Machine Learning, 24, 5–40 (1996)

BEXA: A Covering Algorithm for Learning Propositional Concept Descriptions

HENDRIK THERON AND IAN CLOETE ian@cs.sun.ac.za
Department of Computer Science, University of Stellenbosch, Stellenbosch 7600, South Africa

Editor: Paul Utgoff

Abstract. BEXA is a new covering algorithm for inducing propositional concept descriptions. Existing covering algorithms such as AQ15 and CN2 place rigid constraints on the search process to reduce the learning time. These restrictions may allow useless specializations while at the same time ignoring potentially useful specializations. In contrast BEXA employs three dynamic search constraints that enable it to find simple and accurate concept descriptions efficiently. This paper describes the BEXA algorithm and its relationship to the covering algorithms AQ15, CN2, GREEDY3, PRISM, and an algorithm proposed by Gray. The specialization models of these algorithms are described in the uniform framework of specialization by exclusion of values. BEXA is compared empirically to state-of-the-art concept learners CN2 and C4.5. It produces rules of comparable accuracy, but with greater simplicity.

Keywords: Learning from examples, covering algorithms, irredundant set covers, internal disjunction

1. Introduction

The goal of a concept learning algorithm is to induce a concept description when provided with a set of positive and negative instances of the concept. Covering algorithms are a class of learning algorithms that construct concept descriptions by repeatedly generating conjunctive expressions until all the positive instances of a concept are covered (i.e. matched). This paper describes BEXA (for Basic EXclusion Algorithm), a new covering algorithm for learning concept descriptions. BEXA represents its concept descriptions as expressions in VL_1, Michalski's multiple-valued extension to propositional logic. Table 1 contains a sample learning problem and examples of VL_1 concept descriptions. Empirical evidence is given for ten test databases that BEXA generates concept descriptions of comparable or better accuracy, but with greater simplicity, than well-known state-of-the-art concept learners. BEXA's richer description language, specialization method and search restrictions prefer more general descriptions, while its stop-growth test prevents overfitting. Its bias towards generality can be adjusted by suitable settings of its parameters.

The main problem faced by a covering algorithm is to construct accurate and simple conjunctions. BEXA is one of a family of propositional covering algorithms that construct conjunctions using a general-to-specific search. In this approach, the algorithms start with a general concept description and specialize it in steps until some stop-growth criterion is met. The **key problem** is to determine which specializations must be constructed at each specialization step. On the one hand, too many specializations should not be considered since this may require too much computation time. On the other hand, too few specializations should not be considered since this will reduce the chance of finding a good conjunction. Covering algorithms such as AQ15 and CN2 approach this problem by placing rigid constraints on the number of specializations that are constructed. For example, AQ15 constructs at most a (the number of attributes) specializations per step, while CN2 constructs only pure conjunctions (as opposed to internally disjunctive ones). The main goal of these restrictions is to reduce the learning time by forcefully reducing the number of specializations that can be constructed. The main problem with these types of restrictions is that they can ignore potentially good specializations of a conjunction, while at the same time allowing potentially useless specializations. In contrast, BEXA allows a more general description language and employs dynamic restrictions that exploit general properties of the search problem to avoid useless specializations without excluding potentially useful specializations. These restrictions make it possible for BEXA to find accurate and simple concept descriptions efficiently.

The main ideas presented for covering algorithms are thus: (1) A uniform framework for comparing the specialization models of various covering algorithms, (2) efficiency improvements and search restrictions during the specialization process, and, (3) criteria for terminating specialization. This paper proposes that the specialization of a conjunction can be viewed as a process of excluding values rather than appending atoms, i.e. instead of appending an atom to a conjunction (initially the constant **true**) to make it more specific, a value is removed from the most general conjunction (the conjunction that initially covers all training instances). This view has two advantages. Firstly, it leads to a simple and uniform framework for comparison of the similarities and differences between BEXA and five related covering algorithms, namely AQ15 (Michalski *et al.*, 1986), CN2 (Clark & Boswell, 1991), PRISM (Cendrowska, 1987), GREEDY3 (Pagallo & Haussler, 1990) and an algorithm proposed by Gray (1990). The latter algorithm will be called GALG. We will show that the appending atoms approach followed by algorithms like AQ15 and CN2 can also be viewed as one of implicitly excluding values. The algorithms then differ mainly with respect to the number of values that they exclude when constructing a specialization, and the number of different specializations that are constructed. Secondly, it leads to a precise characterization of the set C_M of most general and consistent conjunctions. All the covering algorithms discussed in this paper construct conjunctions by following a general-to-specific search, thus intentionally being biased towards generality. BEXA allows the option of making this bias explicit by restricting the search for a conjunction so that an element in C_M is generated. The exclusion view of specialization makes it possible to show that conjunctions in C_M correspond to irredundant set covers of the set N of negative instances of a concept. This characaterization of C_M led to an important search restriction (uncover-new-negatives) which both improves the accuracy of conjunctions and the efficiency of the search process.

Section 2 defines the VL$_1$ language and other basic concepts used in this paper. Section 3 describes the role that C_M plays in the search for good concept descriptions and introduces the exclusion view of specialization. BEXA is described and illustrated with an example in Section 4. Section 5 describes the similarities and differences between BEXA and the five covering algorithms mentioned above. This discussion focuses on the differences regarding the algorithms' control structures, evaluation functions and pruning schemes. Section 5.2.1 presents a complexity analysis of the specialization models of the above-mentioned algorithms in terms of the number of specialization steps required to reach a consistent conjunction and the different specializations considered. It exploits the exclusion view of specialization to describe how these algorithms specialize a conjunction and how they restrict the number of specializations. Section 6 presents empirical evidence that BEXA's search restrictions actually reduce its learning time without degrading description accuracy or complexity. This section also presents a detailed empirical comparison between BEXA and CN2, a closely related covering algorithm. BEXA is also empirically compared to C4.5 to evaluate its performance. C4.5 serves as a good performance yard-stick, even though C4.5's specialization process (a decision tree generator) differs substantially from the set covering approach. We close with a summary.

2. Basic concepts and definitions

The propositional concept learning problem is defined as follows. Let $A_1,...,A_n$ denote attributes with domains $D_1,...,D_n$. *Nominal* attributes take a finite set of unordered values, e.g. `outlook` can take the values {`sunny`, `overcast`, `rain`}. A boolean attribute has the domain {`true`, `false`}. *Linear* (integer or real) attributes have linearly ordered domains. For example, the integer attribute `age` can take any value between 0 and 120.

The *instance space* I defined by $A_1, ..., A_n$ is the cross-product $D_1 \times ... \times D_n$. An *instance* is denoted by <**x**,c> where $\mathbf{x} \in I$ and $c \in$ *Concepts*. The *training set* T presented to a concept learner is a subset of $I \times$*Concepts*. Instances in the set $P \subset T$ that belong to a specific concept are called its *positive instances* (examples) and instances in the set $N = T - P$ are called its *negative instances*. Table 1 contains a sample learning problem. `TS` will be used to denote the training set in this table throughout the paper.

VL$_1$ (Michalski, 1975) is a multiple-valued extension to propositional logic. We follow Haussler (1988) and describe the relevant subset of VL$_1$ using standard logic terminology. Attributes are related to values via *atoms* (*selectors* in Michalski's terminology). *Elemen-*

Table 1. A sample learning problem and examples of VL$_1$ concept descriptions

Concept to learn

It will stop raining tomorrow

Attributes

Name	**Type**	**Domain**
outlook	nominal	{sunny,overcast,rain}
autumn	nominal	{yes,no}
temp	linear	{15..35}

The training set `TS`

#	**outlook**	**autumn**	**temp**	**class**
1	sunny	yes	17	−
2	overcast	no	18	
3	rain	yes	16	−
4	sunny	yes	22	−
5	sunny	no	29	−
6	overcast	yes	30	−
7	overcast	no	35	−
8	rain	yes	23	−
9	rain	no	27	−
10	sunny	yes	28	+
11	overcast	no	23	+
12	sunny	no	27	+
13	rain	no	23	+

Examples of VL$_1$ concept descriptions

1. [outlook $\in$ {overcast,sunny}][22 $<$ temp $\leq$ 28] $\vee$ [autumn = no][temp = 23] $\Rightarrow$ +
2. [outlook = sunny][22 $<$ temp $\leq$ 28] $\vee$ [autumn = no][temp = 23] $\Rightarrow$ +

tary atoms take the form $[A_i = a_i]$ for nominal attributes, e.g., [sex = male]. For linear attributes elementary atoms take the form $[A_i \# a_i]$ with # $\in \{=, <, \leq, >, \geq\}$ or $[a_i \# A_i \# b_i]$ with # $\in \{<, \leq\}$, e.g., [age $<$ 10] and [20 $<$ weight $\leq$ 100]. *Compound* nominal atoms take the form $[A_i = a_i \vee \ldots \vee a_l]$. Such atoms will be denoted as $[A_i \in S_i]$, where $S_i = \{a_i, \ldots, a_l\}$, e.g. [outlook $\in$ {sunny, rain}]. This convention is introduced because it will help to explain BEXA's specialization model. Compound linear atoms consist of any disjunction of elementary linear atoms, e.g. [(temp $=$ 10) $\vee$ (20 $\leq$ temp $<$ 30) $\vee$ (temp $>$ 50)].

VL$_1$ *expressions* are defined as follows. (1) An atom is an expression. (2) A *conjunctive* expression is the conjunction of one or more atoms. (Michalski calls a conjunction a *complex*). Adjacent atoms have an implicit $\wedge$ (and) between them. (3) A *disjunctive* expression is the disjunction of one or more conjunctions. An expression containing only elementary atoms is said to be *pure* (Haussler, 1988), otherwise it is *internally disjunctive*. (4) An expression that implies a concept is called a *rule* or a *concept description*. For example, Rule 1 in Table 1 is internally disjunctive and Rule 2 is pure. A set of disjunctive rules can always be written as an equivalent set of *production rules* and vice versa. For example, Rule 1 in Table 1 is equivalent to the two production rules

[outlook $\in$ {overcast, sunny}][22 $<$ temp $\leq$ 28] $\Rightarrow$ +

[autumn = no][temp = 23] $\Rightarrow$ +.

Instances and expressions are related as follows. Let h denote an expression and let $B \subseteq I$ denote a set of instances. Then the *extension* of h in B, denoted by $X_B(h)$, is defined as all those instances in B that match h. We say that h *covers* a subset of instances in B. For example,

X_{TS}([outlook $\in$ {overcast, sunny}][22 $<$ temp $\leq$ 28]) $= \{10, 11, 12\}$

where 10, 11 and 12 denote the numbers of the instances in Table 1. The extension of a single attribute value $a_i \in D_i$ in a set $B \subseteq I$ is defined as the extension of the expression $[A_i = a_i]$ in B and is denoted by $X_B(a_i)$. For example,

X_{TS}(sunny) $= X_{\mathrm{TS}}$([outlook = sunny]) $= \{1, 4, 5, 10, 12\}$.

The extension of a value or expression in the set P and the set N is called its *positive extension* and its *negative extension* respectively. A *set cover* of a set B is a set $C = \{C_1, \ldots, C_n\}$ of subsets of B such that their union equals B. C is an *irredundant* set cover of B if the deletion of any set C_i from C will cause the union of the remaining sets in C to be a proper subset of B. For example, the first conjunction of rule 1 in Table 1 has as its positive extension the set $\{10, 11, 12\}$ while its second conjunction covers $\{11, 13\}$. Rule 1 thus corresponds to the set cover $\{\{10, 11, 12\}, \{11, 13\}\}$ of $P = \{10, 11, 12, 13\}$ in TS. This is also an irredundant set cover of P. An expression E is said to be *consistent* if it covers none of a concept's negative instances, and it is *complete* if it covers all the concept's positive instances. For example, both conjunctions in rule (1) are consistent because they cover no negative instances in TS, and the rule itself is complete because it covers all the positive instances in TS.

3. Searching for good concept descriptions

This section presents a uniform framework for the analysis of the various algorithms' specialization models in Section 5.2.1. This analysis provides the basis for BEXA's most important search restriction and leads to the specialization method of excluding values rather than appending atoms.

Assume the goal of a concept learning algorithm is to find accurate and simple concept descriptions, with accuracy being the primary concern, followed by simplicity. All the covering algorithms considered in this paper can generate disjunctive concept descriptions or an equivalent set of production rules. The quality of these descriptions is determined by the quality of their conjunctions. The main problem is therefore to find accurate and simple conjunctions. All the covering algorithms employ an evaluation function that estimates the accuracy of a conjunction. All of these evaluation functions (see Table 4 in Section 5) prefer conjunctions that cover many positive instances and few negative instances, thus being biased to find general rather than more specific conjunctions. These evaluation functions thus have high values for consistent conjunctions, i.e. conjunctions that cover no negative instances, and higher values for most general consistent conjunctions. Assume for the moment, therefore, that an appropriate bias (Schaffer, 1993) is to find most general and consistent conjunctions. (We will show later that BEXA's parameters allow the selection of a bias more appropriate to a domain of application.)

