

**DECISION TREE GENERATION FROM EXAMPLES AND  
TENDENCY RULES**

**Scott B. Wilson**

**May 1988  
CSD-880043**



UNIVERSITY OF CALIFORNIA

Los Angeles

Decision Tree Generation from  
Examples and Tendency Rules

A thesis submitted in partial satisfaction of the  
requirements for the degree Master of Science  
in Computer Science

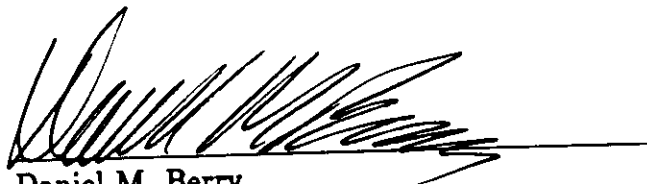
by

Scott B. Wilson


1985



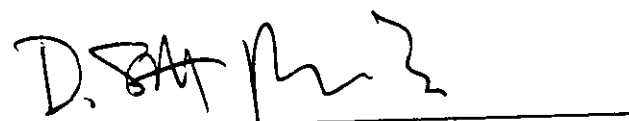
The thesis of Scott B. Wilson is approved.



Daniel M. Berry



David Martin



D. Stott Parker, Committee Chair

University of California, Los Angeles

1985

From wonder into wonder

Existence opens.

— Lao Tzu

## Table of Contents

1. Introduction	1
1.1 Previous Work	2
1.2 The Approach of the Thesis	3
2. Nomenclature	5
2.1 Objects and Examples	5
2.2 Decision Trees	6
2.3 An Example	10
3. Generalization of Examples	13
3.1 Types of Decision Trees	14
3.2 Significance of Decision Tree Types	15
4. Stepwise Generalization	17
4.1 Stepwise Generalization Algorithm	18
5. A Simple Example	23
6. Tendency Rules	28
6.1 Un-instantiated Tendency Rules	32
6.2 Tendency Rule Algorithm	33
7. A Simple Example Revisited	36
8. Secondary Classification Trees	41
8.1 The Problem	41
8.2 The Solution	42

9. Ordering the Attributes by Priority	44
10. Case Study — Hottest Stocks	45
10.1 Examples	46
10.2 Tendency Rules	47
10.3 Results	50
11. Conclusion	56
List of References	58



## ABSTRACT OF THE THESIS

Decision Tree Generation from  
Examples and Tendency Rules

by

Scott B. Wilson

Master of Science in Computer Science

University of California, Los Angeles, 1985

Professor D. Stott Parker, Chair

A decision tree is a model for the type of classification performed by many of today's expert systems. It is usually the task of a human expert to provide the necessary knowledge for the expert system. Another approach, however, is to extract the necessary knowledge from examples. This work tries to combine these two approaches by providing a method for automatically generating decision trees from examples while utilizing some human knowledge in the form of tendency rules. It is hoped that the resulting decision trees will be better classifiers than the decision trees built from examples alone.

The tendency rule

$$\textit{attribute}(v) \rightsquigarrow \textit{class}(x)$$

says that "the value  $v$  on attribute  $\textit{attribute}$  'tends to imply' the classification value  $x$  for the classification concept  $\textit{class}$ ." Tendency rules carry semantic

information about the concept to be classified, and constrain the structure of the resulting decision tree.

The algorithm presented in this paper was implemented in Prolog and used to build a decision tree for classifying companies according to their stock-price-gains. A set of sixteen tendency rules was constructed for this problem, and decision trees were generated with and without the aid of these rules. From 5% to 20% more correct classifications were made by the trees generated with the aid of tendency rules.

## 1. Introduction

The classification of objects is often performed by considering a finite, and not necessarily complete, set of relevant data. Say for instance that we are trying to cross the street. We would like to classify the current situation as either "safe" or "dangerous." Relevant information could include whether or not the light in our direction is green, whether the approaching cars are moving fast or slow, and whether or not we have our running shoes on. Depending on the answers to these questions we will decide that it's "safe" to cross now or that it's "dangerous" and that we should wait.

A model for this type of classification-scheme is the *decision tree*. Each node in the tree is associated with a question about the object to be classified. The tree is traversed from the root node to a leaf node that contains the name of the appropriate class. The traversal is achieved by following the branches indicated by the answers to the questions in the nodes. A simple medical diagnosis decision tree could have symptoms in the nodes, branches labeled "yes" or "no" indicating that the patient did or did not exhibit a particular symptom, and the names of the possible diseases in the leaves of the tree.

An expert system can loosely be defined as a computer program that performs classifications in a narrow field nearly as well or better than a human expert in that field. Currently almost all expert systems are built from rules extracted from human experts. The process of obtaining these rules from human experts is often difficult because few experts can verbalize their decision-

making process in the straightforward and logical manner required by current knowledge-engineering languages. One way to circumvent this difficulty is to automatically generate decision trees from example objects for which the classes are known.

### 1.1 Previous Work

Surprisingly little work has been done on the automatic generation of decision trees from examples. The most notable recent work along these lines is that of Quinlan<sup>[1]</sup> using Hunt's<sup>[2]</sup> CLS (Concept Learning System) as a starting point. The commercially available expert-system-builder RuleMaster<sup>[3]</sup>, which uses Quinlan's algorithm, has recently been used to build a weather forecasting program that reportedly performs as well as experts in the field.

Quinlan's algorithm attempts to produce the "simplest" decision tree consistent with the examples. (There will in general be many decision trees consistent with the examples.) According to Quinlan, "Simplicity has a clear relationship to generality, and it seems desirable to find rules that are applicable to more instances than those used to build them [1]." His algorithm utilizes a heuristic, which relates the "size" of a rule to a collection of examples, to produce a "simple" rule.

Quinlan's approach works quite well when the size of the training set, the set of objects whose classes are known *a priori*, approaches that of the set of all possible objects. If the training set is small, then the resulting decision tree is

very dependent on the particular objects in the training set. A poor training set can produce a very poor decision tree.

## 1.2 Approach of the Thesis

Perhaps better decision trees can be generated by not leaving the human expert totally out of the system. If we are presented with a number of decision trees that are consistent with the known examples, then we would like to choose that tree which is the best classifier, i.e. that tree that will correctly classify the greatest number of future objects. This work proposes the use of human knowledge, in the form of tendency rules, as a means for choosing the best classifier.

It is the author's belief that a basic type of human knowledge is a *tendency*. We might say that "an animal with large teeth *tends* to be dangerous." Tendencies are often based on some type of causal knowledge, e.g. "the fact that the animal has large teeth means that it can take a large bite."

Consider now the following rule:

*teeth(large) ~> safeness(dangerous).*

This rule reflects the tendency knowledge above where  $\sim>$  is interpreted as "tends to imply." It does not supply any information about the chances of a particular animal being safe or dangerous, yet such tendency rules can be used to help produce better decision trees.

This paper introduces an algorithm that uses a set of tendency rules, which are supplied by an expert, and a set of examples to produce a decision tree. It is hoped that the resulting decision tree will be a better classifier than a tree generated without the tendency rules.

## 2. Nomenclature

This work is concerned with generating decision trees that assign classification values to objects according to their descriptions. The usefulness of the resulting decision tree is very dependent on the properties used to describe the objects. For example, it is much more reasonable to describe chess board configurations in terms of properties like "black king is immediately next to the rook" than the conventional chess notation that describes the placement of the king and rook on the board [1].

Here we assume that we are presented with examples having descriptions, in the form of attribute lists. The task of defining object descriptions is that of the human expert, and no attempt is made here to modify the descriptions.

The definitions in sections 2.1 and 2.2 define a vocabulary that is useful in understanding the decision tree generation algorithms presented in this paper.

### 2.1 Objects and Examples

#### Definition

An *object*  $O$  is a tuple  $\langle id_o, \vec{a}_o \rangle$ .  $id_o$  is the *unique identifier* (name) of  $O$  and  $\vec{a}_o$  its *attribute list* (description).

#### Definition

An *example*  $I$  is a tuple  $\langle id_I, \vec{a}_I, x \rangle$ .  $id_I$  is the *unique identifier* of  $I$ ,  $\vec{a}_I$  its *attribute list*, and  $x$  its *classification value*.

### **Definition**

The *training set*  $S$  is the set of examples that will be used to generate the decision tree.

It is important that no two examples have equal attribute lists and different classification values. In such a case the training set and set of attributes are inconsistent, and either one of the two conflicting examples must be removed or a new set of attributes that resolves this conflict must be found.

