1 1 1 -> 1
74: 1 2 2 1 1 2 -> 1
75: 1 2 2 1 2 1 -> 1
76: 1 2 2 1 2 2 -> 1
77: 1 2 2 1 3 1 -> 1
78: 1 2 2 1 3 2 -> 1
79: 1 2 2 1 4 1 -> 0
80: 1 2 2 1 4 2 -> 0
81: 1 2 2 2 1 1 -> 1
82: 1 2 2 2 1 2 -> 1
83: 1 2 2 2 2 1 -> 1
84: 1 2 2 2 2 2 -> 1
85: 1 2 2 2 3 1 -> 1
86: 1 2 2 2 3 2 -> 1
87: 1 2 2 2 4 1 -> 0
88: 1 2 2 2 4 2 -> 0
89: 1 2 2 3 1 1 -> 1
90: 1 2 2 3 1 2 -> 1
91: 1 2 2 3 2 1 -> 1
92: 1 2 2 3 2 2 -> 1
93: 1 2 2 3 3 1 -> 1
94: 1 2 2 3 3 2 -> 1
95: 1 2 2 3 4 1 -> 0
96: 1 2 2 3 4 2 -> 0

97: 1 3 1 1 1 1 -> 0
98: 1 3 1 1 1 2 -> 0
99: 1 3 1 1 2 1 -> 0
100: 1 3 1 1 2 2 -> 0
101: 1 3 1 1 3 1 -> 1
102: 1 3 1 1 3 2 -> 1
103: 1 3 1 1 4 1 -> 0
104: 1 3 1 1 4 2 -> 0
105: 1 3 1 2 1 1 -> 0
106: 1 3 1 2 1 2 -> 0
107: 1 3 1 2 2 1 -> 0
108: 1 3 1 2 2 2 -> 0
109: 1 3 1 2 3 1 -> 0
110: 1 3 1 2 3 2 -> 0
111: 1 3 1 2 4 1 -> 0
112: 1 3 1 2 4 2 -> 0
113: 1 3 1 3 1 1 -> 0
114: 1 3 1 3 1 2 -> 0
115: 1 3 1 3 2 1 -> 0
116: 1 3 1 3 2 2 -> 0
117: 1 3 1 3 3 1 -> 0
118: 1 3 1 3 3 2 -> 0
119: 1 3 1 3 4 1 -> 0
120: 1 3 1 3 4 2 -> 0
121: 1 3 2 1 1 1 -> 0
122: 1 3 2 1 1 2 -> 0
123: 1 3 2 1 2 1 -> 0
124: 1 3 2 1 2 2 -> 0
125: 1 3 2 1 3 1 -> 1
126: 1 3 2 1 3 2 -> 1
127: 1 3 2 1 4 1 -> 0
128: 1 3 2 1 4 2 -> 0
129: 1 3 2 2 1 1 -> 0
130: 1 3 2 2 1 2 -> 0
131: 1 3 2 2 2 1 -> 0
132: 1 3 2 2 2 2 -> 0
133: 1 3 2 2 3 1 -> 0
134: 1 3 2 2 3 2 -> 0
135: 1 3 2 2 4 1 -> 0
136: 1 3 2 2 4 2 -> 0
137: 1 3 2 3 1 1 -> 0
138: 1 3 2 3 1 2 -> 0
139: 1 3 2 3 2 1 -> 0
140: 1 3 2 3 2 2 -> 0
141: 1 3 2 3 3 1 -> 0
142: 1 3 2 3 3 2 -> 0
143: 1 3 2 3 4 1 -> 0
144: 1 3 2 3 4 2 -> 0
145: 2 1 1 1 1 1 -> 1
146: 2 1 1 1 1 2 -> 1
147: 2 1 1 1 2 1 -> 1
148: 2 1 1 1 2 2 -> 1
149: 2 1 1 1 3 1 -> 1
150: 2 1 1 1 3 2 -> 1
151: 2 1 1 1 4 1 -> 0
152: 2 1 1 1 4 2 -> 0
153: 2 1 1 2 1 1 -> 1
154: 2 1 1 2 1 2 -> 1
155: 2 1 1 2 2 1 -> 1
156: 2 1 1 2 2 2 -> 1
157: 2 1 1 2 3 1 -> 1
158: 2 1 1 2 3 2 -> 1
159: 2 1 1 2 4 1 -> 0
160: 2 1 1 2 4 2 -> 0
161: 2 1 1 3 1 1 -> 1
162: 2 1 1 3 1 2 -> 1
163: 2 1 1 3 2 1 -> 1