Table 2. A small artificial learning problem

	Training set							
Attributes	**#**	**A**	**B**	**Class**	**#**	**A**	**B**	**Class**
A ∈ {a,b,c}	1	a	x	+	4	b	y	+
B ∈ {x,y}	2	a	y	−	5	c	x	−
	3	b	x	+	6	c	y	+

Let C denote the set of all VL$_1$ conjunctions that exist for a given learning problem, and let c and d denote two of these conjunctions. Then c is defined to be 'more specific than or equal to' d, denoted by $c \leq d$, if and only if $X_I(c) \subseteq X_I(d)$ (Mitchell, 1982). Recall that I denotes the instance space of all possible instances for a given learning problem. The conjunctions c and d are considered equal, denoted by $c = d$, when $X_I(c) = X_I(d)$. Conjunction c is strictly more specific than d, denoted by $c < d$, if $c \leq d$ and $c \neq d$. The set C is partially ordered under the $\leq$ relation. For the problems that we consider, C is bounded from above by the most general conjunction and from below by the NULL conjunction defined as that conjunction for which $X_I(NULL) = \emptyset$. The most general conjunction that exists for a given learning problem will henceforth be denoted by **mgc**. The expressions in C thus form a lattice under the partial ordering $\leq$. Table 2 contains a small artificial learning problem, and Figure 1 contains the lattice of the conjunctions that can be constructed for this learning problem.

Elements of C_C are underlined, while elements of C_M also have a line above them.
Attribute names (A and B) have been dropped to save space.

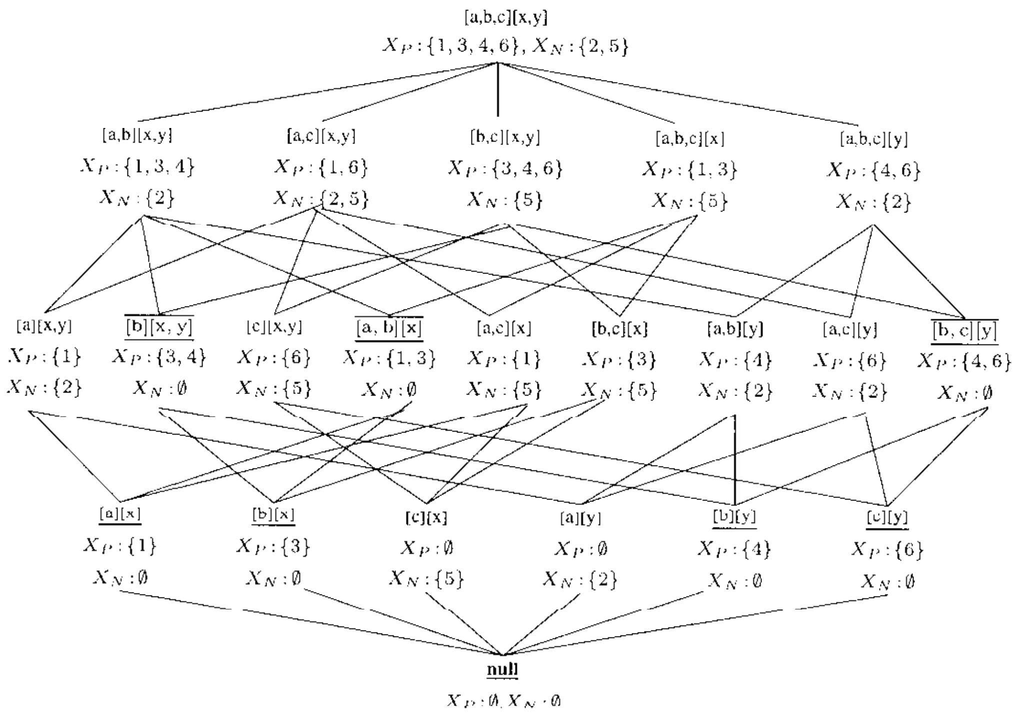
[a,b,c][x,y]
$X_P:\{1,3,4,6\}, X_N:\{2,5\}$
[a,b][x,y] $X_P:\{1,3,4\}$ $X_N:\{2\}$
[a,c][x,y] $X_P:\{1,6\}$ $X_N:\{2,5\}$
[b,c][x,y] $X_P:\{3,4,6\}$ $X_N:\{5\}$
[a,b,c][x] $X_P:\{1,3\}$ $X_N:\{5\}$
[a,b,c][y] $X_P:\{4,6\}$ $X_N:\{2\}$
[a][x,y] $X_P:\{1\}$ $X_N:\{2\}$
[b][x,y] $X_P:\{3,4\}$ $X_N:\emptyset$
[c][x,y] $X_P:\{6\}$ $X_N:\{5\}$
[a,b][x] $X_P:\{1,3\}$ $X_N:\emptyset$
[a,c][x] $X_P:\{1\}$ $X_N:\{5\}$
[b,c][x] $X_P:\{3\}$ $X_N:\{5\}$
[a,b][y] $X_P:\{4\}$ $X_N:\{2\}$
[a,c][y] $X_P:\{6\}$ $X_N:\{2\}$
[b,c][y] $X_P:\{4,6\}$ $X_N:\emptyset$
[a][x] $X_P:\{1\}$ $X_N:\emptyset$
[b][x] $X_P:\{3\}$ $X_N:\emptyset$
[c][x] $X_P:\emptyset$ $X_N:\{5\}$
[a][y] $X_P:\emptyset$ $X_N:\{2\}$
[b][y] $X_P:\{4\}$ $X_N:\emptyset$
[c][y] $X_P:\{6\}$ $X_N:\emptyset$
null
$X_P:\emptyset, X_N:\emptyset$

Figure 1. The sets C_C and C_M in the lattice of conjunctions for the problem in Table 2

Let C_C denote the subset of consistent conjunctions in C. Conjunctions in C_C are underlined in Figure 1. Let C_M denote the set of most general consistent conjunctions. Formally

$C_M = \{m \in C_C \mid \text{there is no element } c \in C_C \text{ such that } m < c\}$.

The elements of C_M in Figure 1 also have a line above them. Under the assumption that consistent conjunctions are sought, the goal is to find conjunctions in C_C that cover many positive instances and contain few atoms. Elements of C_M thus have the highest values for these evaluation functions, since a conjunction in C_C can only be improved by generalizing it until it becomes an element of C_M, while any further generalization will cause the conjunction to become inconsistent.

The secondary goal is to find the simplest conjunctions among those with a high value for the evaluation function. The most common measure for description complexity is the total number of atoms in a concept description (Michalski *et al.*, 1986; Clark & Niblett, 1989; Clark & Boswell, 1991; Lavrac *et al.*, 1986; Pagallo & Haussler, 1990). To make the partial ordering among conjunctions clear, all the conjunctions in Figure 1 were written such that they contain all the attributes. However, the usual convention is to discard all those attributes that take all their possible values. The number of atoms in a conjunction can thus be reduced by generalizing it to such an extent that one or more of its attributes take all their values and hence can be discarded. No conjunction in C_M can thus be simplified any further without causing it to become inconsistent. Therefore conjunctions in C_M contain a least number of atoms.

We now present a characterization of conjunctions in C_M based on the exclusion view of specialization. Construct conjunctions as follows. Specialize the **mgc** by excluding one attribute value at a time. Each excluded value uncovers a subset of positive and a subset of negative instances. The conjunction becomes consistent when enough values have been excluded to uncover all the negative instances, i.e. the negative extensions of the excluded values form a set cover of N. However, the problem is that merely excluding values until all the negative instances are uncovered does not guarantee that the conjunction is an element of C_M. The following key property distinguishes between conjunctions in C_M and those in $C_U - C_M$. Let c denote a conjunction and let $R = \{r_1, ..., r_n\}$ denote the subset of attribute values excluded from the **mgc** to obtain c. Then

$c \in C_M$ if and only if the set $\{X_N(r_1), ..., X_N(r_n)\}$ is an irredundant set cover of N.

For example, the **mgc** for the problem in Table 2 is

g : [A ∈ {a,b,c}] [B ∈ {x,y}].

The conjunction [A ⊂ {b,c}] [B = y] can be obtained by excluding the values

value	X_N(**value**)	X_P(**value**)
[A = a]	{2}	{1}
[B = x]	{5}	{1,3}

from g. The specialization process proceeds as follows:

conjunction	X_N(**conjunction**)	X_P(**conjunction**)	
[A ∈ {a,b,c}] [B ∈ {x,y}]	{2,5}	{1,3,4,6}	the mgc
[A ∈ {b,c}] [B ∈ {x,y}]	{5}	{3,4,6}	exclude [A = a]
m : [A ∈ {b,c}] [B = y]	{}	{4,6}	exclude [B = x]

Now $m \in C_M$ because the negative extensions of the excluded values [A = a] and [B = x] form the irredundant set cover {{2},{5}} of the set $N = \{2, 5\}$.

Specialization by excluding values thus steps through the lattice level by level. Enforcing irredundancy implies that each excluded value uncovers at least one new negative example. In addition fewer specialization steps may be required when at least one new negative example is uncovered at each step, while terminating when an element of C_M is found. This characterization of C_M forms the basis for two of BEXA's search restrictions. In contrast, Section 5.2.1 will show that AQ15, CN2, PRISM, GREEDY3 and GALG all specialize a conjunction by appending atoms to it. Appending an atom may cause the specialization process to move erratically through the levels of the lattice rendering the characterization of C_M no longer valid. Subsets of negative instances uncovered by appended atoms may form an irredundant set cover of N even though the corresponding conjunction is not in C_M. For example, CN2 may construct the conjunction [A = c] [B = y] by appending the atoms [A = c] and [B = y] to **true**. These two atoms respectively uncover the subsets {2} and {5} of negative instances. These two sets form an irredundant set cover of N even though [A = c] [B = y] is not in C_M.

At the beginning of this section we assumed that the goal is to find consistent conjunctions. However, this condition can be relaxed and a different bias implemented by pre-pruning tests, such as BEXA's stop-growth test, which may terminate the specialization process before a conjunction becomes consistent. Pruning is discussed in Sections 4.2, 5.2.6 and 5.3.

4. BEXA

4.1. BEXA's top-level loops

Table 3 contains the BEXA algorithm. Procedure COVER-P generates conjunctions until all the positive instances of a concept are covered, or until a NULL conjunction is returned due to pruning. Procedure Find-Best-Conjunction finds the subsequent best conjunction to add to the concept description. It is described in general terms because it will also serve as a framework for describing some of the differences among the covering algorithms in Section 5. A beam search (Steps (2) and (7) in Table 3) is employed to find the best conjunction. Starting with the **mgc** (Step (1) in Table 3), the current set of conjunctions is specialized in steps (Step (3) in Table 3). The Laplace accuracy estimate (defined in Table 4) is used to select the best specialization after each step (Step (5) in Table 3). This estimate was borrowed from CN2 (Clark & Boswell, 1991).

BEXA can employ CN2's significance test (Step (4) in Table 3). This test compares the distribution of instances covered by a conjunction to that of a conjunction covering instances in the same proportions as they occur in the complete training set. Only conjunctions that are significant according to the log-likelihood ratio test (described in Clark & Niblett (1989)) are compared to the current best conjunction. The significance test thus weeds out conjunctions that do not capture significant patterns in the complete training set.

BEXA employs two stop-growth tests (Step (6) in Table 3). The first is the usual test that discards specializations that cover no negative instances. The reason is that any further specialization can only decrease the number of positive instances covered by these conjunctions. The second stop-growth test is related to the stop-growth tests of TDIDT[1] methods, e.g. the chi-square test employed in early versions of ID3 (Quinlan, 1986). It compares the distribution of instances covered by a specialization to that of its direct predecessor using the log-likelihood ratio test. If this difference is insignificant, the specialization is discarded from further consideration. The stop-growth test thus insists that each specialization step changes a conjunction significantly, while the significance test ensures that a complete conjunction captures a significant pattern in the training set. Another difference is that the significance test simply ignores insignificant conjunctions, while the stop-growth test discards insignificant specializations from the process. We thank a reviewer who pointed out that Meta-DENDRAL's RULEGEN (Lindsay *et al.*, 1980) had two criteria similar in spirit to the significance and stop-growth tests, i.e. RULEGEN had more task-specific requirements that a specialization ought to be "significant" and that a specialization should be an "improvement" over its parent.