## **2.2 Decision Trees**

Decision trees contain two types of nodes: *attribute nodes*, which contain an attribute name, and *leaf nodes*, which contain a classification value. The usual parent-child relationship exists between the nodes. The root node is at the top of the tree, and the leaf nodes are at the bottom of the tree. The branch between a parent and a child is labeled with an attribute value appropriate to the parent.

The definitions below will allow decision trees to be defined in terms of formulas connected by the logical conjunction and disjunction operators.

### **Definition**

An *attribute predicate* is a unary predicate represented by the name of an attribute.



**Definition**

An *attribute value* is a constant that may appear as the argument to an attribute predicate.

**Definition**

The *attribute formula*  $a(v)$  consists of the attribute predicate  $a$  and the attribute value  $v$ .

**Definition**

The *attribute list*  $\vec{a}$  is a set of attribute formulas.

**Definition**

$a(v)$  is **true** for object  $O$  if  $a(v) \in \vec{a}_O$  where  $\vec{a}_O$  is the attribute list of  $O$ .

**Definition**

A *path*  $p$  is the logical conjunction of attribute formulas of the form  $a(v)$ . Letting  $a^i(v^i)$  represent the  $i^{\text{th}}$  formula in this path, then

$$p = a^1(v^1) \wedge a^2(v^2) \wedge \dots \wedge a^N(v^N)$$

where  $a^i$  is the parent of  $a^{i+1}$  for  $1 \leq i < N$ , and there is a branch from  $a^i$  to  $a^{i+1}$  labeled  $v^i$ .

**Definition**

A *leaf predicate* is a unary predicate that is represented by the name *leaf*.

**Definition**

A *classification value* is a constant that may appear as an argument to the predicate *leaf*.

**Definition**

The formula  $leaf(x)$  consists of the attribute predicate *leaf* and the classification value  $x$ .  $leaf(x)$  is always **true**.

**Definition**

A *completed path* is a path

$$p_c = a^1(v^1) \wedge a^2(v^2) \wedge \dots \wedge a^N(v^N) \wedge leaf(x)$$

that is terminated with the formula  $leaf(x)$ .

**Definition**

The completed paths

$$a^1(v^1) \wedge a^2(v^2) \wedge \dots \wedge a^N(v^N) \wedge leaf(x)$$

and

$$b^1(w^1) \wedge b^2(w^2) \wedge \dots \wedge b^M(w^M) \wedge leaf(y)$$

are *connected* if  $a^i = b^i$  and  $v^{i-1} = w^{i-1}$  for  $i \geq 1$ . These connected paths have the logical interpretation

$$\begin{aligned} & a^1(v^1) \wedge a^2(v^2) \wedge \dots \wedge a^{i-1}(v^{i-1}) \wedge \\ & (a^i(v^i) \wedge a^{i+1}(v^{i+1}) \wedge \dots \wedge a^N(v^N) \wedge leaf(x)) \\ & \vee (a^i(w^i) \wedge b^{i+1}(w^{i+1}) \wedge \dots \wedge b^M(w^M) \wedge leaf(y)). \end{aligned}$$

(It is this interpretation that suggests the rule format for decision trees shown in section 2.3.) A path is *connected* with itself.

### Definition

A *decision tree*  $T$  is a set of one or more connected paths.

### Definition

The *traversal of tree*  $T$  with object  $O$  is the completed path

$$p_o = a^1(v^1) \wedge a^2(v^2) \wedge \dots \wedge a^N(v^N) \wedge leaf(x)$$

where  $p_o \in T$ . The path leads from the root node  $a^1$  to the leaf node  $leaf$ , and  $a^i(v^i)$  is **true** for  $O$  with  $1 \leq i \leq N$ . (There is at most one such path for

any object.) The *traversal of tree  $T$  with object  $O$*  is interpreted as the logical conjunction of formulas  $a^1(v^1) \wedge a^2(v^2) \wedge \dots \wedge a^N(v^N)$  that is **true** for  $O$ .

**Definition**

The formula *leaf( $x$ )* ascribes the classification value  $x$  to object  $O$  in the traversal of tree  $T$  with object  $O$ .

**Definition**

The traversal of tree  $T$  with object  $O$  *fails* if there is no completed path in  $T$  that is **true** for  $O$ . (A traversal which fails can be interpreted to ascribe the classification value *unknown* to object  $O$ .)

**2.3 An Example**

In a later section we will build a decision tree to determine the ability of a given dog to inflict bodily damage upon us. Some things that we might wish to consider are whether or not the dog is leashed, whether its teeth are large or small, and whether or not it is foaming at the mouth. Let the following table be our training set

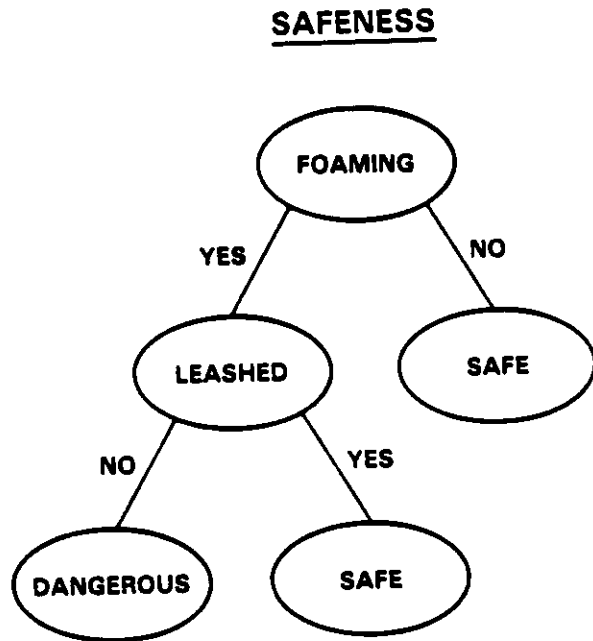
Name	leashed	teeth	foaming	Class
spot	no	large	no	safe
fang	no	small	yes	dangerous
spike	yes	small	yes	safe

which when rewritten as examples becomes

$\{ \langle \text{spot}, \{ \text{leashed}(\text{no}), \text{teeth}(\text{large}), \text{foaming}(\text{no}) \}, \text{safe} \rangle, \langle \text{fang}, \{ \text{leashed}(\text{no}), \text{teeth}(\text{small}), \text{foaming}(\text{yes}) \}, \text{dangerous} \rangle, \langle \text{spike}, \{ \text{leashed}(\text{yes}), \text{teeth}(\text{small}), \text{foaming}(\text{yes}) \}, \text{safe} \rangle \}$ .

*spot*, *fang*, and *spike* are the unique identifiers of the examples. The attribute predicates are *leashed*, *teeth*, and *foaming*. The attribute values associated with the attribute predicate *leashed* are *yes* and *no*. The example with  $id = \text{spot}$  has the attribute list  $\vec{a} = \{ \text{leashed}(\text{no}), \text{teeth}(\text{large}), \text{foaming}(\text{no}) \}$ , and the classification value  $x = \text{safe}$ .

Below is a decision tree consistent with the training set above. This tree says that "all dogs that are not foaming at the mouth are safe, while dogs that are foaming at the mouth are safe if they are leashed and dangerous if they are not leashed."



For the remainder of this paper decision trees will be displayed in a rule format. The rule equivalent to the decision tree above is

```

safeness
foaming(yes)
    leashed(no)
        leaf(dangerous)
    leashed(yes)
        leaf(safe)
foaming(no)
    leaf(safe).

```

This decision tree is the set of connected paths

$$\begin{aligned}
 \text{safeness} = \{ & \text{foaming(no)} \wedge \text{leaf(safe)}, \\
 & \text{foaming(yes)} \wedge \text{leashed(no)} \wedge \text{leaf(dangerous)}, \\
 & \text{foaming(yes)} \wedge \text{leashed(yes)} \wedge \text{leaf(safe)} \}.
 \end{aligned}$$

### 3. Generalization of Examples

If we are presented with a collection of objects and their classification values, then we assume that there is an underlying principal or rule that governs the classification. The aim of this work is to reproduce this rule from a set of examples. The particular type of rule considered in this work is the decision tree.

It is possible to construct many decision trees consistent with a finite number of examples. The problem of generalization from examples is consequently twofold. How do we construct the decision trees consistent with the known examples, and which of these trees do we use for future classifications?

