Artificial Intelligence Developments and Applications
J.S. Gero and R. Stanton (Editors)

Combining decision trees
Initial results from the MIL algorithm

Graham J. Williams[1]
Department of Computer Science
Australian National University
Canberra Australia

Abstract: The induction of a decision tree from a set of examples of decisions provided by an expert has become a useful tool for the construction of knowledge-based systems. Although a number of different decision trees can be constructed for the same domain, no techniques exist for combining them into integrated rule sets. The MIL algorithm is introduced as a means for combining two decision trees into a single set of rules, for use in a rule-based system.

1. INTRODUCTION

Knowledge acquisition is widely recognised as a major bottleneck in the development of an expert system. Typically, the knowledge bases required by such systems take from months to years to be constructed. Knowledge acquisition (and in particular machine learning) has thus become a major area of concern for expert systems research. Out of this research a number of promising algorithms have surfaced (see Michalski, Carbonell, & Mitchell, 1986). ID3 (Quinlan, 1986a) in particular has proved to be a very useful tool for aiding in the construction of knowledge bases. It implements a simple, and yet powerful method for constructing (or inducing) decision trees from examples of decisions. (This set of examples is called the *training set*.) ID3 employs a 'divide and conquer' technique, repeatedly dividing the collection of examples into smaller and more homogenous collections. The divides

[1]Current address:
BBJ Computers International
PO Box 7047, Melbourne Victoria 3004, Australia

correspond to branching in the decision tree, and each node of the tree corresponds to a collection of examples to be conquered. Successful systems have been constructed using this approach (Quinlan *et al.*, 1986).

Williams (1987) describes a number of experiments which investigated various aspects of the ID3 decision tree induction algorithm, and presented a number of techniques for improving upon the decision trees produced by this algorithm. The experiments illustrated, amongst other things, the algorithm's inability to always select a single decision tree from amongst all the possible decision trees. A number of "equally good" decision trees, with respect to the ID3 cost function, can be generated, with ID3 arbitrarily choosing one. A problem exists in deciding which, of a number of apparently equally good decision trees, to use. The **MIL** algorithm, as described here, attempts to take advantage of this situation by introducing the idea of combining decision trees. As well as making use of more of the information in a training set, **MIL** leads to richer knowledge bases which improve upon the combined performance and coverage of the individual ID3 decision trees.

An example application of the ID3 algorithm is presented in the following section. The construction of two "equally good" decision trees is illustrated. Section 3 then introduces the **MIL** algorithm, and is followed by an example application of the algorithm, illustrating the advantages of using **MIL**. Other preliminary results are then presented.

2. DECISION TREE INDUCTION

An example will be used to illustrate the ID3 algorithm; full details of which are provided by Quinlan (1986a). Rule sets consisting of rules of the form "If A and B and ... Then W" will be used to illustrate the decision trees. Each path through a decision tree is simply converted to a single rule, so that rules in a rule set and paths through a decision tree have a one-to-one mapping. The If-part of the rule corresponds to the tests performed at each node of the decision tree, and the Then-part corresponds to the leaf node of the path.

Williams (1987) describes the application of the ID3 algorithm to the task of predicting the viability for grazing cattle in the range-land regions of Australia. The ARID database is used. This is a subset of the Australian Resources Information System (Walker, Cocks, & Young, 1985), and consists of 8413 records. Each record (i.e., object) of the database corresponds to a single grid-cell, which is an approximately 700 square kilometre rectangular region; no regions overlap. For each object seven attributes are recorded. These are the predominant soil type (Soil), the predominant class of upper and lower storey vegetation (UVeg and LVeg respectively), the distance in kilometres to the nearest seaport (DPort), and three moisture indicators (AWMIH, AWMIS, and AWMIW). AWMIH is the average weekly moisture

index for the wettest consecutive 13 weeks of the year; AWMIS is the average weekly moisture index from November to April inclusive (i.e., Summer in Australia); and AWMIW is the average weekly moisture index from May to October inclusive (i.e., Winter in Australia). Soil, UVeg, and LVeg are unordered categorical attributes and can have one of 30, 50 and 41 possible values respectively. The moisture indicators are integer attributes, taking on integer values in the range 0 to 100.

A training set of 106 representative objects, referred to here as the T106 training set, was chosen from the ARID database. Each object in the training set was given a classification, by a domain expert, indicating the viability of grazing cattle in the corresponding region. The viability is expressed as being one of four values: VLow (very low), Low, Medium, and High.

A regression model, based upon the 106 objects in the training set, has been constructed for the prediction of viability (Cocks, Young, & Walker, 1986). This model is used as the oracle against which the performance of the knowledge bases constructed here is compared.

ID3 begins by considering a number of candidate partitions of the training set. Each partition consists of two or more subsets of the training set, with each attribute giving rise to a candidate partition. For categorical attributes, each subset of the partition corresponds to a particular category of the attribute. For integer attributes, two subsets are constructed, one consisting of all objects having a value for the attribute less than some threshold, and the other containing those objects with a value greater than or equal to the threshold. ID3 then employs an entropy-type function to compute a measure of the homogeneity, with respect to the Viability attribute, of the various partitions of the training set. The attribute which partitions the training set into the most homogenous subsets, as measured by this 'cost function', is chosen as the root node of the decision tree. The attribute Soil was chosen as the root node from the T106 training set.