Table 3. BEXA

```
PROCEDURE COVER-P($T$,beam_width)
  rule_set := empty;
  FOR each concept $C_i$ in $T$ DO
    $P$ := instances in $T$ belonging to concept $C_i$; $N := T - P$;
    REPEAT
      best_conj := Find-Best-Conjunction($P$,$N$,beam_width);
      IF best_conj NOT NULL THEN
        Add the rule 'IF best_conj THEN concept $= C_i$' to rule_set;
        $P := P - X_P(\text{best\_conj})$
    UNTIL ($P = \emptyset$) OR (best_conj = NULL);
  RETURN rule_set

PROCEDURE Find-Best-Conjunction ($P$,$N$,beam_width)
  best_conj := NULL;
  specializations := {the mgc for BEXA or the constant true with $X_P = P$ and $X_N = N$};    (1)
  WHILE specializations $\neq \emptyset$ DO                                                    (2)
    specializations := Generate-Specializations($P$,$N$,specializations,beam_width);          (3)
    FOR each conjunction $c \in$ specializations DO
      IF  $c$ is significant according to the significance test AND                          (4)
        $c$ is better than best_conj according to the evaluation function                    (5)
      THEN best_conj := $c$
    Remove from specializations all the conjunctions that cover no negative instances or    (6)
      that satisfy the additional stop-growth test;
    Retain in specializations only the beam_width best conjunctions according to the
      evaluation function                                                                     (7)
  ENDWHILE;
  IF the evaluation function value for best_conj is the same as or worse than that of the
      complete training set THEN                                                              (8)
    RETURN NULL
  ELSE
    RETURN best_conj

PROCEDURE Generate-Specializations($P$,$N$: instance_set, conjunctions : set_of_conjunctions;
    $k$ : beam_width)
  specializations := $\emptyset$;
  FOR each conjunction $c \in$ conjunctions DO
    {First remove from c.usable all the values that will lead to unnecessary specializations.}
    FOR each value or interval $a_i \in$ c.usable DO
      IF  $X_P(c) \subseteq X_P(a_i)$ OR {Prevents conjunctions for which $X_P = \emptyset$}              (9)
        $X_N(c) \cap X_N(a_i) = \emptyset$ OR {Ensures one more negative instance will be uncovered}   (10)
        $\{X_N(b_i) \mid b_i \in$ c.excludedvalues $\cup \{a_i\}\}$ is a redundant partial cover of $N$   (11)
      THEN c.usable := c.usable $- \{a_i\}$
    {Next generate all useful specializations of the conjunction}
    FOR each value $a_i \in$ c.usable DO
      $c'$ := $c$ specialized by removing $a_i$ from it;
      $X_P(c') := X_P(c) - X_P(a_i)$;
      $X_N(c') := X_N(c) - X_N(a_i)$;
      $c'$.usable := c.usable - $\{a_i\}$;
      $c'$.excludedvalues := c.excludedvalues $\cup \{a_i\}$;
      specializations := specializations $\cup \{c'\}$
  ENDFOR;
  IF $k > 1$ THEN Remove from specializations all duplicate conjunctions;
  RETURN specializations
```

The last step (Step (8)) in procedure Find-Best-Conjunction, borrowed from CN2 (Clark & Boswell, 1991), ensures that the newly generated conjunction conveys more information than the default rule that matches all the training instances. A NULL conjunction is returned if the best conjunction's Laplace estimate is lower than that of the complete training set. This terminates the generation of further conjunctions.

4.2. *BEXA's specialization model*

Procedure Generate-Specializations employs BEXA's specialization model. Given a set of conjunctions, it returns the subset of their specializations that must be evaluated. Conjunctions are specialized by excluding single values from them as described in the previous section. One difference is that for linear attributes, intervals of the form $[A_i > a_i]$ and $[A_i \leq a_i]$ are excluded from a conjunction instead of the single value $[A_i = a_i]$. The reason is that excluding single values may lead to conjunctions that contain linear atoms of the form

```
[outlook = sunny][(15≤temp<17) ∨ (17<temp<22) ∨ (22<temp<29) ∨
(29<temp≤35)].
```

This conjunction is obtained by excluding the values 17, 22, and 29 for attribute `temp` from the **mgc**. These conjunctions tend to have a very low accuracy on unseen instances. For example, if `temp` has a real-valued domain, then only those instances that take the values 17, 22, and 29 will not match the above conjunction. Excluding intervals of the form $[A_i > a_i]$ and $[A_i \leq a_i]$, instead of single values, leads to linear atoms of the form $[a_i < A_i \leq b_i]$. These atoms can exclude many negative instances, and are also simpler to understand than the one given above.

Some efficiency improvements implemented in BEXA are mentioned below. (1) Extensions of values for each concept are computed once only and stored in memory for subsequent use during specialization. (2) Generating intervals for linear values instead of excluding only the single value doubles the number of values to consider for each linear attribute. In some domains only a subset of these intervals are considered. Firstly, all the observed values of a linear attribute are sorted from small to large. Then the extension in T of each value is determined. If the instances in the extensions of two or more consecutive values all belong to the same concept, then intervals need only be created for the largest such value and the predecessor of the smallest value. The reason is that intervals in between do not provide any additional information for distinguishing among different concepts. For example, for `TS` no intervals involving values of `temp` less than 22 or greater than 28 need be considered, reducing the number of intervals from twenty to eight. A similar result was proved by Fayyad and Irani (1992) for TDIDT methods employing the entropy function. (3) Intervals are deleted from the `usable` set (discussed in the next section) at the earliest possible moment, i.e. when an interval is selected, other intervals not included by it are deleted to prevent subsequent useless specialization effort.

4.3. *BEXA's search restrictions*

BEXA imposes three restrictions on the search process, i.e. on the number of specializations that are constructed at each specialization step. The search restrictions are employed by maintaining a `usable` and `excludedvalues` set for each specialization. Initially, the usable set contains all the nominal attribute values and all the intervals that are created for linear attribute values, while the excludedvalues set is empty. A conjunction is only specialized by excluding values and intervals in its usable set. Each value or interval that is excluded from a conjunction is removed from its usable set and then added to its excludedvalues set. Values and intervals that satisfy any of the search restrictions are deleted from the usable set to prevent the unnecessary construction of useless specializations at subsequent specialization steps.

4.3.1. *The prevent-empty-conjunctions restriction*

The **prevent-empty-conjunctions** restriction (Step (9) in Table 3) prevents the generation of empty conjunctions, that is conjunctions that cover no positive instances. Such conjunctions are useless since the goal is to find conjunctions that cover all the positive instances. The prevent-empty-conjunctions restriction thus avoids the unnecessary generation of specializations that are known to be useless. This restriction requires almost no additional overhead to employ since the number of positive instances covered by a conjunction must in any case be determined to calculate its Laplace estimate. The same holds for the evaluation functions employed by the other covering algorithms described in this paper (see Table 4). Section 6.1 gives empirical evidence that this restriction reduces BEXA's learning time without decreasing the accuracy of its descriptions.

4.3.2. *The irredundancy restriction*

The purpose of the **irredundancy** restriction (Step (11) in Table 3) is to guide BEXA to construct conjunctions in C_M. The reason, as explained in Section 3, is that these conjunctions are likely to contain only a few atoms and to have a high value for the evaluation function. This restriction thus reduces the number of possible specializations that must be constructed while at the same time increasing the chance of finding conjunctions with only a few atoms that cover many positive instances. The irredundancy restriction may also reduce the number of specializations steps that are required to find a consistent conjunction. The reason is that BEXA starts with the **mgc** and then specializes a conjunction by excluding values and intervals. Fewer values thus need to be excluded to find a conjunction $c \in C_M$ than to find a conjunction $c' < c, c' \in (C_C - C_M)$ because c' is "lower down" in the lattice than c.

Irredundancy is ensured using the scheme described in Wells (1971) for generating the irredundant covers of a set. The basic idea is to maintain for each conjunction an additional duplicates set that contains all those negative instances that are matched by more than one value or interval in a conjunction's excludedvalues set. A value or interval is discarded from a conjunction's usable set if its exclusion will cause the negative extension of any value or interval in the new excludedvalues set to be contained in the duplicates set. Performing the irredundancy test then requires a set intersection and a set union operation to determine the new duplicates set, and one subset test for each value in a conjunction's new excludedvalues set. The number of subset tests that must be performed increases by one with each additional value that is excluded from a conjunction. If many values must be excluded before a consistent conjunction is found, or before the stop-growth test terminates further specialization, then it may be suspected that the additional overhead required by this restriction may outweigh the savings obtained by reducing the number of specializations that must be evaluated. However, Section 6.1 presents empirical evidence that the irredundancy restriction improves the quality of BEXA's concept descriptions and even reduces the learning time.

4.3.3. *The uncover-new-negatives restriction*

The **uncover-new-negatives** restriction (Step (10) in Table 3) is a necessary requirement for the irredundancy restriction as explained in Section 3. It ensures that each newly excluded value or interval uncovers at least one new negative instance. It is employed as a separate restriction because it is more efficient to implement than the irredundancy test. As is the case for the prevent-empty-conjunctions test, no additional overhead is required to employ this test since the number of negative instances covered by a conjunction must be determined for all the evaluation functions described in Table 4. For this reason the uncover-new-negatives restriction should always be enforced before the irredundancy restriction to reduce the number of more expensive irredundancy tests. Section 6.1 presents empirical evidence that the uncover-new-negatives restriction improves the quality of BEXA's concept descriptions substantially and reduces its learning time.

4.4. *A practical example*

For the illustration of BEXA assume a beam width of one, and that only consistent conjunctions are discarded from the set of specializations (i.e. the log-likelihood ratio test is not employed as an additional stop-growth test). A description for the concept in $\texttt{TS}$ is constructed as follows.

Constructing the first conjunction:

The relevant data for selecting the first value to exclude from the conjunction is:

$$\begin{array}{lcl}
\text{conjunction} & = & [\text{outlook} \in \{\text{sunny,overcast,rain}\}][\text{autumn} \in \{\text{yes,no}\}][15 \leq \text{temp} \leq 35] \\
X_P(\text{conjunction}) & = & \{10,11,12,13\} \\
X_N(\text{conjunction}) & = & \{1,2,3,4,5,6,7,8,9\} \\
\text{excludedvalues} & = & \emptyset \\
\text{usable} & = & \{\text{the values listed below}\}
\end{array}$$

In the following new_conj denotes the new conjunction obtained by excluding the listed value from the current conjunction. Laplace denotes the Laplace accuracy estimate of this new conjunction.

Value	X_P(value)	X_N(value)	X_P(new_conj)	X_N(new_conj)	Laplace
[outlook = overcast]	{11}	{2,6,7}	{10,12,13}	{1,3,4,5,8,9}	0.364
[outlook = sunny]	{10,12}	{1,4,5}	{11,13}	{2,3,6,7,8,9}	0.300
[outlook = rain]	{13}	{3,8,9}	{10,11,12}	{1,2,4,5,6,7}	0.364
[autumn = yes]	{10}	{1,3,4,6,8}	{11,12,13}	{2,5,7,9}	0.444
[autumn = no]	{11,12,13}	{2,5,7,9}	{10}	{1,3,4,6,8}	0.250
[temp ≤ 22]	∅	**{1,2,3,4}**	**{10,11,12,13}**	**{5,6,7,8,9}**	**0.455**
[temp ≤ 23]	{11,13}	{1,2,3,4,8}	{10,12}	{5,6,7,9}	0.429
[temp > 23]	{10,12}	{5,6,7,9}	{11,13}	{1,2,3,4,8}	0.375
[temp ≤ 27]	{11,12,13}	{1,2,3,4,8,9}	{10}	{5,6,7}	0.333
[temp > 27]	{10}	{5,6,7}	{11,12,13}	{1,2,3,4,8,9}	0.364
[temp > 28]	∅	{5,6,7}	{10,11,12,13}	{1,2,3,4,8,9}	0.417

Intervals like [temp ≤ 21] and [temp > 29] are not considered as explained in Section 4.2. Intervals [temp > 22] and [temp ≤ 28] are discarded with the prevent-empty-conjunctions test. Excluding [temp < 22] leads to the conjunction with the highest Laplace value.

The relevant data for selecting the second value to exclude from the conjunction is:

conjunction	=	[outlook ∈ {sunny,overcast,rain}][autumn ∈ {yes,no}][22 < temp ≤ 35]
X_P(conjunction)	=	{10,11,12,13}
X_N(conjunction)	=	{5,6,7,8,9}
excludedvalues	=	{[temp ≤ 22]}
usable	=	{the values listed below}

Value	X_P(value)	X_N(value)	X_P(new_conj)	X_N(new_conj)	Laplace
[outlook = overcast]	{11}	{6,7}	{10,12,13}	{5,8,9}	0.500
[outlook = sunny]	{10,12}	{5}	{11,13}	{6,7,8,9}	0.375
[outlook = rain]	{13}	{8,9}	{10,11,12}	{5,6,7}	0.500
[autumn = yes]	{10}	{6,8}	{11,12,13}	{5,7,9}	0.500
[autumn = no]	{11,12,13}	{5,7,9}	{10}	{6,8}	0.400
[temp > 23]	{10,12}	{5,6,7,9}	{11,13}	{8}	0.600
[temp > 27]	{10}	{5,6,7}	{11,12,13}	{8,9}	0.571
[temp > 28]	∅	**{5,6,7}**	**{10,11,12,13}**	**{8,9}**	**0.625**

The values [temp ≤ 23] and [temp ≤ 27] are discarded with the irredundancy test. The value [temp > 28] is selected because it leads to the new conjunction with the highest Laplace value. This conjunction is better than its predecessor and becomes the new best conjunction.