Assume, for the moment, that an algorithm exists that has as its input the training set  $S$  and as its output the set of all decision trees consistent with  $S$ . Let us consider the criterion for choosing the the tree from the output of this algorithm that we will use for future classifications. The most important property that a decision tree can posses is the ability to correctly classify future objects. Tendency rules, which will be introduced later, can be used to address this aspect of the decision tree selection. A second desirable property is the ability to classify objects with descriptions (attribute lists) that differ from those of the objects in the training set  $S$  — this is the process of generalization. The last property of interest is that the tree does not contain any nodes that are not essential for classifying the objects in the training set. These last two properties are somewhat overlapping, and will be discussed in greater detail in the next

two sections.

### 3.1 Types of Decision Trees

Let us consider the types of decision trees that would be generated by an algorithm that has as its output the set of all decision trees consistent with the training set  $S$ .

#### **Definition**

A *minimal classifying tree* is a tree in which each path contains the complete attribute list of an example.

#### **Definition**

A *maximal classifying tree* is a tree in which every attribute node has an outgoing branch, which is contained in a path to a leaf node, for every possible value of that attribute.

#### **Definition**

A *partially irredundant attribute node* is a node that has at least two subtrees from which it is possible to select two leaf nodes, one from each subtree, that contain different classification values.

#### **Definition**

A *fully irredundant attribute node* is a node that has at least two subtrees that if



switched, without changing the labels on the outgoing branches from the node, would cause a misclassification of one or more of the examples in the training set used to build the decision tree. A fully irredundant attribute node is a partially irredundant attribute node.

### **Definition**

A *partially irredundant tree* is a tree that contains only partially irredundant attribute nodes.

### **Definition**

A *fully irredundant tree* is a tree that contains only fully irredundant attribute nodes.

## **3.2 Significance of Decision Tree Types**

A minimal classifying tree can only classify objects which have an attribute list equal to that of one of the examples used to build the tree (all other traversals fail). A maximal classifying tree has the desirable property of being able to classify every possible object.

A fully irredundant tree has the desirable property that no nodes other than those needed to resolve the classification are present in the tree. Unfortunately, whether or not a tree is fully irredundant can not be determined until after the tree has been generated. It is, as we shall see, always possible to create a

partially irredundant tree. This tree has the property that every attribute node is at least superficially involved in differentiating objects (unlike the minimal classifying tree).

The best possible tree, within the context of this discussion, would be a fully irredundant maximal classifying tree. The algorithm for decision tree generation in the next section will always produce a partially irredundant tree. After the tree is generated redundant nodes can be eliminated, although every tree generated in the course of this work has been a fully irredundant tree. Whether or not a maximal classifying tree can be generated is dependent on the examples in the training set. Again, however, every tree generated in the course of this work has been a maximal classifying tree.

#### 4. Stepwise Generalization

Described here is method for generating decision trees from examples. The Stepwise Generalization Algorithm proceeds in the following way: given a tree consistent with the  $N$  examples read so far, if the classification of example  $N + 1$  is inconsistent with the current tree, then the tree is refined by including more attribute tests that resolve the conflict.

The intuitive reasoning behind this type of tree generation is the following. We assume that the decision tree that we have at any given time is only an approximation to the actual classification rule. We expect to encounter new examples in the future that conflict with our current tree. Consequently, our tree will have to be refined in some way that covers the new example, as well as all those examples that preceded it.

The way in which the tree is refined can be understood by considering the following scenario: Let example  $I_x = \langle id_x, \bar{a}_x, x \rangle$  have the traversal

$$a^1(v^1) \wedge a^2(v^2) \wedge \dots \wedge a^N(v^N) \wedge leaf(y, \{id_y\})$$

of  $T_{current}$  with  $x \neq y$ . Leaf nodes, for the purpose of generating the decision tree, now contain a list of unique identifiers as well as a classification value. The *id\_list* of this leaf,  $\{id_y\}$ , references the single example  $I_y$ . (Examples  $I_x$  and  $I_y$  have the same description in the current decision tree.) If the attribute list of  $I_y$  is  $\bar{a}_y$ , then the classification conflict can be resolved by testing any attribute

in the set

$$\{a: a(v) \in \bar{a}_x, a(w) \in \bar{a}_y, v \neq w\}.$$

If more than one example is present in the *id\_list* of the leaf, then the conflict resolution may require the testing of more than one of the attributes in the set above.

#### 4.1 Stepwise Generalization Algorithm

When, during the decision tree generation, a new example is read from  $S$  a check is made to determine the classification value ascribed to this example by the current tree. One of three situations may arise: the tree ascribes the correct classification value to the example, the tree ascribes the classification value *unknown* to the example (the traversal of the tree fails), or the tree ascribes an incorrect classification value to the example. The first two situations are handled by the Decision Tree Population algorithm, which is called from within the Decision Tree Generation by Stepwise Generalization algorithm. This algorithm simply populates a leaf node of the current tree with the unique identifier of the new example. If the call to the Decision Tree Population algorithm fails, i.e. the tree ascribes an incorrect classification value to the example, then the Stepwise Generalization algorithm proceeds to refine the current tree in the manner discussed above.

Let  $\Lambda$  be the null tree, a null set of completed paths. The traversal of  $\Lambda$  on every object fails. The input to the Stepwise Generalization algorithm is the

decision tree  $A$  and the training set  $S$  of examples.

The following definitions will be used in the following algorithms.

**Definition**

The *length of path*  $p$  is the number of attribute formulas in the path.

**Definition**

The *power set*  $P$  of paths formed from a set  $S$  of attribute formulas by forming a path from each non-empty set in  $2^S$ . A path  $p$  is formed from a set of predicate formulas  $A$  by taking the logical conjunction of every element in  $A$ , that is  $p = \bigwedge_{i=1}^N a^i(v^i)$  where  $A = \{a^1(v^1), a^2(v^2), \dots, a^N(v^N)\}$ .

**Definition**

A *classification conflict* occurs when a traversal of tree  $T$  with example  $I$  ascribes an incorrect classification value to  $I$ . That is, given the traversal

$$a^1(v^1) \wedge a^2(v^2) \wedge \dots \wedge a^N(v^N) \wedge leaf(y)$$

and example  $I = \langle id_I, \bar{a}_I, x \rangle$ , then  $y \neq x$ .

**Definition**

A decision tree is *consistent with the examples* if it ascribes the correct classification value to every example in the training set.

**Algorithm:** Decision Tree Population.

*Input:* a decision tree  $T_{in}$  and a set of examples  $S$ .

*Output:* a decision tree  $T_{out}$  consistent with  $S$ , or a statement that the algorithm failed.

*Method:*

1) Read the next example  $I = \langle id_I, \vec{a}_I, x \rangle$  from  $S$  and let  $T_{old} := T_{in}$ . If there are no more examples in  $S$  then let  $T_{out} := T_{in}$ .

2) Find the traversal  $p_I$  of  $T_{old}$  with  $I$ .

2.1) If the traversal fails, then find the path  $p$  in  $T_{old}$  such that  $p = p_{sub} \wedge a^i(v^i) \wedge \dots \wedge a^N(v^N) \wedge leaf(y, id\_list)$ , and  $p_{sub}$  is the longest subpath that is **true** for  $I$ . (A zero-length path is **true** for every example.) Find the new  $v^i$  such that  $a^i(v^i) \in \vec{a}_I$ , and form the connected path  $p_I = p_{sub} \wedge a^i(v^i) \wedge leaf(x, \{id_I\})$ .  $T_{in}$  is  $T_{old}$  with the addition of  $p_I$ .

2.2) If the traversal does not fail, then it will have the form  $p_I = a^1(v^1) \wedge a^2(v^2) \wedge \dots \wedge a^N(v^N) \wedge leaf(y, id\_list)$ .

2.2.1) If  $y = x$ , then  $T_{in}$  is  $T_{old}$  with  $id_I$  added to  $id\_list$ . Go to 1).

2.2.2) If  $y \neq x$ , then there is a classification conflict. Return **fail**.

**Algorithm:** Decision Tree Generation by Stepwise Generalization.

*Input:* a decision tree  $T_{in}$  and a training set  $S$ .

*Output:* a decision tree  $T_{out}$  consistent with  $S$ .

*Method:*

1) Read the next example  $I = \langle id_I, \bar{a}_I, x \rangle$  from  $S$  and let  $T_{old} := T_{in}$ . If there are no more examples in  $S$  then let  $T_{out} := T_{in}$ .

2) Call the Decision Tree Population algorithm with input tree  $T_{old}$  and set of examples  $\{I\}$ .

2.1) If the algorithm does not return **fail**, then  $T_{in}$  is the output of the algorithm.