Each of the subsets of the chosen partition are then considered, in turn, as new training sets (sometimes referred to as training subsets). For example, for the set of objects in T106 with a value of 15 for Soil (9 objects), a number of partitions were constructed, and one (based upon the attribute DPort with a threshold of 783) was chosen by ID3.

This divide and conquer process stops whenever all of the objects in a training set have the same viability rating (i.e., all in the same class). For example, all objects in the training subset containing those items with Soil=15 and DPort$\geq$783 were classified as having medium viability. Thus, this training subset is not further partitioned. In such a case, a leaf node of the decision tree (or conclusion of a rule) has been reached, and the appropiate class is associated with it.

As mentioned previously, a problem arises when the cost function is unable to distinguish between two or more attributes. Such a situation occurs several times in the application of ID3 to the viability problem. For example, when Soil=26, UVeg and DPort tie for the minimum value of the cost function. Rather than arbitrarily choosing between the attributes, it seems sensible to consider them both, thus constructing two different decision trees. For the above task Williams (1987) constructs two such decision trees, which, according to ID3, are equally good. These are the T106DC and T106DI decision trees. The complete decision trees are not reproduced here. However, sample rules from the corresponding rule sets are given below. The samples illustrate the similarities and differences between the two rule sets.

T106DC:

If Soil=2 Then Viabilty=Medium
If Soil=15 and DPort≥783 Then Viabilty=Medium
If Soil=16 and UVeg=9 Then Viabilty=Low
If Soil=26 and UVeg=2 Then Viabilty=VLow

T106DI:

If Soil=2 Then Viabilty=Medium
If Soil=15 and DPort≥783 Then Viabilty=Medium
If Soil=16 and DPort<766 Then Viabilty=Low
If Soil=26 and DPort<635 Then Viabilty=Medium

Two measures of performance are used. They are the coverage, or percentage of objects in the ARID database for which the rule set is able to give classifications, and the accuracy, which is measured with respect to the classifications given to objects by the regression model. The classification of an object by the rule sets here can either agree, mildly disagree, moderately disagree, or strongly disagree, with the classification given to the same object by the regression model. Two classifications of one object mildly disagree if the two classes are neighbours in the natural ordering of the classes: viz VLow, Low, Medium, High. For example, Low and Medium are neighbours, whilst VLow and Medium are not. A moderate disagreement occurs when the two classes are separated by one class, and a strong disagreement occurs when the two are separated by two classes. Thus VLow and Medium represents a moderate disagreement, whilst VLow and High represents a strong disagreement.

These measures for the two rule sets T106DC and T106DI are given in Table 1. It is seen that whilst T106DI is able to classify considerably more objects in the ARID database than T106DC, its accuracy has suffered.

Experiment	Description	Cover	Agree	Mild	Mod	Strong
T106DC	Favour Categoricals	70.8	71.5	26.7	1.8	0.0
T106DI	Favour Integers	84.3	64.8	33.1	2.1	0.0

TABLE 1: *Summary of results from applying the rule sets T106DC and T106DI, to the ARID database. The "Cover" is the percentage of the objects in the ARID database which are classifiable by the rule set. The other columns list the percentage agreement, mild disagreement, moderate disagreement, and strong disagreement between the classification given to objects by the rule set, and the classification given by the regression model. Adapted from Williams (1987).*

This illustrates a major difficulty with ID3. That is, it is possible to construct two different, but apparently equally good, decision trees which differ considerably with respect to their performances.

3. THE MIL ALGORITHM

The previous section illustrates ID3's potential for producing a number of "equally good" rule sets (or decision trees). We now look at a method for combining such rule sets.

The immediate difficulty with combining rule sets is the problem of handling conflicting classifications between the rule sets. That is, when one object is classified differently by two rules from different rule sets. For example, any object from the ARID database with Soil=26, UVeg=2, and DPort<635 will be classified as VLow by T106DC, but as Medium by T106DI. The resolution of these conflicts is the basis of the **MIL** algorithm.

The **MIL** algorithm takes two rule sets, each derived from different decision trees, and produces a combined rule set which, with respect to some *pilot set* of objects, does not produce any conflicts. The heart of the **MIL** algorithm is the **MIL** conflict resolver ($\mathbf{MIL}_{cr}$). $\mathbf{MIL}_{cr}$ is given a pair of rules which give conflicting classifications to some objects in the pilot set, and returns a number of rules which do not reproduce these conflicts, nor introduce any new ones. $\mathbf{MIL}_{cr}$ is iteratively applied until no conflicts remain.

Before describing the algorithm, the notation will be introduced, followed by abstract specifications of **MIL** and $\mathbf{MIL}_{cr}$. Each of the three main tasks of the $\mathbf{MIL}_{cr}$ system will then be described.