The relevant data for selecting the third value to exclude from the conjunction is:

conjunction	=	[outlook ∈ {sunny,overcast,rain}][autumn ∈ {yes,no}][22 < temp ≤ 28]
X_P(conjunction)	=	{10,11,12,13}
X_N(conjunction)	=	{8,9}
excludedvalues	=	{[temp ≤ 22], [temp > 28]}
usable	=	{the values listed below}

Value	X_P(value)	X_N(value)	X_P(new_conj)	X_N(new_conj)	Laplace
[outlook = rain]	**{13}**	**{8,9}**	**{10,11,12}**	∅	**0.800**
[autumn = yes]	{10}	{8}	{11,12,13}	{9}	0.667
[autumn = no]	{11,12,13}	{9}	{10}	{8}	0.500

Values like [outlook = sunny] that would lead to a redundant cover of N have been discarded. The value [outlook = rain] is selected, and the new conjunction

[outlook ⊂ {overcast,sunny}][autumn ∈ {yes,no}][22 < temp ≤ 28]

is again better than the current best conjunction. At this stage, the conjunction has become consistent because it covers no negative instances. It is simplified by deleting all the atoms where attributes take all their possible values to yield

[outlook ∈ {overcast,sunny}][22 < temp ≤ 28]

This conjunction is then returned as the best conjunction that could be found. This conjunction covers the positive instances 10, 11, and 12. The same process is now repeated to obtain the next consistent conjunction covering the remaining positive instance 13. This yields the two rules

$[\text{outlook} \in \{\text{overcast,sunny}\}][22 < \text{temp} \leq 28] \quad \Rightarrow \quad +$
$[\text{autumn} = \text{no}][22 < \text{temp} \leq 23] \quad \Rightarrow \quad +$

If temp has an integer domain, then the atom $[22 < \text{temp} \leq 23]$, obtained by excluding the values $[\text{temp} \leq 22]$ and $[\text{temp} > 23]$, can be replaced with $[\text{temp} = 23]$.

5. Differences among the covering algorithms

This section describes the differences among the set covering algorithms BEXA, AQ15, CN2, PRISM, GREEDY3, and GALG, showing that they all fit into the same general set covering framework. Table 4 summarizes the differences, while the following sections discuss the differences in more detail where necessary.

5.1. *The top-level loop*

All the covering algorithms discussed in this paper employ the top-level loop given in procedure COVER-P for BEXA (Table 3). This is not surprising since this loop embodies the covering nature of these algorithms: Conjunctions are constructed until all the positive instances of a concept are covered.

The first version of CN2 (Clark & Niblett, 1989) generated an ordered list (or *decision list*). Clark & Boswell (1991) modified CN2 to generate either ordered or unordered (production) rules merely by changing its top-level loop and evaluation function. Theron (1994) showed that the same holds for the other algorithms.

5.2. *Controlling the generation of conjunctions*

All the covering algorithms construct conjunctions by following a general-to-specific search. BEXA starts with the **mgc** (Step (1) in Table 3) and specializes it by excluding values. The remaining algorithms start with the constant **true** and specialize a conjunction by appending atoms. The next four subsections discuss the most important issues regarding the generation of conjunctions. Procedure Find-Best-Conjunction in Table 3 serves as a general control structure for this discussion.

5.2.1. *Specialization models*

A covering algorithm's specialization model determines which specializations are constructed for a conjunction at each specialization step. It can be viewed as a separate procedure that, given a conjunction or set of conjunctions, returns all the specializations of these conjunctions that must be evaluated (Step (3) in Table 3). In the actual implementations of the different algorithms, aspects of the model and the control structure may be interwoven for the sake of efficiency. The algorithms restrict the search effort for a good conjunction by

Table 4. Similarities and differences among BEXA, CN2, AQ15, PRISM, GREEDY3 and GALG

Language restrictions

	BEXA	AQ15	CN2	PRISM	GREEDY3	GALG
nominal attributes	√	√[1]	√	√	√[2]	√
linear attributes	√	√	√	–	–	–
internal disjunction	√	√	–	–	–	√

[1] AQ15 also allows tree-structured attributes (Hoff *et al.*, 1983)

[2] GREEDY3 allows only binary attributes.

Optional features in the control structure (c denotes a conjunction)

	BEXA	AQ15	CN2	PRISM	GREEDY3	GALG
Beam widths > 1	√	√	√	–	–	–
Significance test	log-likelihood	–	log-likelihood	–	–	–
Stop-growth test[3]	log-likelihood	–	cannot improve best conjunction	–	–	$\mid X_N(c) \mid < T$
Post-pruning	√	√	–	–	√	√

[3] Only stop-growth thresholds that are applied in addition to the standard $\mid X_N(c) \mid = 0$ test are given. T denotes a threshold.

Evaluation functions (c denotes a conjunction)

Algorithm	**Evaluation function**
CN2[1]	maximize $\frac{\mid X_P(c) \mid + 1}{\mid X_P(c) + \mid X_N(c) \mid + \#\mathrm{concepts}}$ where #concepts is the total number of concepts in the training set.
AQ15[2]	$< (\mathrm{minimize} \mid X_N(c) \mid, t_1), (\mathrm{maximize} \mid X_P(c) \mid, t_2) >$
BEXA	same as CN2's
PRISM	$< (\mathrm{maximize} \frac{\mid X_P(c) \mid}{\mid X_P(c) \mid + X_N(c) \mid}, 0), (\mathrm{maximize} \mid X_P(c) \mid, 0) >$
GREEDY3	maximize $\frac{\mid X_P(c) \mid}{\mid X_P(c) \mid + X_N(c) \mid}$
GALG	Gray (1988) states that attributes are scored according to the number of positive instances that they might be expected to predict after allowance for pure chance occurrences. The chance occurrences are computed using the chi-square test.

[1] The first version of CN2 that generated only ordered rules employed the entropy function.

[2] This is a typical Lexicographic Evaluation Function (LEF) that can be applied at both Steps (5) and (7) in Table 3 since $\mid X_N(c) \mid = 0$ at Step (5). A LEF is an ordered list $< (f_1, t_1), \ldots, (f_n, t_n) >$ where each f_i is an evaluation function with an associated tolerance t_i.

restricting the representation language (summarized in Table 4) and by constructing only a subset of all possible specializations at each step.

For nominal attributes BEXA constructs all the useful specializations of a conjunction at the next lower level in the lattice. For linear attributes, BEXA constructs specializations

Table 5. AQ15's specialization model

```
PROCEDURE Generate-Specializations ($P$,$N$: instance_set; conjunctions : set_of_conjunctions;
    $k$ : beam_width)
  IF conjunctions contains only true THEN select a seed instance $s$ from $P$;
  $n_e$ := any instance in $N$ covered by any one of the conjunctions in conjunctions;
  atoms := $\emptyset$;
  FOR each attribute $A_i$ that takes a different value in $s$ and $n_e$ DO
    atoms := atoms $\cup$ $\{[A_i \in (D_i - \{a_i\}]\}$ where $D_i$ is the domain of $A_i$ and $a_i$ is its value in $n_e$
  specializations := $\{x \wedge y \mid x \in$ conjunctions, $y \in$ atoms$\}$;
  Remove from specializations all conjunctions subsumed by others in specializations;
  RETURN specializations
```

by excluding single intervals. This amounts to constructing all useful specializations that can be formed by increasing the lower bound of an interval, or by decreasing the upper bound of an interval, while at the same time removing the resulting useless specializations to improve efficiency.

All the AQ algorithms use the same basic specialization model (Michalski *et al.*, 1986), given in Table 5 according to Clark & Niblett (1989) and Hoff *et al.* (1983). This table contains **only** the specialization model which is called repeatedly by procedure Find-Best-Conjunction given in Table 3 until all the conjunctions in the current set of specializations (the *star*) become consistent. AQ15 specializes a conjunction by appending the most general atoms that cover the seed but not a negative instance. Such atoms contain all the possible values for an attribute except those that occur in the negative instance. AQ15 restricts the number of specializations at each step by constructing at most a specializations of a conjunction, given that there are a attributes, since a seed and negative instance can take a different value for at most a attributes. In contrast, BEXA may construct up to v specializations for a conjunction at the first specialization step, where v is the number of nominal attribute values plus twice the number of linear attribute values. Remember that BEXA creates two intervals for each linear attribute value. For example, $v = 2a$ for a learning problem with only binary nominal attributes, otherwise ($v > 2a$). The number of specializations that BEXA constructs decreases with each specialization step as its optional search restrictions weed out useless specializations.

CN2 was designed to extend the number of specializations considered by AQ15 and to rid AQ15 of its dependence on a seed and negative instances (Clark & Niblett, 1989). It therefore generates all possible specializations of a conjunction, but restricts its description language to pure concept descriptions. Table 6 contains CN2's specialization model[2]. For each nominal attribute, elementary atoms of the form $[A_i = a_i]$ are created for each one of its values. One new specialization is then created for each such atom by appending the atom to the current conjunction, provided that the attribute does not already appear in it. This prevents the construction of NULL conjunctions such as `[big = yes][big = no]`. CN2 thus excludes all but a single value from a nominal attribute. Each specialization step of CN2 is thus equivalent to a number of specialization steps in AQ15 and BEXA. For a linear attribute, all the values for that attribute that occur in the instances covered by the current conjunction are determined. The values are sorted from small to large. For each

Table 6. CN2's specialization model

```
PROCEDURE Generate-Specializations ($P$,$N$: instance_set; conjunctions : set_of_conjunctions;
    $k$ : beam_width)
  specializations := empty;
  FOR each conjunction $c$ in conjunctions DO
    FOR each nominal attribute $A_i$ that does not appear in $c$ DO
      FOR each value $a_i$ of $A_i$ DO
        Add the conjunction $c \wedge [A_i = a_i]$ to specializations;
    FOR each linear attribute $A_i$ that does not have both bounds set in $c$ DO
      Sort the values of $A_i$ that occur in $X_T(c)$ from small to large;
      FOR each value, from small to large, DO
        Specialize c by appending the atom $[A_i < a_i]$;
        Add to specializations if the cut-point is a local maximum;
        Do the same for intervals of the form $[A_i > a_i]$;
  Remove duplicate conjunctions from specializations;
  RETURN specializations
```

such value, the intervals $[A_i < a_i]$ and $[A_i > a_i]$ are created, where a_i is midway in between the value and its successor. All the specializations that can be formed by appending the atoms of the form $[A_i < a_i]$ to the current conjunction are then evaluated, and those that are local maxima (according to the evaluation function) are added to the set of specializations. The same is done for the atoms of the form $[A_i > a_i]$. CN2 determines at most one lower and one upper bound for a linear interval. As soon as both bounds have been determined, no further specializations are considered that involve that particular linear attribute. CN2 and BEXA thus handle linear attributes in almost the same way. The algorithms differ in that CN2 may discard some of the specializations as they are constructed (those that are not local maxima), while BEXA adds all the possibilities to the set of specializations. Lastly, CN2 adds a check to prevent duplicate conjunctions from being added to the set of specializations.

CN2 requires at most a specialization steps to construct a consistent conjunction when all the attributes are nominal because all but one value is excluded for an attribute when specializing a conjunction. In contrast to CN2, AQ15 and BEXA may require up to $v - a$ steps for a conjunction that covers exactly one instance (at least one value per attribute is required at the lowest level of the lattice).

PRISM generates pure descriptions allowing only nominal attributes. Starting with **true**, it appends single nominal values to the current conjunction until it covers only positive instances. GALG's specialization model is almost identical to that of PRISM. The only difference is that it also appends the complements of attribute values to conjunctions, thus implicitly allowing internally disjunctive expressions. Thus GALG can exclude single values as do BEXA and AQ15. For example, a conjunction may be specialized by appending either the value $\mathtt{sunny}$ or the value $\neg\mathtt{sunny}$ (not sunny). GREEDY3 only allows binary attributes. It specializes a conjunction by appending literals to it (i.e. an attribute or its complement).

Below examples are given of the search paths of BEXA, AQ15 and CN2 through the lattice of conjunctions in Figure 1 (see Table 2), and the relationship between excluding values versus appending atoms is illustrated.

The conjunctions constructed by BEXA are underlined.