Go to 1).

2.2) If the algorithm does return **fail**, then there is a classification conflict that must be resolved. Find the traversal of  $T_{old}$  with  $I$ .  $p_I = a^1(v^1) \wedge a^2(v^2) \wedge \dots \wedge a^N(v^N) \wedge leaf(y, id\_list)$ , where  $y \neq x$ .

2.2.1) Form the power set  $P$  of paths from the set,

$\{a^j(v^j) \in \bar{a}_I: a^j(v^j) \notin \{a^1(v^1), a^2(v^2), \dots, a^N(v^N)\}\}$ , of yet untested attributes.

2.2.1.1) Remove the shortest path  $p$  from  $P$  and append the leaf predicate  $leaf(x, \{id_I\})$  forming completed path  $p_c = p \wedge leaf(x, \{id_I\})$ . Call the Decision Tree Population algorithm with the input tree  $\{p_c\}$  and the set examples whose unique identifiers are in  $id\_list$ .

2.2.1.1.1) If the algorithm does not return **fail**, then let  $T_{sub}$  be the output of the algorithm.  $T_{in}$  is  $T_{old}$  with the traversal  $p_I$  replaced by the set of connected paths  $\{p: p = a^1(v^1) \wedge a^2(v^2) \wedge \dots \wedge a^N(v^N) \wedge p_{sub}, p_{sub} \in T_{sub}\}$ . Go to 1).

2.2.1.1.2) If the algorithm does return **fail**, then return to 2.2.1.1).

**Theorem 1**

The decision tree generated by the Stepwise Generalization Algorithm with input  $T_{in} = \Lambda$  and training set  $S$  is a partially irredundant tree consistent with  $S$ .

**Proof**

The unique identifier of every example in  $S$  is contained in one of the leaf nodes of  $S$ . The traversal of  $T_{out}$  with example  $I$  is the path from the root node to the leaf node that contains the unique identifier of  $I$  and the classification value of  $I$ . Consequently,  $T_{out}$  is consistent with  $S$ . We can rephrase the definition of a partially irredundant attribute node to say that it is a node from which there are at least two paths, containing different children, that terminate in leaf nodes containing different classification values. Since only attributes used to resolve a classification conflict are found in  $T_{out}$ , there are at least two paths from every attribute node, containing different children, which terminate in leaf nodes with different classification values. Thus,  $T_{out}$  is a minimal covering tree consistent with  $S$ .  $\square$



## 5. A Simple Example

The Stepwise Generalization Algorithm can be used to generate a decision tree that addresses the ability of a given dog to inflict bodily damage upon us. Recall the training set from section 2.3:

$$\begin{aligned} & \{ \langle \textit{spot}, \{ \textit{leashed}(\textit{no}), \textit{teeth}(\textit{large}), \textit{foaming}(\textit{no}) \}, \textit{safe} \rangle, \\ & \langle \textit{fang}, \{ \textit{leashed}(\textit{no}), \textit{teeth}(\textit{small}), \textit{foaming}(\textit{yes}) \}, \textit{dangerous} \rangle, \\ & \langle \textit{spike}, \{ \textit{leashed}(\textit{yes}), \textit{teeth}(\textit{small}), \textit{foaming}(\textit{yes}) \}, \textit{safe} \rangle \}. \end{aligned}$$

The *safeness* decision tree is initially  $\Lambda$ . The first example is read and the Population algorithm is called. The traversal of the tree *fails*, so  $\Lambda$  is populated and replaced by the tree below

$$(T1) \quad \begin{array}{c} \underline{\textit{safeness}} \\ \textit{leaf}(\textit{safe}, \{ \textit{spot} \}). \end{array}$$

If no more examples were encountered, all future dogs would be classified as *safe*. The second example is read, and the tree ascribes the classification value *safe* to *fang*. *fang*, however, is *dangerous*, and a classification conflict has occurred. We form the power set  $P$  of paths from the untested attribute predicates in *fang*'s attribute list.

$$\begin{aligned} P = \{ & \textit{leashed}(\textit{no}), \textit{teeth}(\textit{small}), \textit{foaming}(\textit{yes}), \\ & \textit{leashed}(\textit{no}) \wedge \textit{teeth}(\textit{small}), \textit{leashed}(\textit{no}) \wedge \textit{foaming}(\textit{yes}), \\ & \textit{teeth}(\textit{small}) \wedge \textit{foaming}(\textit{yes}), \\ & \textit{leashed}(\textit{no}) \wedge \textit{teeth}(\textit{small}) \wedge \textit{foaming}(\textit{yes}) \}. \end{aligned}$$

At this point we can choose one of the three length 1 paths from  $P$ . Taking the first, we form the completed path  $p_c = leashed(no) \wedge leaf(dangerous, \{fang\})$ .

We now call the Population algorithm with the input tree

safeness  
*leashed(no)*  
*leaf(dangerous, {fang})*

and training set

$\{ \langle spot, \{ leashed(no), teeth(large), foaming(no) \}, safe \rangle \}$ .

The tree above is traversed with *spot* and a classification conflict occurs, so this call to the Population algorithm fails. We return to the Stepwise Generalization algorithm and select the next path from  $P$ . We call the Population algorithm again as we did above, but with the input tree

safeness  
*teeth(small)*  
*leaf(dangerous, {fang})*

and training set

$\{ \langle spot, \{ leashed(no), teeth(large), foaming(no) \}, safe \rangle \}$ .

The traversal on the tree above with *spot* fails at the attribute *teeth*, and a new branch is added resulting in the tree

safeness  
teeth(*small*)  
    *leaf(dangerous, {fang})*  
teeth(*large*)  
    *leaf(safe, {spot})*.

The subtree above replaces the leaf of (T1) where the classification conflict occurred, and the resulting tree is

(T2)

safeness  
teeth(*small*)  
    *leaf(dangerous, {fang})*  
teeth(*large*)  
    *leaf(safe, {spot})*.

The last example is read, and the tree incorrectly ascribes the classification value *dangerous* to *spike*. We form the power set  $P$  of paths from the untested attribute predicates in *spike*'s attribute list.

$$P = \{leashed(yes), foaming(yes), \\ leashed(yes) \wedge foaming(yes)\}.$$

Choosing the length 1 path *leashed(yes)* from  $P$ , we form the completed path *leashed(yes)  $\wedge$  leaf(safe, {spike})*. We now call the Population algorithm with the input tree

safeness  
leashed(*yes*)  
    *leaf(safe, {spike})*

and training set

{<*fang*, {*leashed(no)*, *teeth(small)*, *foaming(yes)*}, *safe*>}.

The traversal on the tree above with *fang* fails at the attribute *leashed*, and a new branch is added resulting in the tree

*safeness*  
*leashed(no)*  
    *leaf(dangerous, {fang})*  
*leashed(yes)*  
    *leaf(safe, {spike})*.

This subtree now replaces the leaf of (T2) where the conflict occurred, and the resulting tree with the example lists removed from the leaves is

(T3)           *safeness*  
                *teeth(small)*  
                    *leashed(no)*  
                        *leaf(dangerous)*  
                    *leashed(yes)*  
                        *leaf(safe)*  
                *teeth(large)*  
                    *leaf(safe)*.

This tree has the following interpretation:

(I1) "All dogs that have large teeth are safe, while dogs that have small teeth are dangerous if they are not leashed and safe if they are leashed."

Another tree would have been generated had the path *foaming(yes)* been removed from *P* before *teeth(small)*. Its interpretation is the following:

(I2) "All dogs that are not foaming at the mouth are safe, while dogs that are foaming at the mouth are safe if they are leashed and dangerous if they are not leashed."

Interpretation (I1) seems to run counter to what we would normally consider a dangerous dog, although the inconsistency is not apparent in the particular examples in the training set. Interpretation (I2) more closely defines what we would normally consider a good classification of a dangerous dog. In the next section we will see that a set of tendency rules can be applied to the decision tree generation algorithm that allows only the tree with interpretation (I2) to be generated.

## 6. Tendency Rules

The results of the previous example would have been acceptable if the space of trees had somehow included the tree with interpretation (I2), but not the tree corresponding to interpretation (I1). Let us consider adding some *human knowledge* in the form of tendency rules to the system.

What information would we like to be able to write down for use in our expert system for classifying dogs? We would like to say that "a leashed dog tends to be safe (because it will have to chew through its leash before being able to attack us)." Unfortunately, logic does not provide a way to state knowledge about tendencies short of introducing a new predicate *tends*.