164: 2 1 1 3 2 2 -> 1
165: 2 1 1 3 3 1 -> 1
166: 2 1 1 3 3 2 -> 1
167: 2 1 1 3 4 1 -> 0
168: 2 1 1 3 4 2 -> 0
169: 2 1 2 1 1 1 -> 1
170: 2 1 2 1 1 2 -> 1
171: 2 1 2 1 2 1 -> 1
172: 2 1 2 1 2 2 -> 1
173: 2 1 2 1 3 1 -> 1
174: 2 1 2 1 3 2 -> 1
175: 2 1 2 1 4 1 -> 0
176: 2 1 2 1 4 2 -> 0
177: 2 1 2 2 1 1 -> 1
178: 2 1 2 2 1 2 -> 1
179: 2 1 2 2 2 1 -> 1
180: 2 1 2 2 2 2 -> 1
181: 2 1 2 2 3 1 -> 1
182: 2 1 2 2 3 2 -> 1
183: 2 1 2 2 4 1 -> 0
184: 2 1 2 2 4 2 -> 0
185: 2 1 2 3 1 1 -> 1
186: 2 1 2 3 1 2 -> 1
187: 2 1 2 3 2 1 -> 1
188: 2 1 2 3 2 2 -> 1
189: 2 1 2 3 3 1 -> 1
190: 2 1 2 3 3 2 -> 1
191: 2 1 2 3 4 1 -> 0
192: 2 1 2 3 4 2 -> 0
193: 2 2 1 1 1 1 -> 1
194: 2 2 1 1 1 2 -> 1
195: 2 2 1 1 2 1 -> 1
196: 2 2 1 1 2 2 -> 1
197: 2 2 1 1 3 1 -> 1
198: 2 2 1 1 3 2 -> 1
199: 2 2 1 1 4 1 -> 0
200: 2 2 1 1 4 2 -> 0
201: 2 2 1 2 1 1 -> 1
202: 2 2 1 2 1 2 -> 1
203: 2 2 1 2 2 1 -> 1
204: 2 2 1 2 2 2 -> 1
205: 2 2 1 2 3 1 -> 1
206: 2 2 1 2 3 2 -> 1
207: 2 2 1 2 4 1 -> 0
208: 2 2 1 2 4 2 -> 0
209: 2 2 1 3 1 1 -> 1
210: 2 2 1 3 1 2 -> 1
211: 2 2 1 3 2 1 -> 1
212: 2 2 1 3 2 2 -> 1
213: 2 2 1 3 3 1 -> 1
214: 2 2 1 3 3 2 -> 1
215: 2 2 1 3 4 1 -> 0
216: 2 2 1 3 4 2 -> 0
217: 2 2 2 1 1 1 -> 1
218: 2 2 2 1 1 2 -> 1
219: 2 2 2 1 2 1 -> 1
220: 2 2 2 1 2 2 -> 1
221: 2 2 2 1 3 1 -> 1
222: 2 2 2 1 3 2 -> 1
223: 2 2 2 1 4 1 -> 0
224: 2 2 2 1 4 2 -> 0
225: 2 2 2 2 1 1 -> 1
226: 2 2 2 2 1 2 -> 1
227: 2 2 2 2 2 1 -> 1
228: 2 2 2 2 2 2 -> 1
229: 2 2 2 2 3 1 -> 1
230: 2 2 2 2 3 2 -> 1
231: 2 2 2 2 4 1 -> 0
232: 2 2 2 2 4 2 -> 0
233: 2 2 2 3 1 1 -> 1
234: 2 2 2 3 1 2 -> 1
235: 2 2 2 3 2 1 -> 1
236: 2 2 2 3 2 2 -> 1
237: 2 2 2 3 3 1 -> 1
238: 2 2 2 3 3 2 -> 1
239: 2 2 2 3 4 1 -> 0
240: 2 2 2 3 4 2 -> 0
241: 2 3 1 1 1 1 -> 0
242: 2 3 1 1 1 2 -> 0
243: 2 3 1 1 2 1 -> 0
244: 2 3 1 1 2 2 -> 0
245: 2 3 1 1 3 1 -> 1
246: 2 3 1 1 3 2 -> 1
247: 2 3 1 1 4 1 -> 0
248: 2 3 1 1 4 2 -> 0
249: 2 3 1 2 1 1 -> 0
250: 2 3 1 2 1 2 -> 0
251: 2 3 1 2 2 1 -> 0
252: 2 3 1 2 2 2 -> 0
253: 2 3 1 2 3 1 -> 0