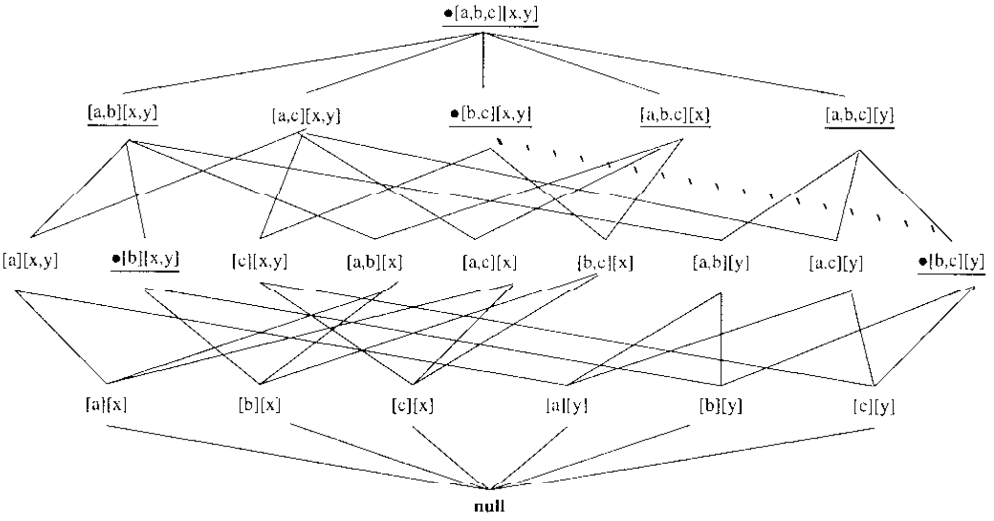

Figure 2. BEXA's path through the lattice of conjunctions

BEXA does not construct the specialization obtained by excluding [A = b] because it does not uncover new negative instances. It is removed from the usable set. Arbitrarily select [A ∈ {b,c}] [B ∈ {x,y}] from the specializations with the highest Laplace estimates obtained by excluding [A = a] or [A = c], and delete [A = a] from its usable set. Next the three specializations formed by excluding [A = c], [B = x] or [B = y] are considered. The value [A = b] is not considered because it had been removed from the usable set, while [B = y] is deleted because excluding it does not uncover new negative instances. The best conjunction is any one of the remaining two specializations, [A = b] [B ∈ {x,y}] or [A ∈ {b,c}] [B = y], obtained by excluding either [A = c] or [B = x].

In Figure 3 assume that AQ15 selected instance 3 as seed and instance 2 as the first negative instance. The most general atom for A that covers 3 and not 2 is [A ∈ {b,c}]. Similarly, the most general atom for B that covers 3 and not 2 is [B = x]. Appending [A ∈ {b,c}] to **true** is equivalent to excluding the value [A = a], taken by the negative instance, from the **mgc** to yield [A ∈ {b,c}] [B ∈ {x,y}]. Similarly, appending [B = x] to **true** is equivalent to implicitly excluding the value [B = y] from the **mgc** to obtain [A ∈ {a,b,c}] [B = x]. The former specialization is the best and must be further specialized to uncover the negative instance 5. Instances 3 and 5 differ for attribute A, leading to the atom [A ∈ {a,b}]. Appending this atom to the current conjunction (equivalent to excluding the value [A = c]) yields the consistent conjunction [A = b] [B ∈ {x,y}] written as [A = b].

Figure 4 illustrates CN2's search for the second conjunction, because CN2's first specialization step selects [A = b] trivially from the created conjunctions [A = a], [A = b], Conjunctions constructed by AQ15 are underlined. The one selected at each level is marked by ♦.

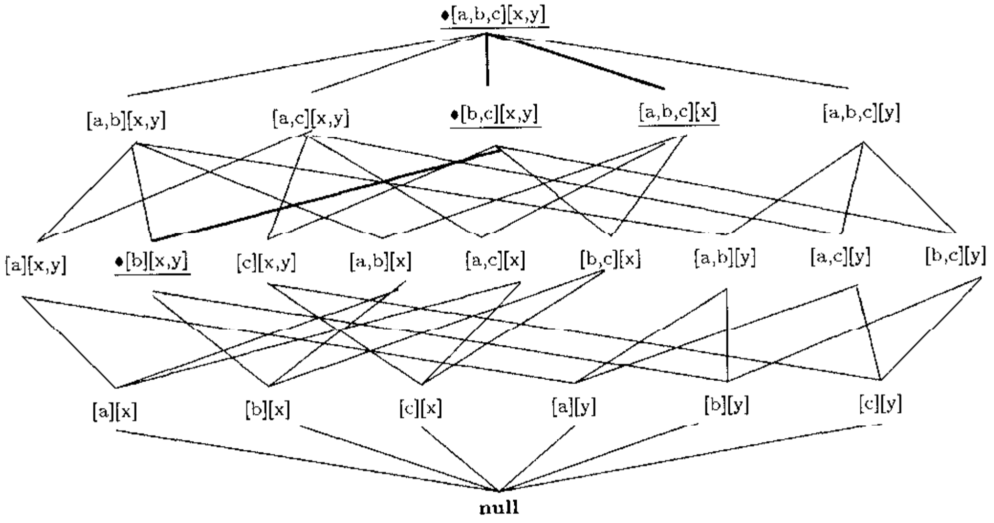

Figure 3. AQ15's path through the lattice of conjunctions

[A = c], [B = x], and [B = y]. The two positive instances that it covers are discarded. For the second conjunction, the first specializations considered are again [A = a], [A = b], [A = c], [B = y], and [B = x] (underlined on the second and third row). Any one except [A = b] can be selected as the best since they cover one positive and one negative instance. Select [B = y] arbitrarily. The next step considers the non-null specializations [A = a][B = y], [A = b][B = y], and [A = c][B = y]. [A = c][B = y] is selected because it covers a positive instance.

CN2 may skip levels in the lattice because it implicitly excludes a nominal attribute's complement. For example, [A = a] (shorthand for [A = a][B ∈ {x,y}]) implicitly excludes [A ∈ {b,c}] from the **mgc**. In contrast, BEXA can consider all specializations at a level and AQ15 at most one per attribute.

Rymon (1993) also describes a general learning framework, but with emphasis on a particular data structure, i.e. an SE-tree which "shares many features of decision tree based algorithms" (p.268) and "generalizes decision trees" (p.274). We describe a general framework for *set covering*, as opposed to the recursive partitioning framework (divide-and-conquer used by the Top Down Induction of Decision Trees approach to learning), and show how five other algorithms fit into such a framework. (It can be argued that using set covering, which does not repeatedly split the training set, has "more evidence" available at each specialization step because only the covered positive examples are removed.) Specialization in an SE-tree occurs by appending, not by excluding values. SE-tree rules are interpreted conjunctively, while the **mgc** has a conjunctive interpretation but with internal disjunction; however, it seems that disjunction cannot be represented directly, but has to be resolved during the classification phase (see the OR example with a resolution criterion based on voting, p.274). The remarks on "hypothesis expressibility" (p.273) seem to sug-

Conjunctions considered by CN2 are underlined. The one selected at each level is marked by ♦.

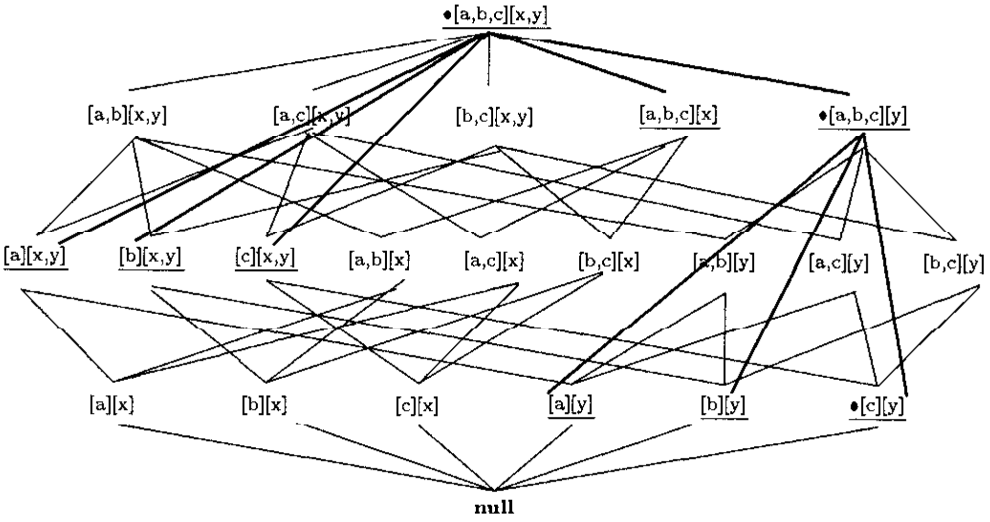

Figure 4. CN2's path through the lattice of conjunctions (when constructing the second conjunction)

gest that some hypotheses cannot be expressed as succinctly as with internal disjunction (the same problem that CN2 has with its restriction to pure descriptions). Impotent expansion 1 (p.270) corresponds to BEXA's prevent-empty-conjunctions. Rymon (1993) does not give an empirical evaluation of this restriction; we present empirical results in Section 6. Impotent expansion 2 (p.270) for rules with the same extension corresponds to BEXA's equality test for duplicate conjunctions. BEXA does not require an expensive subsumption test due to its specialization method, while "procedure expand" (p.271) needs to test for subsumption. We share the bias of finding maximal general solutions, which BEXA can also adjust to the learning problem as elaborated on again later ((Rymon, 1993) (p.269): "Intuitively, while learning, we adopt most general principles"). BEXA can (optionally) guarantee that a conjunction in C_M (a "kernel" element) will be found. Selection of conjunctions are prioritized in the SE-tree representation, while BEXA does so in the lattice, selecting sets in C_M. BEXA already has a stop-growth criterion (see Future research directions) and makes provision for intervals.

5.2.2. *Complexity of specialization*

In this section the worst case complexity of specialization is summarized, assuming that consistent and complete conjunctions are required, using a beam width of one. Assume there are p positive and n negative examples and the a attributes take v possible values in total: Then all the covering algorithms generate conjunctions until all the positive examples are covered, i.e. at least one and at most p conjunctions will be generated. Now consider the generation of a single conjunction. AQ requires at most n specialization steps for a consistent conjunction if only one negative example is excluded per step. CN2 performs at most a steps since only one attribute value is retained at each step, while BEXA requires at most $v - a$ steps because only a single value is excluded from the conjunction per step. Now consider the total number of specializations constructed to obtain a consistent conjunction: AQ thus generates at most an specializations to obtain a consistent conjunction; BEXA generates v specializations at the first step, then $v - 1$ for the next, and so on until all attributes take only a single value; and CN2 generates v specializations at the first step, then the sum of the number of values in the remaining attributes, and so on until all attributes take only a single value.

To compare CN2 and BEXA in more detail, assume a learning problem with only nominal attributes, each taking k values, then $v = ka$. CN2 generates at most $ka+k(a-1)+\ldots+k \approx ka^2/2$ specializations, while BEXA generates at most $ka + ka - 1 + \ldots \approx k^2a^2/2$; thus BEXA constructs a factor k more specializations.

If BEXA applies Test 2, then the number of steps to obtain a consistent conjunction is at most the minimum of $v - a$ and n (since, like AQ, at least one negative example must be excluded at each specialization step).

The complexity analysis above, however, cannot quantify the number of specializations not considered due to BEXA's optional search restrictions because it is highly dependent on the training data. We provide empirical evidence in Section 6 that BEXA's search restrictions is very effective in weeding out unnecessary specialization effort, even though BEXA allows internal disjunction.

5.2.3. *Employing a beam search*

When generating conjunctions the beam width number of best alternative specializations are retained for the next step (Step (7) in Table 3). Further specialization ceases when the set of alternative best conjunctions becomes empty (Step (2)) due to the stop-growth tests (discussed in Section 5.2.5). Only BEXA, AQ15 and CN2 allow a user-specified beam width (Table 4).

The motivation for using beam widths greater than one is that it increases the chance of finding a simple conjunction covering many positive instances. However, large beam widths can increase the computational requirements substantially. Theron & Cloete (1993) and Theron (1994) presented empirical evidence for ten test databases that beam widths of ten or twenty usually do not produce significantly better rules than a beam width of one for BEXA. Since AQ15 limits the number of specializations considered (see Section 5.2.1) a large beam width is more likely to improve the descriptions found because it increases the total number of different specializations that are considered. Similarly, the experiments reported in Section 6.2.1 suggest that CN2's restriction to pure conjunctions also benefits from using beam widths larger than one.

5.2.4. *Selecting the best specializations*

Covering algorithms employ *evaluation functions* to select the best conjunction (Step (5) in Table 3) and to retain the beam width number of best specializations (Step (7) in Table 3) after each specialization step. AQ15 employs a combination of different evaluation functions called a Lexicographic Evaluation Function (LEF) (Michalski, 1983). A LEF is an ordered list $< (f_1, t_1), ..., (f_n, t_n) >$ where each f_i is an evaluation function with an associated tolerance t_i. The functions are applied in the order in which they are specified. When two conjunctions do not differ more than the specified tolerance for f_i, they are evaluated with respect to f_{i+1}. AQ15 typically employs two different LEF's at Steps (5) and (7). For example, let c denote a conjunction. Then the LEF <(minimize $| X_N(c) |$, t_1), (maximize $| X_P(c) |$,t_2)> can be applied at Step (7) and the LEF <(maximize $| X_P(c) |$, t_1), (minimize the number of atoms in c,t_2)> at Step (5). All the other covering algorithms, including BEXA, employ the same evaluation function or LEF at both Steps (5) and (7). The different evaluation functions are listed in Table 4.

5.2.5. *Stopping further specialization*

An important question to address is when to terminate further specialization of a conjunction. This decision is employed in the form of a stop-growth test (Step (6) in Table 3). When all conjunctions in the current set of specializations satisfy this test, the set becomes empty. This terminates the search for the best conjunction (Step (2) in Table 3). Further specialization of a conjunction can be stopped the moment it becomes clear that additional specialization will not improve the conjunction's value for the evaluation function. The standard criterion is to terminate further specialization when a conjunction becomes consistent, i.e. $| X_N(c) | = 0$. See Table 4 for a summary.