Consider the introduction of the following "tendency operator"

$$\sim \rightarrow \equiv \text{"tends to imply"}$$

where

$$a(v) \sim \rightarrow \text{class}(x)$$

can be interpreted as "all other things being equal, the value  $v$  on attribute  $a$  'tends to imply' the classification value  $x$  for the classification concept *class*." This definition is purposely vague here in order to suggest how it relates to the human knowledge that it expresses. A formal definition of the tendency rule is given shortly.

Now we can use the tendency rule

$$\textit{leashed}(\textit{yes}) \sim \> \textit{safeness}(\textit{safe})$$

to express the knowledge "a leashed dog tends to be safe." Normally implied by the statement "a leashed dog tends to be safe" is the statement that "an unleashed dog tends to be dangerous", which has the corresponding tendency rule

$$\textit{leashed}(\textit{no}) \sim \> \textit{safeness}(\textit{dangerous}).$$

Say now that we are presented with two dogs that are identical except that one is leashed and the other is not. The tendency rules allow us to say that "it is likely that the unleashed dog is *more* dangerous than the leashed dog." The tendency rules do not, however, allow us to say that the leashed dog is safe and the unleashed dog dangerous. It may very well be the case that both dogs are perfectly safe.

### **Definition**

A *tendency rule* satisfies the following properties:

Given the tendency rule

$$a(v) \sim \> \textit{class}(x)$$

and attribute lists  $\bar{a}_1$  and  $\bar{a}_2$  which differ only in the value of attribute  $a$  such that  $a(v) \in \bar{a}_1$  and  $a(v) \notin \bar{a}_2$ ,

then

$$P(\text{class}(x) | \vec{a}_1) > P(\text{class}(x) | \vec{a}_2)$$

where  $P(\text{class}(x) | \vec{a}_1)$  is the probability, given  $\vec{a}_1$ , that the classification value of object  $O_1$  with attribute list  $\vec{a}_1$  is  $x$ .

Given the tendency rules

$$a(v) \rightsquigarrow \text{class}(x)$$

$$a(w) \rightsquigarrow \text{class}(x)$$

with  $a(v) \in \vec{a}_1$  and  $a(w) \in \vec{a}_2$ ,

then nothing is implied about the relation between  $P(\text{class}(x) | \vec{a}_1)$  and  $P(\text{class}(x) | \vec{a}_2)$ .

Tendency rules will now play a key role in resolving *classification conflicts*. The way that these will be utilized is best understood by considering the following scenario. Let example  $I_x = \langle id_x, \vec{a}_x, x \rangle$  have the traversal

$$a^1(v^1) \wedge a^2(v^2) \wedge \dots \wedge a^N(v^N) \wedge \text{leaf}(y, \{id_y\})$$

of  $T_{\text{current}}$  with  $x \neq y$ . (Examples  $I_x$  and  $I_y$  have the same description in the current decision tree.) When no tendency rules are known, if the attribute list of  $I_y$  is  $\vec{a}_y$ , then the classification conflict can be resolved by testing any attribute in the set

$$\{a: a(v) \in \vec{a}_x, a(w) \in \vec{a}_y, v \neq w\}.$$



When tendency rules are known, we choose an attribute which implies

$$P(\text{class}(x) | \bar{a}_x) > P(\text{class}(x) | \bar{a}_y)$$

reflecting the knowledge

$$P(\text{class}(x) | \bar{a}_x) = 1$$

$$P(\text{class}(x) | \bar{a}_y) = 0$$

contained in the examples. Formally, given a set of tendency rules  $R$  the classification conflict can be resolved by testing any attribute in the set

$$\{a: a(v) \in \bar{a}_x, a(w) \in \bar{a}_y, v \neq w, a(v) \rightsquigarrow \text{class}(x) \in R\}.$$

If more than one example is present in the *id.list* of the leaf, then the conflict resolution may require the testing of more than one of the attributes in the set above. Choosing to test only attributes in this reduced set does not affect the resulting classification of  $I_x$  or  $I_y$ , but it will affect the classification of objects that have attribute lists which differ slightly from those of  $I_x$  or  $I_y$ .

Returning to the previous example, besides the tendency rules concerning leashed dogs, there are tendency rules that can be written for *foaming* and *teeth* as well:

$$\text{teeth}(\text{large}) \rightsquigarrow \text{safeness}(\text{dangerous}),$$

$$\text{teeth}(\text{small}) \rightsquigarrow \text{safeness}(\text{safe}),$$

$$\text{foaming}(\text{yes}) \rightsquigarrow \text{safeness}(\text{dangerous}),$$

$$\text{foaming}(\text{no}) \rightsquigarrow \text{safeness}(\text{safe}).$$

These rules say that “dogs with large teeth tend to be more dangerous than dogs with small teeth,” and “dogs that are foaming at the mouth tend to be more dangerous than dogs not foaming at the mouth.”

As we shall see in section 7, only the tree with interpretation (I2) can be generated when the trees are forced to be consistent with the tendency rules above.

### 6.1 Un-instantiated Tendency Rules

As there may be times that no tendencies are known for a particular attribute, it is nice to notice that the effect of the Stepwise Generalization Algorithm, which did not consider tendency rules, is reproduced with the Tendency Rule Algorithm and *un-instantiated tendency rules* of the form

$$a(v) \rightsquigarrow \text{class}(X).$$

$X$  is a variable that can be assigned any classification value. The above rule is actually just a shorthand way of writing down the tendency rules

$$a(v) \rightsquigarrow \text{class}(x_1)$$

$$a(v) \rightsquigarrow \text{class}(x_2)$$

⋮

$$a(v) \rightsquigarrow \text{class}(x_N)$$

where the set of possible classification values is  $\{x_1, x_2, \dots, x_N\}$ . Thus, within the context of the tendency rule definition in section 6, nothing can be implied about the relation between  $P(class(x_i) | \bar{a}_1)$  and  $P(class(x_i) | \bar{a}_2)$ .

Un-instantiated tendency rules are used in the Tendency Rule Algorithm for attributes with no associated tendency rule. The Tendency Rule Algorithm will use the following definition.

**Definition**

$R^*$  is the *closure* of the set of tendency rules  $R$ . Let  $R^* = R$ , if

$$(\forall x)(a(v) \rightsquigarrow class(x) \notin R),$$

then add  $a(v) \rightsquigarrow class(X)$  to  $R^*$  for every  $a(v)$  that appears in the attribute lists of the examples.

**6.2 Tendency Rule Algorithm**

The basic structure of this algorithm is taken from the Stepwise Generalization Algorithm, but classification conflicts are resolved in a way that is consistent with the set of tendency rules  $R$ .

During the tree generation, it is possible to encounter an example that is inconsistent with the current decision tree and the tendency rules. In such an event, the inconsistent example is removed from the training set. (The "removed" examples can be thought of as bad data points.)

**Algorithm:** Decision Tree Generation by Stepwise Generalization with Tendency Rules.

*Input:* a decision tree  $T_{in}$ , a training set  $S_{in}$ , and a set of tendency rules  $R$ .

*Output:* a decision tree  $T_{out}$  consistent with  $S_{out}$  and  $R$ .

*Method:*

0)  $S_{out} := S_{in}$ .

1) Read the next example  $I = \langle id_I, \bar{a}_I, x \rangle$  from  $S_{in}$ , and let  $T_{old} := T_{in}$ . If there are no more examples in  $S_{in}$  then let  $T_{out} := T_{in}$ .

2) Call the Decision Tree Population algorithm with input tree  $T_{old}$  and set of examples  $\{I\}$ .

2.1) If the algorithm does not return **fail**, then  $T_{in}$  is the output of the algorithm.

Go to 1).

2.2) If the algorithm does return **fail**, then there is a classification conflict that must be resolved. Find the traversal of  $T_{old}$  with  $I$ .  $p_I = a^1(v^1) \wedge a^2(v^2) \wedge \dots \wedge a^N(v^N) \wedge leaf(y, id\_list)$ , where  $y \neq x$ .

2.2.1) Form the power set  $P$  of paths from the set,

$\{a^j(v^j) \in \bar{a}_I: a^j(v^j) \notin \{a^1(v^1), a^2(v^2), \dots, a^N(v^N)\},$

$a^j(v^j) \rightsquigarrow class(x) \in R^*\}$ , of yet untested attributes consistent with  $R$ .