254: 2 3 1 2 3 2 -> 0
255: 2 3 1 2 4 1 -> 0
256: 2 3 1 2 4 2 -> 0
257: 2 3 1 3 1 1 -> 0
258: 2 3 1 3 1 2 -> 0
259: 2 3 1 3 2 1 -> 0
260: 2 3 1 3 2 2 -> 0
261: 2 3 1 3 3 1 -> 0
262: 2 3 1 3 3 2 -> 0
263: 2 3 1 3 4 1 -> 0
264: 2 3 1 3 4 2 -> 0
265: 2 3 2 1 1 1 -> 0
266: 2 3 2 1 1 2 -> 0
267: 2 3 2 1 2 1 -> 0
268: 2 3 2 1 2 2 -> 0
269: 2 3 2 1 3 1 -> 1
270: 2 3 2 1 3 2 -> 1
271: 2 3 2 1 4 1 -> 0
272: 2 3 2 1 4 2 -> 0
273: 2 3 2 2 1 1 -> 0
274: 2 3 2 2 1 2 -> 0
275: 2 3 2 2 2 1 -> 0
276: 2 3 2 2 2 2 -> 0
277: 2 3 2 2 3 1 -> 0
278: 2 3 2 2 3 2 -> 0
279: 2 3 2 2 4 1 -> 0
280: 2 3 2 2 4 2 -> 0
281: 2 3 2 3 1 1 -> 0
282: 2 3 2 3 1 2 -> 0
283: 2 3 2 3 2 1 -> 0
284: 2 3 2 3 2 2 -> 0
285: 2 3 2 3 3 1 -> 0
286: 2 3 2 3 3 2 -> 0
287: 2 3 2 3 4 1 -> 0
288: 2 3 2 3 4 2 -> 0
289: 3 1 1 1 1 1 -> 1
290: 3 1 1 1 1 2 -> 1
291: 3 1 1 1 2 1 -> 1
292: 3 1 1 1 2 2 -> 1
293: 3 1 1 1 3 1 -> 1
294: 3 1 1 1 3 2 -> 1
295: 3 1 1 1 4 1 -> 0
296: 3 1 1 1 4 2 -> 0
297: 3 1 1 2 1 1 -> 1
298: 3 1 1 2 1 2 -> 1
299: 3 1 1 2 2 1 -> 1
300: 3 1 1 2 2 2 -> 1
301: 3 1 1 2 3 1 -> 1
302: 3 1 1 2 3 2 -> 1
303: 3 1 1 2 4 1 -> 0
304: 3 1 1 2 4 2 -> 0
305: 3 1 1 3 1 1 -> 1
306: 3 1 1 3 1 2 -> 1
307: 3 1 1 3 2 1 -> 1
308: 3 1 1 3 2 2 -> 1
309: 3 1 1 3 3 1 -> 1
310: 3 1 1 3 3 2 -> 1
311: 3 1 1 3 4 1 -> 0
312: 3 1 1 3 4 2 -> 0
313: 3 1 2 1 1 1 -> 1
314: 3 1 2 1 1 2 -> 1
315: 3 1 2 1 2 1 -> 1
316: 3 1 2 1 2 2 -> 1
317: 3 1 2 1 3 1 -> 1
318: 3 1 2 1 3 2 -> 1
319: 3 1 2 1 4 1 -> 0
320: 3 1 2 1 4 2 -> 0
321: 3 1 2 2 1 1 -> 1
322: 3 1 2 2 1 2 -> 1
323: 3 1 2 2 2 1 -> 1
324: 3 1 2 2 2 2 -> 1
325: 3 1 2 2 3 1 -> 1
326: 3 1 2 2 3 2 -> 1
327: 3 1 2 2 4 1 -> 0
328: 3 1 2 2 4 2 -> 0
329: 3 1 2 3 1 1 -> 1
330: 3 1 2 3 1 2 -> 1
331: 3 1 2 3 2 1 -> 1
332: 3 1 2 3 2 2 -> 1
333: 3 1 2 3 3 1 -> 1
334: 3 1 2 3 3 2 -> 1
335: 3 1 2 3 4 1 -> 0
336: 3 1 2 3 4 2 -> 0
337: 3 2 1 1 1 1 -> 1
338: 3 2 1 1 1 2 -> 1
339: 3 2 1 1 2 1 -> 1
340: 3 2 1 1 2 2 -> 1
341: 3 2 1 1 3 1 -> 1
342: 3 2 1 1 3 2 -> 1