AQ15 employs a slightly weaker version of this standard stop-growth test: It stops further specialization when all the conjunctions in the current set become consistent. The best conjunction in this set is returned. Some conjunctions may become consistent before others do. These conjunctions will be specialized further by AQ15 until all the conjunctions in the current set are consistent. This unnecessary specialization effort can be avoided by removing a conjunction from the set of specializations the moment that it becomes consistent (Step (6) in Table 3). The best overall conjunction will still be returned because the best conjunction is retained after each specialization step.

CN2 employs both a weaker and a stronger version of the standard stop-growth test. The weaker version terminates further specialization only when a conjunction covers no negative and no positive instances. The stronger version terminates further specialization of a conjunction when it cannot improve on the current best conjunction. This is determined by calculating a conjunction's Laplace estimate under the assumption that it covers no negative instances. If this estimate is lower than the current best estimate, further specialization is terminated because none of the conjunction's specializations can improve on the current best conjunction. This second test also implicitly stops further specialization when a conjunction becomes consistent, since its specializations can only cover fewer positive instances and therefore have a lower Laplace estimate than the current conjunction.

5.2.6. *Pre-pruning*

It is well known that pruning concept descriptions in the presence of noise leads to considerably simpler descriptions that usually have higher accuracy on unseen instances than the unpruned descriptions (Quinlan, 1987a; Michalski *et al.*, 1986). Two types of pruning are distinguished (Breiman *et al.*, 1984). *Pre-pruning* tests are applied during the specialization process, while *post-pruning* is applied to a complete set of rules, i.e. as a post-processor. This section discusses pre-pruning and the next post-pruning.

Only BEXA, CN2 and GALG employ pre-pruning schemes. GALG can stop further specialization when the number of negative instances covered by a conjunction is below a user-specified level. CN2 uses a significance test (Step (4) in Table 3 to ignore conjunctions that do not capture significant patterns in the training data. BEXA performs pre-pruning with an additional stop-growth test (Step (6) in Table 3) that discards all the insignificant specializations of a conjunction. The differences between the latter two approaches were discussed in Section 4.1.

CN2 performs another check that can also be viewed as a form of pre-pruning. It checks whether the Laplace estimate of the best conjunction is greater than that of the default rule (Clark & Boswell, 1991). If this is not the case, then the new conjunction does not contribute any new information and the NULL conjunction is returned (Step (8) in Table 3), terminating the generation of conjunctions for the current concept. BEXA borrowed this test from CN2.

5.3. *Post-pruning*

BEXA employs the post-pruning scheme described by Quinlan (1987a,1987b). This scheme prunes rules in two phases. The first phase simplifies individual rules by deleting atoms that are insignificant according to Fisher's exact test. The second phase simplifies the set of rules by deleting those rules that will not reduce the accuracy of the collection as a whole on the training instances. A detailed description of this pruning scheme can be found in Quinlan (1987a,1987b).

CN2 and PRISM do not employ any post-pruning scheme. GALG only post-prunes individual rules by deleting those atoms that will not reduce the accuracy of the rule on the training set (Gray, 1988). This ensures that no conjunction contains any redundant tests. GREEDY3 follows an approach opposite to that of GALG. It simplifies a rule by first discarding all its atoms. A process analogous to the specialization process is then followed by returning atoms to the conjunction while this will increase its accuracy on a separate test set (Pagallo & Haussler, 1990).

AQ15 post-prunes a set of rules in one of two ways (Michalski *et al.*, 1986). The first approach is to discard all the conjunctions describing a particular concept that cover fewer than two positive instances uniquely. The alternative approach is to discard all the conjunctions for a particular concept except the one covering the most positive instances. AQ15 thus does not simplify individual rules[3].

Table 7. The ten test databases

Database	Ref.	Description	#Inst., Attr., Class	References
lymphography	lym	identify disease	148, 18, 4	(Michalski *et al.*, 1986)
breast cancer	bcr	predict recurrence	286 , 9, 2	(Michalski *et al.*, 1986)
primary tum.	pt	predict tumour type	339, 17, 21	(Michalski *et al.*, 1986)
iris	irs	identify iris type	150, 4, 3	(Mingers, 1989)
voting	vot	predict democrat or republican senator	435, 16, 2	(Clark & Boswell, 1991)
digit	dgt	identify led digit	500, 7, 10	(Mingers, 1989)
soybean	soy	identify disease	307, 35, 19	(Michalski & Chil., 1980)
tictactoe	ttt	predict win or loose	958, 9, 2	(Wirth & Catlett, 1988)
internal disj.	id	predict class or not	500, 5, 2	
cleveland	cle	predict presence of heart disease	303, 13, 2	(Clark & Boswell, 1991)

6. Empirical evaluation of BEXA

This section presents an empirical evaluation of BEXA's three search restrictions. Experiments are reported where search restrictions are added one by one to show that they reduce the learning time and usually improve description quality. BEXA is also compared to the state-of-the-art concept learners CN2 and C4.5. A detailed comparison with CN2 is presented because both BEXA and CN2 employ the covering approach to concept learning. C4.5 is compared to BEXA since it serves as a yard-stick for rule quality.

It has become common practice to compare learning algorithms on a number of different test databases (Clark & Boswell, 1991; Quinlan, 1987b; Buntine & Niblett, 1992) because comparisons on a small number of databases do not yield conclusive results. We report results for ten test databases (see Table 7) that were selected because they are representative of many different types of learning problems. They differ regarding the number of training instances that are available, the degree of noise in these instances, the number of concepts to learn and the proportion of instances belonging to each concept, the number of nominal, integer and real valued attributes used to describe the instances, and the application domain from which the data was obtained. For each database an abbreviation is given to denote it in tables and graphs. All the databases, except internal disjunction (id), have been used by other authors. References are given where the interested reader can find more detail about these databases. We constructed the internal disjunction database to illustrate the advantage of generating internally disjunctive instead of pure descriptions for some databases. Instances in this database are described with five attributes, A to E, where each attribute takes five possible values, e.g. a_1 to a_5. There is only one concept with the disjunctive description

$$[A \in \{a_1, a_2, a_3\}][B \in \{b_3, b_4\}] \vee [B \in \{b_2, b_4, b_5\}][C \in \{c_3, c_4, c_5\}]$$

Five hundred random instances were generated and classified as positive or negative, depending on whether they matched this description or not. Noise was then introduced in each instance by replacing each on of its five attribute values with a probability of 10% with another, and by flipping the instance's classification with a probability of 10%. This database thus contains noisy instances and two irrelevant attributes, D and E.

Ten random training and test sets were generated for each of the ten databases. In each case, the training set contained 70% of the available instances while the remaining 30% constituted the test set. Rules were induced from the training set and the accuracy of the rule set was then determined on the corresponding test set. Each experiment was repeated ten times, once for each training and test set, and the average rule complexity, learning time, and classification accuracy are reported. The same ten training and test sets were used for all the experiments on a particular database.

6.1. An evaluation of BEXA's search restrictions

Four experiments were performed for each database to evaluate the individual search restrictions: (1) No restrictions to serve as a yard-stick for measuring improvements, (2) Employ only the prevent-empty-conjunctions restriction (Test 1), (3) Apply Test 1 and then the uncover-new-negatives restriction (Test 2), (4) Use all three restrictions in the order advocated in Section 4.3.3 and below. No results are reported for the cleveland database because it required an impractical amount of time to perform the ten experiments when no restriction was employed. The order in which the first two restrictions are applied is largely immaterial; both require very little overhead because the positive and negative extensions of a conjunction are computed in any case. We prefer Test 1 first though, because it is more efficient to test for non-zero than for new negatives uncovered, and because it is of no use to waste effort first on the more time consuming Tests 2 and 3 for a conjunction that will then be eliminated by Test 1. Section 4.3.3 explained that the uncover-new-negatives restriction should always be employed before the irredundancy restriction so that the latter, more time consuming restriction, is executed as few times as possible.

All the experiments were performed using a beam width of ten and no pre-pruning was performed, i.e. consistent conjunctions were generated with the significance and stop-growth thresholds set to 0. Section 5.2.1 showed that the specialization models of BEXA, AQ15 and CN2 are closely related, and if BEXA benefits from the search restrictions, it can be suspected that this may also be the case for AQ15 and CN2. The best results for CN2 were obtained with beam widths larger than one in all but two cases. Consequently, BEXA was executed with a beam width of ten to provide stronger evidence that AQ15 and CN2 may also benefit from its search restrictions.

When comparing complexity and accuracy of rules, post-pruned rules usually yield better quality rules than pre-pruned rules. For these experiments BEXA therefore did not post-prune its rules to isolate the effects of the search restrictions, reported in Figures 7 and 8.

Figure 5 gives the number of specializations considered with each combination of restrictions, expressed as a percentage of the number of specializations considered when no restriction was applied. The actual number of specializations that were evaluated is given above each bar. Each additional restriction reduced the number of specializations to evaluate for all nine databases. In some cases, the reduction was quite dramatic. For example, for the breast cancer database, the total number of specializations that were constructed dropped from 238815 to 12875 when all three the restrictions were employed. The small additional reduction caused by the irredundancy restriction is not surprising. The reason is

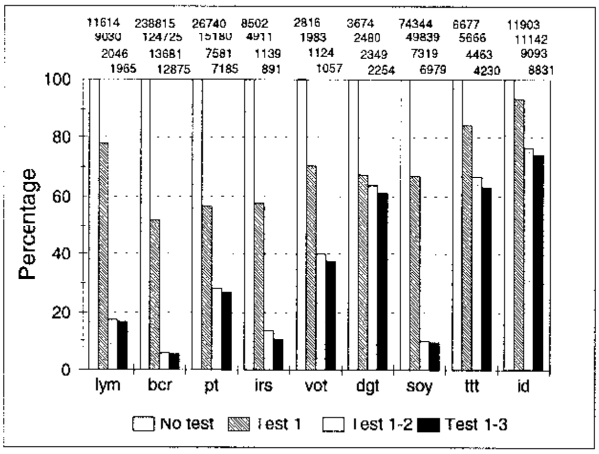

Figure 5. The total number of specializations constructed with each additional restriction

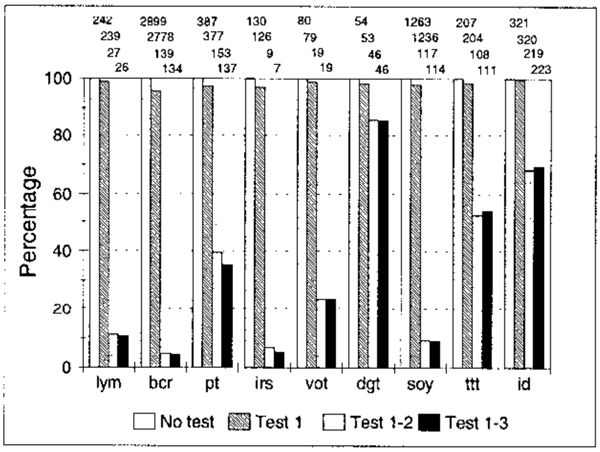

Figure 6. The learning times required with each additional restriction

that the uncover-new-negatives restriction also discards values that will lead to redundant conjunctions. Recall that this restriction is a necessary requirement for irredundancy. The irredundancy restriction therefore discards only the few remaining values and intervals that may lead to redundant conjunctions.

Figure 6 plots the learning time in seconds of each combination of restrictions as a percentage of the time required to learn the concept descriptions with no restrictions. The runtimes in seconds are given above each bar. For all the databases except tic-tac-toe and internal disjunction, each additional restriction reduced the learning time. For the tic-tac-toe and internal disjunction databases the learning time increased by 3% and 2% respectively with the introduction of the irredundancy restriction, despite a reduction in the number of specializations to evaluate. The reason is that these databases are fairly large and only a relatively small percentage of values can be discarded during the specialization process. Consequently, the irredundancy restriction had to be applied frequently. The overhead due to so many subset tests on large bitsets slightly exceeded the savings obtained by evaluating fewer specializations.

The prevent-empty-conjunctions restriction (Test 1) usually led to a large reduction in the number of specializations that were evaluated, but this reduction was not always accompanied by a similar reduction in the learning time. For example, for the lymphography database the number of specializations dropped by 22% from 11614 to 9030, while the learning time decreased by only 1% from 242 seconds to 239 seconds. The reason is that a conjunction covers the fewest positive instances towards the end of the specialization process. Consequently, the prevent-empty-conjunctions restriction discards most values and intervals from the `usable` set towards the end of the specialization process. It thus avoids only a small number of unnecessary evaluation steps. When the uncover-new-negatives restriction (Test 2) was applied, a reduction in the number of specializations was accompanied by a similar reduction in learning time. The reason is that values and intervals that exclude many negative instances are likely to be excluded from a conjunction right from the start of the specialization process. Consequently, a large number of negative instances are excluded after only a few specialization steps. The uncover-new-negatives restriction will therefore discard many values from the `usable` set after only a few specialization steps. None of these values will be excluded subsequently, resulting in the large drop in learning time when this restriction is activated.