2.2.1.1) Remove the shortest path  $p$  from  $P$  and append the leaf predicate

$leaf(x, \{id_I\})$  forming completed path  $p_c = p \wedge leaf(x, \{id_I\})$ . Call the Decision

Tree Population algorithm with the input tree  $\{p_c\}$  and the set examples whose unique identifiers are in  $id\_list$ .

2.2.1.1.1) If the algorithm does not return **fail**, then let  $T_{sub}$  be the output of the algorithm.  $T_{in}$  is  $T_{old}$  with the traversal  $p_I$  replaced by the set of connected paths  $\{p: p = a^1(v^1) \wedge a^2(v^2) \wedge \dots \wedge a^N(v^N) \wedge p_{sub}, p_{sub} \in T_{sub}\}$ . Go to 1).

2.2.1.1.2) If the algorithm does return **fail**, then return to 2.2.1.1).

**Theorem 2**

The decision tree generated by the Tendency Rule Algorithm with input  $T_{in} = \Lambda$  and training set  $S_{in}$  is a partially irredundant tree consistent with the reduced training set of examples  $S_{out}$ .

**Proof**

Same as Theorem 1. The only difference between this algorithm and the Stepwise Generalization algorithm is that the set of attribute formulas used to form the power set  $P$  is reduced since the formulas must also satisfy the tendency rules.  $\square$

## 7. A Simple Example Revisited

This discussion parallels the discussion in section 5 where we formulated a concept about the ability of a given dog to inflict bodily damage upon us. We shall now use the Tendency Rule Algorithm with the set of tendency rules

$$R = \{ \textit{leashed}(\textit{yes}) \rightsquigarrow \textit{safeness}(\textit{safe}), \\ \textit{leashed}(\textit{no}) \rightsquigarrow \textit{safeness}(\textit{dangerous}), \\ \textit{teeth}(\textit{large}) \rightsquigarrow \textit{safeness}(\textit{dangerous}), \\ \textit{teeth}(\textit{small}) \rightsquigarrow \textit{safeness}(\textit{safe}), \\ \textit{foaming}(\textit{yes}) \rightsquigarrow \textit{safeness}(\textit{dangerous}), \\ \textit{foaming}(\textit{no}) \rightsquigarrow \textit{safeness}(\textit{safe}) \}.$$

These tendency rules say that “dogs that are not leashed tend to be more dangerous than dogs that are leashed,” “dogs with large teeth tend to be more dangerous than dogs with small teeth,” and “dogs which are foaming at the mouth tend to be more dangerous than dogs not foaming at the mouth.”

The training set is the same:

$$\{ \langle \textit{spot}, \{ \textit{leashed}(\textit{no}), \textit{teeth}(\textit{large}), \textit{foaming}(\textit{no}) \}, \textit{safe} \rangle, \\ \langle \textit{fang}, \{ \textit{leashed}(\textit{no}), \textit{teeth}(\textit{small}), \textit{foaming}(\textit{yes}) \}, \textit{dangerous} \rangle, \\ \langle \textit{spike}, \{ \textit{leashed}(\textit{yes}), \textit{teeth}(\textit{small}), \textit{foaming}(\textit{yes}) \}, \textit{safe} \rangle \}.$$

The *safeness* decision tree is initially  $\Lambda$ . The first example is read and the Population algorithm is called. The traversal of the tree *fails*, so  $\Lambda$  is populated and replaced by the tree below:

(T1)  $\frac{\textit{safeness}}{\textit{leaf}(\textit{safe}, \{\textit{spot}\})}$ .

The second example is read, and the tree ascribes the classification value *safe* to *fang*. *fang*, however, is *dangerous*, and a classification conflict has occurred. We form the power set *P* of paths from the untested attribute predicates in *fang*'s attribute list that are consistent with *R*. ( $R^* = R$  in this simple example.) *fang* is *dangerous*, so only the three tendency rules with the classification values *dangerous* apply

$\textit{leashed}(\textit{no}) \sim \textit{safeness}(\textit{dangerous})$ ,

$\textit{teeth}(\textit{large}) \sim \textit{safeness}(\textit{dangerous})$ ,

$\textit{foaming}(\textit{yes}) \sim \textit{safeness}(\textit{dangerous})$ .

Of these three, *fang*'s attribute list satisfies the first and third. Consequently, we have decided that the fact that *fang* has small teeth is not a good reason for explaining why *fang* should be considered dangerous. It is this decision that keeps the tree with the unacceptable interpretation (I1) from being generated.

The resulting power set of paths is

$P = \{\textit{leashed}(\textit{no}), \textit{foaming}(\textit{yes}),$

$\textit{leashed}(\textit{no}) \wedge \textit{foaming}(\textit{yes})\}$ .

At this point we can choose one of the two length 1 paths from *P*. Taking the first, we form the completed path  $p_c = \textit{leashed}(\textit{no}) \wedge \textit{leaf}(\textit{dangerous}, \{\textit{fang}\})$ .

We now call the Population algorithm with the input tree

safeness  
 leashed(no)  
     leaf(dangerous,{fang})

and training set

{<spot, {leashed(no), teeth(large), foaming(no)}, safe>}.

The tree above is traversed with *spot* and a classification conflict occurs, so this call to the Population algorithm fails. We return to the Stepwise Generalization algorithm and select the next path from *P*. We call the algorithm again as we did above, but with the input tree

safeness  
 foaming(yes)  
     leaf(dangerous,{fang})

and training set

{<spot, {leashed(no), teeth(large), foaming(no)}, safe>}.

The traversal on the tree above with *spot* fails at the attribute *foaming*, and a new branch is added resulting in the tree

safeness  
 foaming(yes)  
     leaf(dangerous,{fang})  
 foaming(no)  
     leaf(safe,{spot}).



The subtree above replaces the leaf of (T1) where the classification conflict occurred, and the resulting tree is

(T2)                    safeness  
                           foaming(yes)  
                                   leaf(dangerous,{fang})  
                           foaming(no)  
                                   leaf(safe,{spot}).

The last example is read, and the tree incorrectly ascribes the classification value *dangerous* to *spike*. We form the power set  $P$  of paths from the untested attribute predicates in *spike*'s attribute list that are consistent with  $R$ . Only the three tendency rules with the classification value *safe* are considered, and the two untested attributes satisfy these rules. The resulting power set  $P$  of paths is

$$P = \{ \text{leashed}(\text{yes}), \text{teeth}(\text{small}), \\ \text{leashed}(\text{yes}) \wedge \text{teeth}(\text{small}) \}.$$

Choosing the length 1 path *leashed(yes)* from  $P$ , we form the completed path *leashed(yes)  $\wedge$  leaf(safe, {spike})*. We now call the Population algorithm with the input tree

safeness  
 leashed(yes)  
 leaf(safe,{spike})

and training set

$$\{ \langle \text{fang}, \{ \text{leashed}(\text{no}), \text{teeth}(\text{small}), \text{foaming}(\text{yes}) \}, \text{safe} \rangle \}.$$

The traversal on the tree above with *fang* fails at the attribute *leashed*, and a new branch is added resulting in the tree

```

safeness
  leashed(no)
    leaf(dangerous,{fang})
  leashed(yes)
    leaf(safe,{spike}).

```

This subtree now replaces the leaf of (T2) where the conflict occurred, and the resulting tree with the example lists removed from the leafs is

```

(T3)      safeness
          foaming(yes)
            leashed(no)
              leaf(dangerous)
            leashed(yes)
              leaf(safe)
          foaming(no)
            leaf(safe).

```

This tree has the acceptable interpretation:

(I2) "All dogs that are not foaming at the mouth are safe, while dogs that are foaming at the mouth are safe if they are leashed and dangerous if they are not leashed."

## 8. Secondary Classification Trees

### 8.1 The Problem

Let us consider the classification of companies according to their stock-price-gains. (This is described in detail in section 10). One of the tendency rules for this problem is

$$\textit{company\_size}(\textit{small}) \rightsquigarrow \textit{stock\_price\_gains}(\textit{top})$$

which reflects the knowledge "small companies tend to produce large stock price gains." The attribute lists of the examples, however, do not contain any attribute named *company-size*. The company-size is itself a classification concept that is dependent on the revenues, number of employees and shareholders of the company. The three attributes *revenues*, *employees*, and *shareholders* are found in the attribute lists of the examples and could be considered in the *stock-price-gains* tree. This scheme, however, has two drawbacks: there are no tendency rules relating the revenues, employees or shareholders to the stock-price-gains, and the knowledge relating the company-size to the stock-price-gains would not be used.