3.1 Terminology

The set $\mathbf{A} = \{A_1, \ldots, A_p\}$ is the set of *domain attributes* which are used to describe the objects of the domain. Each attribute takes on either integer

values or categorical values, and are called integer and categorical attributes respectively. Categorical values come from a (small) set of possible values associated with the categorical attribute. $A_i(o)$ denotes the value of the attribute A_i associated with the object o. The universe of all objects is denoted by **Ou**; thus, $o \in \mathbf{Ou}$. A distinguished attribute, $Class \notin \mathbf{A}$, is called the *conclusion attribute,* and is the only attribute appearing as the conclusion of a rule.

A *rule set* is a set of rules, each rule having the form

$$R_j\text{: } Cond \Rightarrow Class_i,$$

where *Cond* specifies the conditions under which the conclusion $Class_i$ can be derived for some object. $Cond(o)$ is a predicate which is true only if the object o satisfies the conditions specified by *Cond*. If $Cond(o)$ is true, then the above rule is said to *trigger* with respect to the object o, and the predicate trigger(R, o) is true. The predicate conclude$(R, o, Class_i)$ is true if rule R triggers on object o, concluding that the object belongs to $Class_i$. If the first argument to this predicate is a rule set rather than a single rule, then the predicate is true if there exists at least one rule in the rule set which triggers on object o, and concludes that o belongs to $Class_i$. *Cond* is called a *condition* and has the form:

$$C_1 \wedge \ldots \wedge C_n,$$

where C_j has one of the forms

$$A_i = v,$$
$$A_i < v,$$
$$A_i \geq v,$$

where $A_i \in \mathbf{A}$ and v is from the set of possible values for the attribute (either a category or an integer). C_j is referred to as a *condition triplet* since it always consists of an attribute, a relational operator, and a value (or set of values), and is written, for example, as $[A_i, =, v]$. Further relational operators are introduced later.

$\mathbf{R}_1$ and $\mathbf{R}_2$ will denote two rule sets derived directly from different decision trees; both decision trees being constructed using the same training set, **Tr**. A rule set is *derived directly* from a decision tree by constructing a single rule for each path through the decision tree.

P is a *pilot set*—a set of objects (called *pilot objects*) to which the rule sets will be applied in order to identify conflicts between the conclusions reached by the rule sets. (It is called a pilot set since it is used to guide the application of **MIL** "through unknown places.") A *conflict* between two

rules, R_1 and R_2 from $\mathbf{R}_1$ and $\mathbf{R}_2$ respectively, consists of a non-empty set of objects from $\mathbf{P}$ for which both rules trigger to give different conclusions. This set of objects is called the *conflict set*, and is denoted by $\mathbf{Cs}$. The two rules will have the following form:

$$R_1\text{: } Cond_1 \Rightarrow Class_1,$$
$$R_2\text{: } Cond_2 \Rightarrow Class_2.$$

The conflict set is then defined as:

$$\mathbf{Cs} = \{o | o \in \mathbf{P},\ Cond_1(o) \wedge Cond_2(o)\}.$$

The conflict is denoted by $\mathcal{Q} = [R_1, R_2, \mathbf{Cs}]$.

The following two sets are the subsets of the training set containing those objects corresponding to R_1 and to R_2 respectively:

$$\mathbf{Tr}_1 = \{o | o \in \mathbf{Tr},\ Cond_1(o)\}$$
$$\mathbf{Tr}_2 = \{o | o \in \mathbf{Tr},\ Cond_2(o)\}$$

$\mathbf{Tr}_1 \cup \mathbf{Tr}_2$ is denoted by $\mathbf{Tc}$. The two sets

$$\mathbf{P}_1 = \{o | o \in \mathbf{P},\ Cond_1(o)\},$$
$$\mathbf{P}_2 = \{o | o \in \mathbf{P},\ Cond_2(o)\},$$

are the corresponding subsets of the pilot set. Thus $\mathbf{Cs} = \mathbf{P}_1 \cap \mathbf{P}_2$.

For the description of the **MIL** algorithm which follows, it is assumed that $\mathbf{Tr}$ and $\mathbf{P}$ remain constant. Further, the rule sets to which $\mathbf{R}_1$ and $\mathbf{R}_2$ refer to will remain the same. A rule with the subscript 1 will always be a member of $\mathbf{R}_1$, and similarly for rules with a subscript of 2. The form of a rule from $\mathbf{R}_1$ ($\mathbf{R}_2$) will also be identified with a subscript of 1 (2). Primes ($'$) are used to distinguish between rules from the same rule set.

3.2 Specifications

Abstract specifications of **MIL** and $\mathbf{MIL}_{cr}$ will make clear the intention of the **MIL** algorithm. It is recalled that the task of **MIL** is to combine two rule sets, and it does this by calling $\mathbf{MIL}_{cr}$ iteratively to resolve all conflicts between the two rule sets.