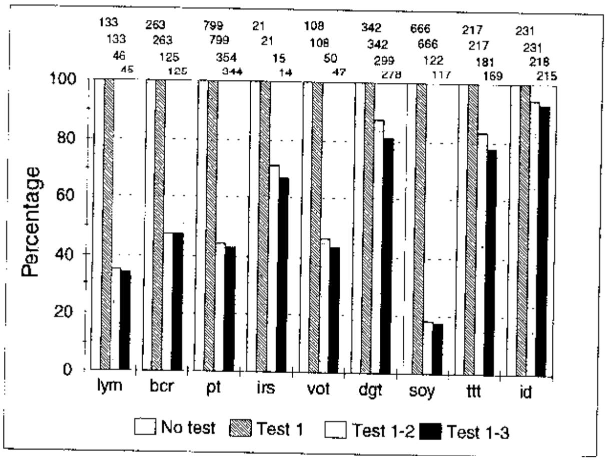

Figure 7. The description complexity achieved with each additional restriction

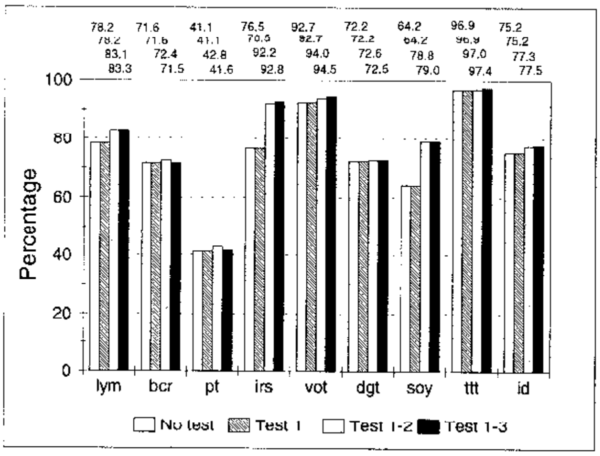

Figure 8. The description accuracy obtained with each additional restriction

Figure 7 expresses the complexity of the concept descriptions generated with each combination of restrictions as a percentage of the complexity of the descriptions generated when no restrictions were applied. The actual number of atoms obtained with each experiment is given above each bar. Figure 8 plots the accuracies obtained with each combination of restrictions for each of the databases. The prevent-empty-conjunctions restriction (Test 1) did not change the accuracy or complexity of any of the concept descriptions. The reason is that BEXA's Laplace estimate, and the evaluation functions of the other covering algorithms, has a higher value for a conjunction that covers at least one positive instance than one covering no positive instances. The prevent-empty-conjunctions restriction therefore simply prevents the evaluation of conjunctions that will in any case not be selected as the new best one.

The uncover-new-negatives restriction (Test 2) caused a large reduction in description complexity without reducing description accuracy. It even improved description accuracy in some cases. We explain this with an example, assuming two classes, a beam width of one, and that Test 2 is not applied. A conjunction's `usable` set consists of two disjoint subsets: Let $W0$ denote those elements that when excluded from the conjunction do not uncover any new negative instances, and let $W1$ denote those whose exclusion uncover new negatives (i.e. $W0$ will be deleted from `usable` by Test 2). Consider a conjunction w covering 30 positive and 2 negative instances (Laplace accuracy estimate 0.912). Assume that the best specialization of w by excluding a single value chosen from $W0$ leads to a conjunction excluding 2 positive instances (Laplace 0.906), while the best specialization from $W1$ excludes 12 positive and 1 negative instances (Laplace 0.905). Then the specialization due to $W0$ is preferred and thus retained for the next specialization step. Note that w's Laplace estimate cannot be improved unless a value from $W1$ is chosen and that w has a better estimate than any of its one-step specializations. Further specialization steps are thus required. These steps are useless when values are chosen from $W0$ because either they exclude positive instances only (which cannot improve on w's Laplace estimate), or they keep the extension of the specialized conjunction unchanged. This continuing process makes the conjunction more specific and may increase its complexity, until eventually a value from $W1$ is chosen, leading to a conjunction covering, say, 10 positive instances (Laplace 0.917). If, instead, Test 2 forced the choice of a value from $W1$ from the outset, a much more general conjunction might be found. The uncover-new-negatives restriction therefore leads to more general conjunctions, obtained by excluding fewer values and intervals from the most general conjunction. Such general conjunctions tend to cover more positive instances. This increases their accuracy estimate and reduces their complexity.

The irredundancy restriction (Test 3) yielded a slight additional reduction in description complexity and usually slightly improved accuracy or kept it unchanged. In two cases a reduction in accuracy not exceeding 1.2% was observed. The drop in accuracy was due to slightly more general conjunctions containing fewer atoms than those found without this restriction.

Finally, all the algorithms except BEXA restrict the number of specializations that are considered, and then leave it to the evaluation function to select the best specialization. The experiments in this section showed that the evaluation function alone cannot be trusted to find the best descriptions because the quality of the descriptions improved when the uncover-new-negatives and irredundancy restrictions were applied. The reason is that the evaluation function may select specializations that increase the complexity of a conjunction as was explained above. BEXA's search restrictions therefore "override" the short-sightedness of the evaluation function.

6.2. *BEXA versus CN2 and C4.5*

BEXA, CN2 and C4.5 each has a number of parameters whose values determine the quality of their induced concept descriptions. The parameter values that give the best results also differ from one database to another. In an attempt to find the best possible results for each algorithm, a number of different parameter combinations were evaluated for each database.

BEXA requires four different parameters. The first, the beam width, was kept at one as motivated in Section 5.2.3. The second parameter is a significance threshold. Recall that BEXA can employ the log-likelihood ratio criterion as a significance test, just like CN2. This criterion has a chi-square distribution, and thus requires the specification of a confidence level. For example, a confidence level of 90% may lead to fairly mild pruning, while a confidence level of 99.9% may lead to severe pruning. For all the experiments reported here for BEXA, the confidence level for the significance test was set to 0%, i.e. the significance test was effectively ignored. The reason is that Theron (1994) found that the stop-growth test usually produced similar or better results than CN2's significance test for most of the test databases. The third parameter is the confidence level for BEXA's stop-growth test. Recall from Section 4.1 that BEXA can also employ the log-likelihood ratio criterion as a stop-growth test. Usually, there is no way of knowing which threshold will produce the best results for a particular database. Consequently four stop-growth thresholds were evaluated for each database, namely 0% (no pruning), 90%, 99% and 99.9%. The last parameter specifies whether the unpruned rules (generated with stop-growth of 0%) must be post-pruned or not. Theron (1994) found that the stop-growth test produced better quality rules than Quinlan's (1987a) post-pruning technique for three of the test databases; post-pruning produced better rules for three other databases, while rules of similar quality were obtained by either technique for the remaining four databases. Both pruning and no pruning must thus be evaluated to find the best set of rules. A total of five experiments were thus performed for BEXA for each database, namely (1) generating unpruned rules, (2-4) generating rules with stop-growth thresholds of 90%, 99%, 99.9%, and (5) post-pruning the unpruned rules. All these experiments were performed with a beam width of one, and with a significance threshold of 0%. BEXA can easily be extended to perform these five experiments automatically and then return the best of the five sets of results. Thus employed, no parameters need to be specified for BEXA. These five experiments were not excessively time-consuming since it required a maximum of ninety seconds to perform (using a 33MHz 80486SX PC) and amounts to selecting an appropriate bias (from this limited set of possibilities) for a particular learning problem. Thus in spite of BEXA's greater computational complexity, its search restrictions limit the search effort so that its average case behavior is comparable to that of the other algorithms.

Since a number of experiments were performed for each database with each learning algorithm, the problem becomes to select the best results for that algorithm for comparison with the other algorithms. The best rules found with a particular algorithm for a given database was selected to be the simplest set of those sets whose accuracy did not differ significantly from the most accurate set found. The significance of differences in accuracy and complexity was determined using the two-tailed paired t-test to compare the results for the ten random training and test sets that were selected for each database. The best results obtained with each algorithm were then compared as follows. If the significance of the difference in description accuracy was 95% or higher, the most accurate rule set was the best. Otherwise the accuracies were deemed equivalent and description complexity was compared similarly. The most common measure for description complexity is the total number of atoms in a concept description (Michalski *et al.*, 1986; Clark & Niblett, 1989; Clark & Boswell, 1991; Lavrac *et al.*, 1986; Pagallo & Haussler, 1990), but the number of values or intervals mentioned in a rule is given as well for databases with nominal attributes taking more than three values. (Since BEXA writes out the complement of an atom if the atom contains more than half of its domain, counting values instead of atoms only influences the count for these attributes.)

6.2.1. *BEXA versus CN2*

There are a number of parameters that determine the quality of the rules generated by CN2. Firstly, CN2 can generate either ordered or unordered rules. Unordered rules were generated for all the databases because Clark & Boswell (1991) found that unordered rules have higher accuracy than ordered rules and because BEXA generates only unordered rules. Secondly, CN2 can use either the Laplacian or the naive (plain percentage correct) accuracy estimate. The Laplace accuracy estimate was selected because Clark & Boswell (1991) used it in their experiments. The Laplace estimate is also better than the naive estimate because it will

Table 8. BEXA versus CN2

%Correct	lym	bcr	pt	irs	vot	dgt	soy	ttt	id	cle
BEXA	80.2	76.9	42.3	93.9	**95.3**	73.2	79.5	97.1	**82.0**	78.2
CN2	79.6	74.3	39.8	92.6	94.1	72.3	82.6	**98.9**	75.5	77.5
T-test	19%	89%	89%	63%	97%	82%	75%	98%	100%	39%
#Atoms (#Values)	lym	bcr	pt	irs	vot	dgt	soy	ttt	id	cle
BEXA	**7**(8)	2(2)	**11**	**3**	**3**	**41**	**59**(60)	48	**6**(11)	**12**(12)
CN2	27	3	117	5	22	208	84	68	262	31
T-test	100%	90%	100%	100%	100%	100%	100%	100%	100%	100%
Parameters that gave the best results	lym	bcr	pt	irs	vot	dgt	soy	ttt	id	cle
BEXA:										
Post-pr.	√	√	√	√		√		√		√
Stop-gr.	99%	99.9%	99%	99.9%	99.9%	90%	90%		90%	
CN2:										
Beam W	10	1	20	10	10	10	20	20	1	20
Signif.	90%	99.9%	99.9%	99%	99%	90%	0%	0%	0%	99%

prefer a conjunction covering twenty positive and no negative instances to one covering two positive and no negative instances, while the naive estimate has the same value (100%) for both. The remaining two important parameters were the beam width and the significance threshold. CN2 does not employ post-pruning or a stop-growth test. For the experiments reported in Clark & Boswell (1991), CN2's beam width was set to twenty. BEXA's results suggest that CN2 may also give good results with small beam widths. Consequently, three beam widths were evaluated, namely one, ten and twenty. Clark & Boswell (1991) also evaluated only two significance thresholds for CN2, namely 0% (no pruning) and 99.5%. We evaluated four thresholds, namely 0%, 90%, 99% and 99.9% for each of the three beam widths, giving a total of twelve experiments for each database. The best of these twelve rule sets were selected as described in the previous section.

Table 8 contains the best results obtained with CN2's twelve experiments and with BEXA's five experiments. Results in bold indicate significant differences. For two databases, voting and internal disjunction, BEXA's rules were significantly more accurate than those of CN2. For six other databases where accuracy differences were not significant BEXA produced significantly simpler rules than CN2. CN2 produced the best results for the tic-tac-toe database where its rules were significantly more accurate. To determine the reasons for differences in performance, the differences between BEXA and CN2 are briefly noted. BEXA allows (1) a richer description language (internal disjunction and the complement ("not") of an atom) implemented by excluding values instead of appending atoms, (2) optional search restrictions (Tests 1-3), (3) a stop-growth test, and (4) post-pruning of rules. Other minor differences affect the efficiency of BEXA and not its performance, as explained in Section 4.2. When the voting database is considered, the only causes for performance differences are Tests 1-3 and the stop-growth test, since voting contains only binary attributes and the best results were obtained without post-pruning. We therefore performed four additional experiments (for the same stop-growth threshold of 99.9%) in which Tests 1-3 were first omitted and then included one by one, and in all cases the same rule set was found. When relaxing the stop-growth threshold to 99%, description complexity increased to nine and accuracy dropped by 0.6%, illustrating that the stop-growth test terminated specialization early enough.