We should allow for the simultaneous generation of the primary *stock-price-gains* tree, which contains as one of its attributes *company-size*, and the secondary *company-size* tree, which contains the three attributes *revenues*, *employees*, and *shareholders*.

## 8.2 The Solution

Assume that we are given the tendency rule

$$class^s(v^s) \rightsquigarrow class^p(x)$$

where  $class^s$  is the name of the concept associated with the secondary tree  $T^s$ , and  $class^p$  is the name of the concept associated with the primary tree  $T^p$ . Let example  $I_x = \langle id_x, \vec{a}_x, x \rangle$  have the classification value  $x$ , and let example  $I_y = \langle id_y, \vec{a}_y, y \rangle$  represent any example, used to build the current  $T^p$  tree, with the classification value  $y$ . There are two times during the decision tree generation that the attribute  $class^s$  may be encountered: during a traversal of the primary tree, or during the resolution of a classification conflict which occurs in the primary tree. In either case we want to refine the tree, if possible, in such a way that it implies

$$P(class^p(x) | \vec{a}_x) > P(class^p(x) | \vec{a}_y).$$

If the  $class^s$  attribute is encountered in the traversal of the  $T^p$  tree with example  $I_x$ , then an attempt is made to refine the  $T^s$  tree so that it ascribes the classification value  $v^s$  to  $I_x$ . The old  $T^s$  tree is replaced by the refined  $T^s$  tree. If the  $class^s$  attribute is encountered during the resolution of a classification conflict, then an attempt is made to refine the  $T^s$  tree so that it ascribes the

classification value  $v^*$  to  $I_x$ ; however, the old  $T^*$  tree is replaced by the refined  $T^*$  tree only if the *class*<sup>\*</sup> attribute is actually used in the classification conflict resolution.

The refinement of the  $T^*$  tree is made by calling the Tendency Rule Algorithm with  $T^*$  as the primary tree, and the training set consisting of single example  $I_x^* = \langle id_x, \bar{a}_x, v^* \rangle$ .  $I_x^*$  has the classification value  $v^*$ , which was substituted for the classification value  $x$  of example  $I_x$ . If the algorithm cannot refine the tree in a way that is consistent with the examples that were previously used to build the tree, then *class*<sup>\*</sup> simply returns the classification value that the current  $T^*$  tree ascribes to example  $I_x$ .

## 9. Ordering the Attributes by Priority

Because it may be known *a priori* that particular attributes are very important and should be tested first, we should allow the attributes to be ordered so that all higher priority attributes appear closer to the root than any lower priority attributes. Given a priority ordering function,  $priority(a^j)$ , on the attributes, this condition can be achieved by simply requiring all the paths generated in the Tendency Rule Algorithm to satisfy the conditions:

given a path

$$p = a^1(v^1) \wedge a^2(v^2) \wedge \dots \wedge a^{i-1}(v^{i-1}) \wedge a^i(v^i) \wedge \dots \wedge a^N(v^N) \wedge leaf(x)$$

and

$$priority(a^i) = M$$

then

$$priority(a^1) = 1$$

$$priority(a^{i-1}) \leq priority(a^i)$$

$$(\forall a^j)(priority(a^j) < M, a^j \in \{a^1, a^2, \dots, a^{i-1}\}).$$

The decision tree generated using priority ordered attributes may not be a partially irredundant tree because the algorithm is forced to place attributes, which may not be needed to resolve a conflict, into the paths that make up the decision tree.

## 10. Case Study — Hottest Stocks

The ideas presented in this paper were formulated after reading the *Hottest Stocks* article in the March 1985 issue of **California Business**<sup>[5]</sup>. The article contains a table of the 75 California companies which had the greatest percentage stock-price-increase. The table contains headings such as the number of employees, the number of shareholders, the P-E ratios for 1984 and 1985, and the trading turnover rate. Also included in the article is a description of properties that *tend* to make a top stock. Here is a portion of the article's text:

Analysis of the statistics for this year's 75 ranked stocks indicates that price gains can be attributed to the following circumstances:

- A higher P-E ratio that frequently reflects a strong per share earnings gain in 1984 extending a several-year uptrend.
- A strong 1984 per share earnings gain that usually does not extend a several-year uptrend and is accompanied by a P-E ratio either unchanged or reduced to a more conservative rate....

Additionally, review shows that investment performance may benefit from recognizing the following:

- Investing in stocks of smaller companies yields more frequent stock price gains than investing in larger-company stocks.
- Investing in lower-priced stocks generates more frequent strong price gains than investing in stocks trading at or above the popular \$20 level.

A refined Tendency Rule Algorithm, which allowed for the generation of secondary decision trees and priority ordered attributes, was implemented in Prolog<sup>[4]</sup>. In order to illustrate the workings of the program the first 25 examples

were classified as *top* and the last 25 were classified as *bottom*. It should be noted that this is a rather unusual classification. We will be attempting to formulate a concept that distinguishes "very very good" companies from "very good" companies.

### 10.1 Examples

A set of eight attributes are found in the attribute lists of the companies: *employees*, *holders*, *revenues*, *holdings*, *stock\_price*, *e-p-share*, *p-e-ratio*, and *turnover*. Each attribute had a set of either two or three possible attribute values. Below is the rule for determining whether the attribute value of *employees* for a particular company should be *large* or *small*.

```
employees  
if (number_employees < 500) then  
    return small  
else  
    return large
```

This is a sample example

```
<starr_surgical, {employees(small), holders(small), holdings(small),  
revenues(small), stock_price(low), e-p-share(down), p-e-ratio(none),  
turnover(large)}, top > .
```



## 10.2 Tendency Rules

An initial concern was that the tendencies described in the article were actually found by examining the statistics of the particular 75 stocks presented. Using tendency rules that described such tendencies would actually defeat the benefit of having tendency rules. Tendency rules are introduced in order to supply some knowledge about the concept that is not contained in the examples. Chuck Erickson, an author of the article and senior financial analyst for SRI International, alleviated this concern<sup>[8]</sup>. Behind each of the tendencies was a, usually complex, chain of reasoning that indicated why the particular tendency should be true. The tendency "Investing in stocks with significant institutional ownership often provides larger price gains than investing in stocks lacking institutional positions" is, at least superficially, contradicted by the statistics of the 75 stocks listed. The reason for this tendency, however, is that significant institutional ownership often reflects some knowledge possessed by the institutional investors that is not possessed by the private investor.

The tendency rules for the primary, *stock\_price\_gains* tree are shown below:

$$\begin{aligned} R_{spg} = \{ & \text{holdings}(\text{large}) \rightsquigarrow \text{stock\_price\_gains}(\text{top}), \\ & \text{holdings}(\text{small}) \rightsquigarrow \text{stock\_price\_gains}(\text{bottom}), \\ & \text{stock\_price}(\text{low}) \rightsquigarrow \text{stock\_price\_gains}(\text{top}), \\ & \text{stock\_price}(\text{high}) \rightsquigarrow \text{stock\_price\_gains}(\text{bottom}), \\ & \text{e\_p\_share}(\text{up}) \rightsquigarrow \text{stock\_price\_gains}(\text{top}), \\ & \text{e\_p\_share}(\text{down}) \rightsquigarrow \text{stock\_price\_gains}(\text{bottom}), \\ & \text{turnover}(\text{high}) \rightsquigarrow \text{stock\_price\_gains}(\text{top}), \\ & \text{turnover}(\text{low}) \rightsquigarrow \text{stock\_price\_gains}(\text{bottom}) \}, \end{aligned}$$

and

$$\begin{aligned} R_{spg\text{-size}} = \{ & \text{company\_size}(\text{small}) \rightsquigarrow \text{stock\_price\_gains}(\text{top}), \\ & \text{company\_size}(\text{large}) \rightsquigarrow \text{stock\_price\_gains}(\text{bottom}) \}. \end{aligned}$$

The set of tendency rules  $R_{spg}$  relates the stock-price-gains to attributes contained in the attribute lists of the companies. No tendency rule could be written for the *p\_e\_ratio* attribute, so the closure of the set of tendency rules used in the algorithm will contain three un-instantiated tendency rules:

$$\begin{aligned} & \text{p\_e\_ratio}(\text{up}) \rightsquigarrow \text{stock\_price\_gains}(X), \\ & \text{p\_e\_ratio}(\text{down}) \rightsquigarrow \text{stock\_price\_gains}(X), \\ & \text{p\_e\_ratio}(\text{none}) \rightsquigarrow \text{stock\_price\_gains}(X), \end{aligned}$$

where a *p-e-ratio* of *none* means that the company recorded net losses in both 1984 and 1985 so that no P-E Ratio trend could be computed. The set of tendency rules  $R_{spg-size}$  contains the rules that connect the primary *stock-price-gains* tree to the secondary *company-size* tree.