MIL can be thought of as a function with four arguments, taking two rule sets, the training set, and the pilot set, and returning a rule set:

$$\mathbf{MIL}(\mathbf{R}_1, \mathbf{R}_2, \mathbf{Tr}, \mathbf{P}) = \mathbf{R}.$$

The training set is a parameter since it is used by $\mathbf{MIL}_{cr}$, and the pilot set is required to identify conflicts.

The following properties of the output rule set, $\mathbf{R}$, must hold:

$$\forall\ o \in \mathbf{P}, \text{conclude}(\mathbf{R}_1, o, \mathit{Class}_1) \wedge \text{conclude}(\mathbf{R}_2, o, \mathit{Class}_2) \Rightarrow (\text{conclude}(\mathbf{R}, o, \mathit{Class}_1) \oplus \text{conclude}(\mathbf{R}, o, \mathit{Class}_2))$$

$$\forall\ o \in \mathbf{Tr}, \text{conclude}(\mathbf{R}_1, o, \mathit{Class}_1) \Rightarrow \text{conclude}(\mathbf{R}, o, \mathit{Class}_1)$$

$$\forall\ o \in \mathbf{Tr}, \text{conclude}(\mathbf{R}_2, o, \mathit{Class}_2) \Rightarrow \text{conclude}(\mathbf{R}, o, \mathit{Class}_2)$$

where $\oplus$ is exclusive or.

The first property states that $\mathbf{R}$ resolves all conflicts between $\mathbf{R}_1$ and $\mathbf{R}_2$, with respect to $\mathbf{P}$. The second and third properties state that $\mathbf{R}$, like $\mathbf{R}_1$ and $\mathbf{R}_2$, is $\mathbf{Tr}$-consistent. That is, it gives the same classifications to objects in $\mathbf{Tr}$ as given by either $\mathbf{R}_1$ or $\mathbf{R}_2$.

Similarly, $\mathbf{MIL}_{cr}$ can be viewed as a function taking a conflict (consisting of two rules and a non-empty conflict set) and the training set, and returning either two or four rules. MIL removes R_1 and R_2, and places the new rules in $\mathbf{R}$.

$\mathbf{MIL}_{cr}(R_1, R_2, \mathbf{Cs}, \mathbf{Tr}) = \{\mathit{Cmb}_1, \mathit{Cmb}_2, \mathit{Cmb}_3, \mathit{Cmb}_4\}$,
where $[R_1, R_2, \mathbf{Cs}]$ is a conflict, and Cmb_3 and Cmb_4 may be empty rules.

The following properties must hold, with the last two being conditional upon Cmb_3 and Cmb_4 existing:

$$\forall\ o \in \mathbf{Tr} \cup \mathbf{P} \setminus \mathbf{Cs},\ \text{conclude}(R_1, o, \mathit{Class}_1) \Rightarrow \text{conclude}(\mathit{Cmb}_1, o, \mathit{Class}_1)$$
$$\text{conclude}(R_2, o, \mathit{Class}_2) \Rightarrow \text{conclude}(\mathit{Cmb}_2, o, \mathit{Class}_2)$$

$$\forall\ o \in \mathbf{Cs},\ \neg(\text{trigger}(\mathit{Cmb}_1, o) \vee \text{trigger}(\mathit{Cmb}_2, o))$$
$$\neg\ \exists\ o \in \mathbf{Tr} \cup \mathbf{P} \setminus \mathbf{Cs},\ \text{trigger}(\mathit{Cmb}_3, o) \vee \text{trigger}(\mathit{Cmb}_4, o),$$
$$\forall\ o \in \mathbf{Cs},\ \text{trigger}(\mathit{Cmb}_3, o) \oplus \text{trigger}(\mathit{Cmb}_4, o)$$

The notation $\mathbf{A} \setminus \mathbf{B}$ refers to the set of those objects in $\mathbf{A}$ which are not in $\mathbf{B}$ (i.e., set minus).

Cmb_1 and Cmb_2, then, are replacements for R_1 and R_2 and are called *replacement rules*, whilst Cmb_3 and Cmb_4 deal with the objects in conflict and are called the *introduced rules*. The replacement rules will give the same classification to those objects in $\mathbf{Tr}$ and $\mathbf{P}$, but not in $\mathbf{Cs}$ (i.e. $\mathbf{Tr} \cup \mathbf{P} \setminus \mathbf{Cs}$), as they previously received, whilst the introduced rules will only trigger on those objects in $\mathbf{Cs}$, and both cannot trigger on the same object.

$\mathbf{R}$ will consist of all rules in $\mathbf{R}_1$ and $\mathbf{R}_2$, except for those rules which are replaced by those returned by $\mathbf{MIL}_{cr}$.

3.3 The MIL Conflict Resolver

The basic steps of the $\mathbf{MIL}_{cr}$ algorithm are now described. Each step is briefly introduced, followed by a detailed description of the operations involved. The $\mathbf{MIL}_{cr}$ system firstly constructs candidate descriptions which describe some difference between the two training subsets $\mathbf{Tr}_1$ and $\mathbf{Tr}_2$. One of these candidates is then selected. The rules R_1 and R_2 are then modified (producing the replacement rules) and new rules are constructed (producing the introduced rules) using the selected description.