343: 3 2 1 1 4 1 -> 0
344: 3 2 1 1 4 2 -> 0
345: 3 2 1 2 1 1 -> 1
346: 3 2 1 2 1 2 -> 1
347: 3 2 1 2 2 1 -> 1
348: 3 2 1 2 2 2 -> 1
349: 3 2 1 2 3 1 -> 1
350: 3 2 1 2 3 2 -> 1
351: 3 2 1 2 4 1 -> 0
352: 3 2 1 2 4 2 -> 0

353: 3 2 1 3 1 1 -> 1
354: 3 2 1 3 1 2 -> 1
355: 3 2 1 3 2 1 -> 1
356: 3 2 1 3 2 2 -> 1
357: 3 2 1 3 3 1 -> 1
358: 3 2 1 3 3 2 -> 1
359: 3 2 1 3 4 1 -> 0
360: 3 2 1 3 4 2 -> 0
361: 3 2 2 1 1 1 -> 1
362: 3 2 2 1 1 2 -> 1
363: 3 2 2 1 2 1 -> 1
364: 3 2 2 1 2 2 -> 1
365: 3 2 2 1 3 1 -> 1
366: 3 2 2 1 3 2 -> 1
367: 3 2 2 1 4 1 -> 0
368: 3 2 2 1 4 2 -> 0
369: 3 2 2 2 1 1 -> 1
370: 3 2 2 2 1 2 -> 1
371: 3 2 2 2 2 1 -> 1
372: 3 2 2 2 2 2 -> 1
373: 3 2 2 2 3 1 -> 1
374: 3 2 2 2 3 2 -> 1
375: 3 2 2 2 4 1 -> 0
376: 3 2 2 2 4 2 -> 0
377: 3 2 2 3 1 1 -> 1
378: 3 2 2 3 1 2 -> 1
379: 3 2 2 3 2 1 -> 1
380: 3 2 2 3 2 2 -> 1
381: 3 2 2 3 3 1 -> 1
382: 3 2 2 3 3 2 -> 1
383: 3 2 2 3 4 1 -> 0
384: 3 2 2 3 4 2 -> 0
385: 3 3 1 1 1 1 -> 0
386: 3 3 1 1 1 2 -> 0
387: 3 3 1 1 2 1 -> 0
388: 3 3 1 1 2 2 -> 0
389: 3 3 1 1 3 1 -> 1
390: 3 3 1 1 3 2 -> 1
391: 3 3 1 1 4 1 -> 0
392: 3 3 1 1 4 2 -> 0
393: 3 3 1 2 1 1 -> 0
394: 3 3 1 2 1 2 -> 0
395: 3 3 1 2 2 1 -> 0
396: 3 3 1 2 2 2 -> 0
397: 3 3 1 2 3 1 -> 0
398: 3 3 1 2 3 2 -> 0
399: 3 3 1 2 4 1 -> 0
400: 3 3 1 2 4 2 -> 0
401: 3 3 1 3 1 1 -> 0
402: 3 3 1 3 1 2 -> 0
403: 3 3 1 3 2 1 -> 0
404: 3 3 1 3 2 2 -> 0
405: 3 3 1 3 3 1 -> 0
406: 3 3 1 3 3 2 -> 0
407: 3 3 1 3 4 1 -> 0
408: 3 3 1 3 4 2 -> 0
409: 3 3 2 1 1 1 -> 0
410: 3 3 2 1 1 2 -> 0
411: 3 3 2 1 2 1 -> 0
412: 3 3 2 1 2 2 -> 0
413: 3 3 2 1 3 1 -> 1
414: 3 3 2 1 3 2 -> 1
415: 3 3 2 1 4 1 -> 0
416: 3 3 2 1 4 2 -> 0
417: 3 3 2 2 1 1 -> 0
418: 3 3 2 2 1 2 -> 0
419: 3 3 2 2 2 1 -> 0
420: 3 3 2 2 2 2 -> 0
421: 3 3 2 2 3 1 -> 0
422: 3 3 2 2 3 2 -> 0
423: 3 3 2 2 4 1 -> 0
424: 3 3 2 2 4 2 -> 0
425: 3 3 2 3 1 1 -> 0
426: 3 3 2 3 1 2 -> 0
427: 3 3 2 3 2 1 -> 0
428: 3 3 2 3 2 2 -> 0
429: 3 3 2 3 3 1 -> 0
430: 3 3 2 3 3 2 -> 0
431: 3 3 2 3 4 1 -> 0
432: 3 3 2 3 4 2 -> 0