The attributes were ordered with the following priority function:

$priority(p-e-ratio) = 1$ ,  $priority(e-p-share) = 2$ ,  $priority(company-size) = 2$ ,  
 $priority(holdings) = 2$ ,  $priority(turnover) = 2$ , and  $priority(stock-price) = 2$ .

Below is the set of tendency rules for the secondary decision tree, *company-size*:

$$R_{size} = \{ employees(small) \rightsquigarrow company\_size(small),$$

$$employees(large) \rightsquigarrow company\_size(large),$$

$$holders(small) \rightsquigarrow company\_size(small),$$

$$holders(large) \rightsquigarrow company\_size(large),$$

$$revenues(small) \rightsquigarrow company\_size(small),$$

$$revenues(large) \rightsquigarrow company\_size(large) \}.$$

These rules simply reflect the knowledge that "small things are made up of small things" and "large things are made up of large things." The priority function for these attributes is

$priority(employees) = 1$ ,  $priority(holders) = 1$ , and  $priority(revenues) = 1$ .

Decision trees were generated with and without the tendency rules above with the hope of showing that using the tendency rules produced a tree that was a better classifier. The set of tendency rules used for the first tree was

$$R = R_{spg} \cup R_{spg-size} \cup R_{size}.$$

The set of tendency rules used for the second tree was

$$R = R_{spg-size}.$$

It was necessary to include these rules that connected the primary and secondary tree. If these rules were not included, then no secondary tree would be generated, and the ability to differentiate companies with the attributes *employees*, *revenues*, and *holders* would be lost.

### 10.3 Results

For the two cases a training set of ten examples was used; five of the examples were *top* companies and the other five were *bottom* companies. These decision trees are shown in figures 1 and 2. These trees were then used to predict the classification values of the remaining forty examples. The first tree, which used the full set of tendency rules, correctly classified 28 out of the 40 or 70% of the examples, while the second tree correctly classified 21 of the 40 or 53% of

the examples. This test was conducted on twenty random training sets each containing ten examples. As one would expect, the trees generated without the tendency rules were very dependent on the particular training set. The resulting trees correctly classified between 40% and 70% of the remaining examples. The trees generated with the tendency rules tended to be less dependent on the training set and correctly classified between 55% and 70% of the remaining examples. Only one training set produced a tree that was a better classifier without the tendency rules than with them.

The results of this test were better than what was originally expected considering the similarity of the two types of companies.

The decision trees generated with training sets containing all fifty examples are shown in figures 3 and 4. The *company-size* trees are the same as those shown in figures 1 and 2 respectively.

Figure 1

stock\_price\_gains

*p\_e\_ratio(up)*  
    *turnover(large)*  
        *leaf(top)*  
    *turnover(small)*  
        *leaf(bottom)*  
*p\_e\_ratio(none)*  
    *leaf(top)*  
*p\_e\_ratio(down)*  
    *company\_size(small)*  
        *e\_p\_share(down)*  
            *leaf(bottom)*  
        *e\_p\_share(up)*  
            *leaf(top)*  
    *company\_size(large)*  
        *leaf(bottom)*

company\_size

*employees(large)*  
    *leaf(large)*  
*employees(small)*  
    *leaf(small)*

Figure 2

stock\_price\_gains  
p\_e\_ratio(up)  
    company\_size(small)  
        leaf(top)  
    company\_size(large)  
        leaf(bottom)  
p\_e\_ratio(none)  
    holdings(large)  
        leaf(bottom)  
    holdings(small)  
        leaf(top)  
p\_e\_ratio(down)  
    company\_size(small)  
        leaf(top)  
    company\_size(large)  
        leaf(bottom)

company\_size  
employees(small)  
    leaf(large)  
employees(small)  
    holders(small)  
        leaf(large)  
    holders(large)  
        leaf(small)

Figure 3

stock\_price\_gains

*p-e\_ratio(up)*

*turnover(large)*

*leaf(top)*

*turnover(small)*

*leaf(bottom)*

*p-e\_ratio(none)*

*leaf(top)*

*p-e\_ratio(down)*

*company\_size(small)*

*e-p\_share(down)*

*stock\_price(low)*

*leaf(top)*

*stock\_price(high)*

*leaf(bottom)*

*e-p\_share(up)*

*leaf(top)*

*company\_size(large)*

*leaf(bottom)*



Figure 4

```
stock_price_gains  
p_e_ratio(up)  
  company_size(small)  
    leaf(top)  
  company_size(large)  
    stock_price(high)  
      holdings(small)  
        e_p_share(down)  
          leaf(bottom)  
        e_p_share(up)  
          leaf(top)  
      holdings(large)  
        turnover(large)  
          leaf(top)  
        turnover(small)  
          leaf(bottom)  
    stock_price(low)  
      turnover(large)  
        leaf(top)  
      turnover(small)  
        leaf(bottom)  
p_e_ratio(none)  
  holdings(large)  
    leaf(bottom)  
  holdings(small)  
    leaf(top)  
p_e_ratio(down)  
  company_size(small)  
    stock_price(low)  
      leaf(bottom)  
    stock_price(high)  
      e_p_share(down)  
        leaf(bottom)  
      e_p_share(up)  
        holdings(large)  
          leaf(bottom)  
        holdings(small)  
          leaf(top)  
  company_size(large)  
    stock_price(low)  
      turnover(small)  
        e_p_share(down)  
          leaf(top)  
        e_p_share(up)  
          turnover(large)  
            leaf(top)  
    stock_price(high)  
      leaf(bottom)
```

## 11. Conclusion

“... in learning in general the key is learning to inhibit responses to irrelevant variables.”

This quote is from Guilford's The Nature of Human Intelligence<sup>[6]</sup> which summarized some of Harlow's<sup>[7]</sup> work on concept-learning problems. (Guilford points out that this is an oversimplification, yet it is undoubtedly an aspect of learning.) In this discussion it has been the role of the tendency rules to “inhibit responses to irrelevant variables” by reducing the set of attributes which can be used to resolve a classification conflict. This is particularly important when the size of the training set is small compared to the size of the set of objects that the automatically generated decision tree is expected to classify.

Temptations to think of a tendency rule as some sort of “inexact reasoning” rule with a hidden confidence factor should be resisted. A tendency rule does not imply anything about the classification value of a particular object. The tendency rules carry semantic information about the concept to be classified, and should really be viewed as operating on the space of acceptable decision trees.

It is the author's belief that this work is a first step towards the use of a multi-leveled view of knowledge for non-deductive inferencing. On the bottom level of this knowledge scheme are facts, which are in some sense atomic units. Each higher level is associated with a type of knowledge that is more abstract

than the previous level. In this work the lowest, factual level corresponds to objects. The second level contains the decision tree which carries the abstract notion of a concept and operates on the objects. On the third level is the tendency rule which conveys semantic information and operates on the space of decision trees. We have shown how to take knowledge from two of these levels, the examples in the training set and human knowledge in the form of tendency rules, to construct a new piece of knowledge, the decision tree. The resulting decision tree is hopefully better for the introduction of this second type of knowledge, the tendency rule.

## List of References

- [1] Hunt, Earl B., Marin, Janet, and Stone, Philip J., *Experiments in Induction*, Academic Press, New York, N.Y. (1966).
- [2] Quinlan, J. R., "Discovering Rules by Induction from Large Collections of Examples," in *Expert Systems in the Micro-electronic Age*, ed. Donald Michie, Edinburgh University Press, Edinburgh (1979).
- [3] Michie, Donald, Muggleton, Stephen, and Riese, Charles, "Rulemaster: A Second-generation Knowledge-engineering Facility," in *The First Conference on Artificial Intelligence Applications*, IEEE Computer Society (1984).
- [4] Clocksin, W.F., and Mellish, C.S., *Programming in Prolog*, Springer-Verlag, Berlin Heidelberg (1981).
- [5] Erickson, Charles E., and Harris, Michael T., "Hottest Stocks", in *California Business*, March 1985.
- [6] Guilford, J.P., *The Nature of Human Intelligence*, McGraw-Hill, N.Y. (1967).
- [7] Harlow, H.F., "The Evolution of Learning," in *Behavior and Evolution*, eds. A. Roe and G.G. Simpson, Yale, New Haven, Conn. (1958).
- [8] Erickson, Charles E., Personal Communication, June 1985.