Constructing Candidate Descriptions

Given R_1, R_2, and $\mathbf{Tr}$, $\mathbf{MIL}_{cr}$ considers the subsets of $\mathbf{Tr}$ corresponding to the two given rules: namely $\mathbf{Tr}_1$ and $\mathbf{Tr}_2$. Candidate descriptions which differentiate between the objects in $\mathbf{Tr}_1$ and $\mathbf{Tr}_2$, and also partition the objects in $\mathbf{Cs}$, are then searched for. These candidate descriptions consist of pairs of condition triplets, each pair involving just a single attribute, with each attribute appearing in at most one pair of condition triplets. The first major task for $\mathbf{MIL}_{cr}$ can be summarised as:

Step 1. For each attribute in $\mathbf{A}$ attempt to construct a candidate description, consisting of a pair of condition triplets, $(Cons_1, Cons_2)$, such that $Cons_1$ is true for every object in $\mathbf{Tr}_1$, but not for any object in $\mathbf{Tr}_2$, and $Cons_2$ is true for $\mathbf{Tr}_2$, but not $\mathbf{Tr}_1$. Each candidate description must also partition $\mathbf{Cs}$.

The last requirement listed in Step 1 specifies that $Cons_1$ must describe a non-empty subset of $\mathbf{Cs}$, and $Cons_2$ must describe all the other objects in $\mathbf{Cs}$ not described by $Cons_1$, and must also be non-empty. Certain attributes can be eliminated from consideration even before we attempt to construct descriptions involving them. Any attribute appearing in a condition triplet of the form $[A_i, =, v]$ in either of $Cond_1$ or $Cond_2$, can be removed from consideration, since every object in $\mathbf{Cs}$ satisfies both $Cond_1$ and $Cond_2$, and so every object in $\mathbf{Cs}$ has the value v for A_i. Thus no description involving this attribute alone can partition $\mathbf{Cs}$.

For the remaining attributes, this step involves two cases, corresponding to the two types of attributes. For integer attributes, binary splits are looked for, whilst for categorical attributes, two disjoint sets of values of the attribute are constructed. These two cases are considered separately. V_1 and V_2 are used to denote the sets of values of an attribute associated with the two training sets. For attribute A_i, $V_1 = \{A_i(o) | o \in \mathbf{Tr}_1\}$, and $V_2 = \{A_i(o) | o \in \mathbf{Tr}_2\}$.

Case 1.1. A is an integer attribute: Find some value of A, γ say, such that all the values of A in $\mathbf{Tr}_1$ are less than γ and all values in $\mathbf{Tr}_2$ are greater than or equal to γ, or vice versa, and such that γ partitions

Cs. If no such γ can be found, then no candidate description is constructed for this attribute.

The two situations referred to in Case 1.1 (*viz.* $V_1 < V_2$ and $V_2 < V_1$) are symmetric, and so only the former is dealt with here. Let

$$\alpha = \max_{o \in \mathbf{Tr}_1} A(o), \text{ and } \beta = \min_{o \in \mathbf{Tr}_2} A(o).$$

That is, α is the maximum value of A in V_1, and β is the minimum value in V_2 and $\alpha < \beta$. We attempt to find some γ such that $\alpha < \gamma \leq \beta$. One such γ is $\frac{\alpha+\beta}{2}$, rounded to the nearest integer, *à la* the ID3 threshold. If the pair of condition triplets $Cons_1 = [A, <, \gamma]$ and $Cons_2 = [A, \geq, \gamma]$, describes two non-empty subsets (i.e. partitions) of **Cs**, then $(Cons_1, Cons_2)$ is a candidate description. However, if this γ does not partition **Cs**, then, if the maximum value of A in **Cs** is greater than α, and the minimum value is less than β, some other value for γ, with $\alpha < \gamma \leq \beta$, will. In this case, $\gamma = \alpha + 1$ if the minimum value of A in **Cs** is less than or equal to α, otherwise, $\gamma = \beta$. If no appropriate value for γ can be found, then no candidate description based on A is constructed.

Case 1.2. A is a categorical attribute: Find two sets of values of the attribute A such that every object in $\mathbf{Tr}_1$ has a value for A which is in one of the sets, and every object in $\mathbf{Tr}_2$ has a value which is in the other set. If no appropriate disjoint sets cannot be found, or if the sets of values do not partition **Cs**, then no candidate description is constructed for this attribute.

V_1 and V_2 are constructed as the sets of values of A found in $\mathbf{Tr}_1$ and $\mathbf{Tr}_2$ respectively. If the intersection of V_1 and V_2 is empty, then the triplets $Cons_1 = [A, \in, V_1]$ and $Cons_2 = [A, \in, V_2]$ are candidate descriptions if they partition **Cs**. Otherwise, no candidate description based on A is constructed.

Description Selection

The second task of $\mathbf{MIL}_{cr}$ involves choosing one description from the set of candidate descriptions. If no descriptions have been constructed in Step 1, though, then Step 2 does nothing.

The approach taken is based on the following heuristic:

Heuristic: Descriptions using integer attributes should be preferred to those using categorical attributes.

This heuristic results from the experiments described in Williams (1987). The underlying principle is that we wish to cover as many objects as possible. If a pair of condition triplets involves an integer attribute, every possible value of that attribute is covered by the description. This is not typically

true for categorical attributes, where the description only covers those values explicitly mentioned in the training set. Generalising this then, we want to select the description having the widest coverage of objects in **Tr**. We do this with respect to **Tr** since this set is presumed to be representative of the values of the attributes. Step 2 is thus:

Step 2. If the set of candidate descriptions is non-empty, then choose the description which accounts for the most objects in **Tr**.

This step clearly favours integer attributes. If there is still some choice available, then a prespecified ordering of the attributes is used.

Rule Construction

We now deal with the task of removing the conflict. This is done by firstly constructing the replacement rules for R_1 and R_2. If candidate descriptions have been constructed, the introduced rules are then constructed, based on the description selected in Step 2.

Step 3. Construct the replacement rules: Modify R_1 and R_2 such that neither triggers whenever the other does, and such that they still classify objects in **Tr** correctly.

Replacements rules satisfying these constraints are:

$$Cmb_1\colon\ Cond_1 \wedge \neg Cond_2 \Rightarrow Class_1$$
$$Cmb_2\colon\ Cond_2 \wedge \neg Cond_1 \Rightarrow Class_2$$

The conjunction $Cond_1 \wedge \neg Cond_2$ describes all those objects in **Tr** that are also described by $Cond_1$ alone, and similarly for $Cond_2 \wedge \neg Cond_1$. Hence Cmb_1 and Cmb_2 will classify the same objects in **Tr** as classified by R_1 and R_2. Further, $Cond_1 \wedge \neg Cond_2$ describes no object in **Cs**, but may describe objects in $\mathbf{P} \setminus \mathbf{Cs}$. These objects, though, will receive the same classification as given by R_1 previously. Similarly for R_2. Thus, by replacing R_1 and R_2 by Cmb_1 and Cmb_2, **Tr**-consistency is maintained, and classifications given to objects in $\mathbf{P} \setminus \mathbf{Cs}$ are unchanged.

Step 4. If candidate descriptions exist, then construct the introduced rules from the pair of conditions from the chosen description: $Cons_1$ and $Cons_2$.

The introduced rules are:

$$Cmb_3\colon\ Cond_1 \wedge Cond_2 \wedge Cons_1 \Rightarrow Class_1$$
$$Cmb_4\colon\ Cond_1 \wedge Cond_2 \wedge Cons_2 \Rightarrow Class_2$$

There are no objects in **Tr** for which $Cond_1 \wedge Cond_2$ holds, and so **R** remains **Tr**-consistent. Further, **Cs** contains all those objects in **P** for which $Cond_1 \wedge Cond_2$ holds, thus the effect of the above two rules is local to **Cs** only.

Finally, the new rules are returned:

Step 5. Return the rules Cmb_1,Cmb_2, Cmb_3, Cmb_4, where Cmb_3 and Cmb_4 exist only if candidate descriptions were constructed.

4. APPLYING THE MIL ALGORITHM

$\mathbf{MIL}_{cr}$ is now applied to the task of resolving a conflict which occurs when T106DC and T106DI are combined. Two rules which result in conflict are: R_1 from T106DC and R_2 from T106DI.

$$R_1\text{: Soil} = 26, \text{UVeg} = 2 \Rightarrow \mathit{VLow},$$
$$R_2\text{: Soil} = 26, \text{DPort} < 635 \Rightarrow \mathit{Medium}.$$

Table 2 lists all the objects from T106 which correspond to these two rules.

Region	Soil	UVeg	LVeg	DPort	AWMIH	AWMIS	AWMIW	Class
30503	26	2	4	801	16	16	09	*VLow*
21423	26	3	2	467	52	09	33	*Medium*
21424	26	3	2	469	49	09	29	*Medium*

TABLE 2: *The objects from the T106 training set associated with the two rules R_1 and R_2 are listed. The first object corresponds to R_1 (*Soil = 26 *and* UVeg = 2*) whilst the last two correspond to R_2 (*Soil = 26 *and* DPort < 635*).*

The conflict associated with the above two rules only exists when the condition

$$\text{Soil} = 26, \text{UVeg} = 2, \text{DPort} < 635$$

holds. In order to avoid both R_1 and R_2 firing for objects which satisfy this condition, these two rules are modified by adding appropriate conditions to their If-parts. The condition added to R_1 is the negation of the condition of R_2, which, after removing redundancies, is DPort $\geq$ 635. For rule R_2, we add the negation of the condition of R_1, which, with respect to the union of the corresponding training sets, is UVeg=3. The replacement rules are thus:

$$Cmb_1\text{: Soil} = 26, \text{UVeg} = 2, \text{DPort} \geq 635 \Rightarrow \mathit{VLow},$$
$$Cmb_2\text{: Soil} = 26, \text{DPort} < 635, \text{UVeg} = 3 \Rightarrow \mathit{Medium},$$

We now consider a number of objects which are in conflict with respect to the two rules R_1 and R_2. Eight such objects from the ARID database

Region	Soil	UVeg	LVeg	DPort	AWMIH	AWMIS	AWMIW
28542	26	2	3	630	20	17	08
30543	26	2	3	602	19	18	07
30552	26	2	3	517	22	18	06
32542	26	2	4	513	20	18	05
35523	26	2	4	621	16	15	03
45562	26	2	4	575	24	17	05
64372	26	2	10	539	19	08	09
65371	26	2	10	537	18	08	09

TABLE 3: *Eight objects from the ARID database with* Soil = 26, UVeg = 2, *and* DPort < 635 *are listed. Both rules* R_1 *and* R_2 *fire on these objects.*

are listed in Table 3, and we assume here that they belong to the pilot set. These objects form the conflict set, **Cs**.

The candidate descriptions must now be constructed. The discusion following step 1 of the algorithm above indicates that Soil and UVeg can be eliminated from consideration, since each object in the conflict set has a value of 26 for Soil and 2 for UVeg. Table 2 shows that LVeg may be a candidate, since all objects from $\mathbf{Tr}_1 \cup \mathbf{Tr}_2$ with LVeg=4 are classified as VLow, whilst those with LVeg=2 are Medium. However, **Cs** is not partitioned by these values of LVeg. For DPort the possible values of γ are 635, 470, and 801—none of which partition **Cs**. The values of γ for AWMIH are 33, 17, and 49, leading to the candidate description [AWMIH,<,17] [AWMIH,≥,17]. The only other candidate description is [AWMIS,≥,13] [AWMIS,<,13].

Of the two candidates, AWMIS is chosen, leading to the two introduced rules:

Cmb_3: Soil = 26, UVeg = 2, DPort < 635, AWMIS ≥ 13 ⇒ *VLow.*
Cmb_4: Soil = 26, UVeg = 2, DPort < 635, AWMIS < 13 ⇒ *Medium,*

With this modification (replacing R_1 and R_2 by Cmb_1, Cmb_2, and introducing Cmb_3, and Cmb_4), we apply the combined rule set to the ARID database. In the ARID database there is a total of 35 objects for which both R_1 and R_2 fired. Only two of these (regions 65371 and 64372 of Table 3) are classified by rule Cmb_3, whilst the others are classified by Cmb_4. These classifications can be compared to those given to these 35 objects by the regression model. It is found that 22 of the objects which were in moderate disagreement between T106DI and the model, are now in agreement with the model.

Table 4 presents the results of applying the combined rule set to the ARID database. This rule set consists of all the rules from both T106DC and T106DI, except that the rules R_1 and R_2 have been removed, and $Cmb_1, \ldots, Cmb_4$ have been added. Any remaining conflicts (from other rules) are treated as Null classifications (i.e., not assigning any class to the object). It is seen that, with just this one conflict removed, we have a rule set which has improved upon the coverage obtained by T106DC, whilst improving upon the accuracy of T106DI.

Experiment	Description	Cover	Agree	Mild	Mod	Strong
CombDCDIa	Modified Rule Set.	79.7	67.4	30.8	1.8	0.0
T106DC	Favour Categoricals	70.8	71.5	26.7	1.8	0.0
T106DI	Favour Integers	84.3	64.8	33.1	2.1	0.0

TABLE 4: *Comparison of the results from using the modified combined rule set (replacing R_1 and R_2 with Cmb_1, Cmb_2, and introducing Cmb_3 and Cmb_4) and the original decision trees. The conflicts yet to be dealt with by* **MIL** *in CombDCDIa are regarded as Null classifications.*

5. MORE RESULTS

Further experimentation with the **MIL** algorithm is in progress. Early results have confirmed that improvements, as observed in the previous example, occur in other applications. Here, we look at two instances where $\mathbf{MIL}_{cr}$ is applied to all conflicts occurring in the combined rule set, resulting in a conflict-free rule set.

These experiments deal with the task of predicting how well a student without English as a first language will perform at an Australian University. The data is completely fictitious, being derived from a simple equation representing the model. Four parameters are considered; Y is the number of years spent at school in Australia before starting University; A is the current age of the student; E is a rating of the English language ability of the student; and S is the matriculation score, or equivalent, of the student. Y and A are integer attributes, with Y ranging from 0 to 10, and A from 15 to 35. E and S are both 5-valued categorical attributes, having the values Very Poor (0), Poor (1), Ok (2), Good (3), and Excellent (4). U is the predicted performance, being one of three values: Poor, Ok, Good. The model computes U as:

$$U = 2 \times Y + 2 \times |25 - A| + 15 \times E + 25 \times S.$$

The three classes of U correspond to the three ranges 0...69, 70...139, and 140...200.

The database contains 5775 objects. Two random samplings of twenty objects each are used as the training sets for the two experiments, A and B. For each experiment, ID3 is used to construct two "equally good" rule sets. The two rule sets are then combined, using **MIL** to resolve conflicts. Tables 5 and 6 list the results. In the first case the coverage is the same as that of the best coverage of either rule sets (i.e., the same as AI), whilst the accuracy is better than that obtained by AI. The second case is very similar.

Experiment	Description	Cover	Agree	Mild	Mod
AC	Favour Categoricals	80.0	81.1	18.9	0.0
AI	Favour Integers	92.0	67.3	32.7	0.0
AComb	Combined Rule Set	92.0	72.5	27.4	0.0

TABLE 5: *The rule set AComb results from combining AC and AI, and then applying* $\mathbf{MIL}_{cr}$ *to all conflicts. No conflicts remain.*

Experiment	Description	Cover	Agree	Mild	Mod
BC	ID3 Favouring Categoricals	64.0	76.8	21.5	1.6
BI	ID3 Favouring Integers	96.0	49.9	44.8	5.4
BComb	Combined Rule Set	93.1	60.5	37.5	3.5

TABLE 6: *The rule set BComb results from combining BC and BI, and then applying* $\mathbf{MIL}_{cr}$ *to all conflicts. No conflicts remain.*

Further experiments, but using rule sets as models, are currently under way.

6. DISCUSSION

The **MIL** algorithm allows decision trees, as produced by ID3, to be combined. The preliminary experiments presented here have illustrated that **MIL** produces rule sets which approach (and may attain) both the coverage and the accuracy of the best of the ID3 decision trees it combines. That is, **MIL** appears to provide a sensible way of combining decision trees, producing more consistency in the resulting rule sets, with respect to accuracy and coverage, than the ID3 algorithm alone does.

MIL also provides a means for generating If-Then type rules from decision trees. There are many advantages with using rules rather than decision trees. Amongst these are that rules are the most popular form of knowledge

representation in expert systems, for good reasons (Waterman, 1986). Each rule represents a single chunk of knowledge which can be easily modified or removed, usually independently of other rules in the system. A decision tree does not provide the same degree of flexibility offered by rules. It is difficult for example to modify individual paths through a decision tree.

Quinlan (1986b) has also presented a technique for generating rules from decision trees. An initial rule set is generated from a single decision tree in the manner described at the beginning of Section 2. Each rule is then considered in turn to see if it can be generalised (simplified) by dropping conditions from its left-hand side. A statistical test (Fisher's exact test) is then used to determine if the hypothesis that the dropped condition is irrelevant to the conclusion of the rule can be rejected. A certainty factor is also associated with each rule. Quinlan concludes that this method has proved especially powerful.

Quinlan's method can be contrasted with **MIL**. Whereas Quinlan generalises rules, and thus possibly increasing the coverage of the rule set, **MIL** specialises rules. Consequently, **MIL** cannot produce a rule set with greater coverage than the best of the decision trees it combines. Also, both methods make more use of the data in the training set. Quinlan uses statistics to guide the generalisations, whilst **MIL** constructs further decision trees from the same data, adding more rules to the rule set. **MIL** though requires the existence of another set of objects—the pilot set—which is used to guide the specialisation and introduction of further rules. Finally, Quinlan comments on the need to be able to combine decision trees (rule sets). **MIL** represents an attempt to do this.

Whilst we have demonstrated the usefulness of the **MIL** algorithm, its full potential has yet to be explored. A number of limitations do exist. For example, **MIL** has only been used so far to combine two decision trees at a time. But ID3 may produce more than two equally good trees. The implications of combining more decision trees have not been explored.

It is important that we be able to combine rule sets in general. As well as the situation described here, where one induction algorithm produces a number of alternate rule sets, other possibilities for the application of **MIL** exist. For example, we may want to combine the "best" rule sets generated by a number of induction algorithms, or to combine an induced rule set with an expert-written rule set. Combining rule sets derived from different human experts may also be a candidate problem.

References

Cocks, K. D., Young, M. D., & Walker, P. A. (1986). Mapping relative viability prospects for pastoralism in Australia, *Agricultural Systems*, **20**: 175-193.

Michalski, R. S., Carbonell, J. G., & Mitchell, T . M. (1986). *Machine Learning: An Artificial Intelligence Approach*, Vol. 2, Kaufmann, Palo Alto, California.

Quinlan, J. R. (1986a). Induction of decision trees, *Machine Learning*, **1**(1): 81-106.

Quinlan, J. R. (1986b). Simplifying decision trees, *Knowledge Acquisition for Knowledge-Based Systems Workshop*, Banff, Canada.

Quinlan, J. R., Compton, P. J., Horn, K. A., & Lazarus, L. (1986). Inductive knowledge acquisition: A case study, in *Proceedings of the Second Australian Conference on Applications of Expert Systems*, New South Wales Institute of Technology.

Walker, P. A., Cocks, K. D., & Young, M. D. (1985). Regionalising continental data sets, *Cartography*, **14**(1): 66-73.

Williams, G. J. (1987). Some experiments in decision tree induction, *The Australian Computer Journal*, **19**(2): 84-91.