The id database contains only nominal attributes, each taking five values. In this case the possible causes for performance differences are BEXA's specialization method and accompanying richer description language, Tests 1-3 and the stop growth test. For the best results given in Table 8 BEXA found the exact concept description. When repeating the experiment with a 95% stop-growth threshold, pre-pruning was too strict; accuracy dropped to 78.1% with only three atoms in the description. At the 90% stop growth threshold, several experiments were performed with combinations of Tests 1-3. Test 1 had no effect on the generated descriptions; Tests 2 and 3 did not affect the accuracy, but performing only Test 2 caused, for the first time, a description containing the irrelevant attribute E (i.e. a more complex description than necessary). For the id database CN2's restriction to pure descriptions does not allow such a succinct description of the target concept as is possible with descriptions containing internal disjunction, but there is in principle no reason why CN2 cannot generate the equivalent 15 pure rules describing the target concept. However, CN2 generated very specific conjunctions usually containing three atoms and covering one to three examples. BEXA's specialization method of excluding one value at a time biases it to find descriptions such as [X $\in$ {1,2,3}] more easily than CN2, since CN2 must construct three rules, one for each value of X. In the presence of noise where attributes other than X are contenders for selection, CN2 has no bias to select attribute X again to construct a subsequent rule (i.e. to lead to three pure conjunctions equivalent to [X $\in$ {1,2,3}]); instead other attributes may be taken. In this way CN2's language restriction may break up a natural concept description into an accurate and highly specific, but difficult to understand, description (this is also one of Gray's (1990) criticisms of decision trees). For the id database, therefore, BEXA's specialization method biased it to prefer most general descriptions, Tests 2 and 3 prevented specialization of irrelevant attributes (causing them to be eliminated), and its stop-growth test terminated specialization before the description fits the noise.

CN2's better results for tic-tac-toe are not surprising since this database has inherently pure concept descriptions (e.g. [top-left-square = 'X'][top-middle-square = 'X'][top right square = 'X'] or ...). BEXA thus over-generalized for this database where its accuracy was 1.8% less. The internal disjunction database on the other hand illustrates the advantages of being able to generate internally disjunctive expressions where BEXA's rules were markedly more accurate.

6.2.2. *BEXA versus C4.5*

BEXA was compared to C4.5 to present a yard-stick for rule quality, not with the intention of comparing their specialization methods. C4.5 has a facility to generate a set of pruned production rules from a decision tree. Quinlan (1987a,1987b) found that these rules are much simpler and also more accurate than pruned decision trees. We thus compared BEXA's results to the best rules generated by C4.5. The tree pruning defaults were used because

Table 9. BEXA versus C4.5

				%Correct						
	lym	bcr	pt	irs	vot	dgt	soy	ttt	id	cle
BEXA	80.2	76.9	**42.3**	93.9	95.3	73.2	79.5	97.1	82.0	78.2
C4.5	77.1	74.9	39.3	94.0	95.1	72.9	85.0	98.3	83.3	75.9
Signif. (T-test)	74%	80%	97%	1%	37%	53%	87%	76%	69%	81%
				#Atoms						
	lym	bcr	pt	irs	vot	dgt	soy	ttt	id	cle
BEXA	**7**	**2**	**11**	**3**	**3**	**41**	59	48	**6**	12
C4.5	13	7	23	5	7	79	**44**	**35**	8	12
Signif. (T-test)	100%	100%	100%	100%	99%	100%	100%	100%	100%	0%
				Parameters that gave the best results						
	lym	bcr	pt	irs	vot	dgt	soy	ttt	id	cle
BEXA:										
Post-pr. rules	√	√	√	√		√		√		√
Stop-gr. Thresh					99.9%		90%		90%	
C4.5:										
Subsets (-s)	×	√	√	×	×	×	√	×	√	√
Confidence (-c)	1%	25%	25%	40%	1%	1%	1%	25%	1%	1%
Redundancy (-r)	2	0.5	0.5	0.1	0.5	2	0.5	0.1	0.1	0.5

C4.5 always generates rules from unpruned trees. Pruning the trees therefore would have no influence on the quality of the generated rules. No windowing was required since all the datasets were small enough to fit into memory. Furthermore, Wirth & Catlett (1988) found that windowing rarely improves the quality of decision trees. The initial trees were generated using the default gain ratio criterion as the evaluation function because Quinlan (1986) found it to be superior to the older gain criterion. Secondly, C4.5 can combine branches in a tree so that some branches are labelled with subsets of values instead of single values, providing the option of internally disjunctive rules. Trees were generating using both settings of this parameter. Three parameters control C4.5's rule pruning process. The *confidence level* is used to determine which atoms to drop from a rule during the pruning process (default 25%). A lower confidence level leads to more severe pruning and a higher confidence level prunes less severely. We evaluated confidence levels of 40%, 25%, 5% and 1%. The second parameter specifies the confidence level that must be used if the older Fisher's exact test must also be used to prune rules. Theron (1994) found that this test makes no difference, i.e. the primary pruning test for which the confidence level must be specified succeeds in weeding out all insignificant atoms. The last rule pruning parameter is a *redundancy factor* that determines which rules must be retained when the collection of rules is simplified by deleting complete rules. The default redundancy factor is 1.0. Lower values lead to more severe pruning, and higher values lead to less severe pruning. We evaluated redundancy factors of 2.0, 1.0, 0.5 and 0.1. A total of 32 experiments were thus performed for each database to evaluate all possible combinations of the four confidence thresholds and redundancy factors, both with and without the subset construction feature during tree generation.

Table 9 contains the best results obtained with BEXA and C4.5 for each database in the same format as that of Table 8. BEXA produced significantly more accurate rules (3%)

than C4.5 for the pt database. C4.5 produced the simplest rules for only two databases (soybean and tic-tac-toe); both algorithms produced equivalent rules for cle, while BEXA had significantly simpler rules for the remaining six databases. C4.5 generated simpler rules than BEXA for tic-tac-toe for the same reason as CN2: both generated pure descriptions (C4.5 gave the best results when the subset construction parameter was not set).

For soy both algorithms generated internally disjunctive rules; BEXA's rules were not post-pruned, instead pre-pruning with the application of Tests 1-3 produced the best results. This database contains 17 binary and 18 other nominal attributes with three or more values per domain. When comparing the soy results, we note that BEXA's rules were more general than C4.5's; BEXA used the "not" operator which caused many atoms to cover a larger subset of the domain than C4.5's atoms. BEXA's rules had 60 values, compared to the 55 of C4.5; thus having reasonably similar complexity. For id, containing only non-binary nominal attributes, similar results were obtained with identical parameter settings for BEXA (the only difference for C4.5 compared to the soy experiment was the redundancy factor). Here BEXA had slightly less complex rules than C4.5. As illustrated by the additional experiments for BEXA (when compared to CN2 on the voting database in Section 6.2.1), Tests 2 and 3 weeded out irrelevant attribute values, while the stop-growth test prevented overfitting.

When comparing the soy and id experiments to pt, we note that pt contains 14 binary and three other nominal attributes (the linear attribute is converted to nominal with three intervals; thus each attribute takes three values). With the "not" operator, the number of values and atoms for BEXA's rules is therefore the same. When the "not" operator is not taken into account, BEXA generated internally disjunctive rules for four of the ten experiments which produced the best results, while C4.5's rules were pure (even though the subset option was used). BEXA's stop-growth test was not used. When Tests 2 and 3 were not used the rules before post-pruning contained 15 more atoms. These tests, therefore, caused BEXA to generate maximally general rules, making it possible for the post-pruning process to spot the really informative atoms because it evaluates the rule-set as a whole.

The main reason for BEXA's success is its ability to generate very simple rules. BEXA's richer description language, specialization method and search restrictions prefer more general descriptions, while its stop-growth test prevents overfitting. This bias towards generality can be adjusted by suitable settings of its parameters.

7. Summary

This paper presented a uniform framework of excluding of values, instead of appending atoms, for specialization models employed by AQ15, CN2, GREEDY3, PRISM, and GALG. This view of the specialization process led to BEXA, which has two novel features. Firstly, it specializes a conjunction by explicitly excluding single values or intervals instead of appending atoms to it as is done by the other covering algorithms. Secondly, BEXA does not restrict the number and type of specializations that are constructed for a conjunction at each specialization step. However, it keeps the search effort manageable by employing three dynamic search restrictions that avoid the construction of useless specializations. The key search restriction is the irredundancy test and its special case, the uncover new negatives test, which biases BEXA towards general conjunctions. These conjunctions contain few atoms and have high values for the evaluation function because they cover many positive instances. We also provided a theoretical foundation for BEXA's search constraints and compared it empirically to CN2 and C4.5. BEXA generates concept descriptions of comparable accuracy, but with greater simplicity, than these well-known state-of-the-art concept learners.

Acknowledgments

Peter Clark, Robin Boswell, and N.A.B. Gray sent us copies of their articles and reports for which we are grateful. We also thank the anonymous reviewers and the editor, Paul Utgoff, whose valuable and detailed comments greatly improved the clarity of this paper. The breast cancer, iris and digit databases were supplied by John Mingers. The other databases were provided by Patrick Murphy at the UCI Repository of Machine Learning Databases and Domain theories (ml-repository@ics.uci.edu).

Notes

[1] Top Down Induction of Decision Trees

[2] This discussion is based on the code of CN2, version 5.1, that was obtained from the Turing Institute.

[3] AQ15 was recently modified to employ post-pruning that simplifies individual rules. The new system is called POSEIDON (Bergadano *et al.*, 1992).

References

Bergadano, F., Matwin, S., Michalski, R.S. & Zhang, J. (1992). Learning two-tiered descriptions of flexible concepts: The POSEIDON system. *Machine Learning*, *8*, 5 - 43.

Breiman, L., Friedman, J.H., Olshen, R.A., & Stone, C.J. (1984). *Classification and regression trees*. Belmont: Wadsworth.

Buntine, W. & Niblett, T. (1992). A further comparison of splitting rules for decision-tree induction. *Machine Learning*, *8*, 75 - 85.

Cendrowska, J. (1987). PRISM: An algorithm for inducing modular rules. *International Journal of Man-Machine Studies*, *27*, 349 - 370.

Clark, P. & Niblett, T. (1989). The CN2 induction algorithm. *Machine Learning*, *3*, 261 - 283.

Clark, P. & Boswell, R. (1991). Rule induction with CN2: Some recent improvements. In Y. Kodratoff (Ed.), *Machine Learning – European Working Session on Learning EWSL–91*, (pp. 151 - 163). Berlin: Springer-Verlag.

Fayyad, U.M. & Irani, K.B. (1992). On the handling of continuous-valued attributes in decision tree generation. *Machine Learning*, *8*, 87 - 102.

Gray, N.A.B. (1988). *Why grow trees?*, Technical Report, University of Wollongong, N.S.W., Australia.

Gray, N.A.B. (1990). Capturing knowledge through top-down induction of decision trees. *IEEE Expert*, June, 41 - 50.

Haussler, D. (1988). Quantifying inductive bias: AI learning algorithms and Valiant's learning framework. *Artificial Intelligence*, *36*, 177 - 221.

Hoff, W.A., Michalski, R.S. & Stepp, R.E. (1983). INDUCE 2: A program for learning structural descriptions from examples. Report, University of Illinois at Urbana-Champaign.

Lavrac, N., Mozetic, I, & Kononenko, I. (1986). An experimental comparison of two learning programs in three medical domains. *Proceedings of the ISSEK workshop*, Turing Institute, Glasgow.

Lindsay, R.K., Buchanan, B.G., Feigenbaum, E.A., & Lederberg, J. (1980). *Applications of artificial intelligence for organic chemistry: The DENDRAL project.* New York: McGraw-Hill.

Michalski, R.S. (1975). Variable-valued logic and its applications to pattern recognition and machine learning. In D.C. Rine (Ed.), *Computer Science and Multiple-valued logic: Theory and applications*, North Holland, 506 - 534.

Michalski, R.S. & Chilauski, R.L. (1980). Learning by being told and learning from examples: An experimental comparison of the two methods of knowledge acquisition in the context of developing and expert system for soybean disease diagnosis. *International Journal of Policy Analysis and Information Systems, 4*, 125 - 161.

Michalski, R.S. (1983). A theory and methodology of inductive learning. In R. S. Michalski, J. G. Carbonell, & T. M. Mitchell (Eds.), *Machine learning: An artificial intelligence approach*. Los Altos, CA: Morgan Kaufmann.

Michalski, R.S., & Stepp, R.E. (1983). Learning from observation: Conceptual clustering. In R. S. Michalski, J. G. Carbonell, & T. M. Mitchell (Eds.). *Machine learning: An artificial intelligence approach*. Los Altos, CA: Morgan Kaufmann.

Michalski, R.S., Mozetic, I., Hong, J., & Lavrac, N. (1986). The multi-purpose incremental learning system AQ15 and its testing application to three medical domains. *Proceedings of the American association of artificial intelligence* (pp. 1041 - 1045). Los Altos, CA: Morgan Kaufmann.

Mingers, J. (1989). An empirical comparison of selection measures for decision-tree induction. *Machine Learning, 3*, 319 - 342.

Mitchell, T. M. (1982). Generalization a Search. *Artificial Intelligence, 18*, 203 - 226.

Pagallo, G. & Haussler, D. (1990). Boolean feature discovery in empirical learning. *Machine Learning, 5*, 71 - 99.

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

Quinlan, J.R. (1987a). Simplifying decision trees. *International Journal of Man-Machine Studies, 27*, 221 - 234.

Quinlan, J.R. (1987b). Generating production rules from decision trees. *International Joint Conference on Artificial Intelligence*, 304 - 307.

Rymon, R. (1993). An SE-tree based characterization of the induction problem. *10th International Conference on Machine Learning*, 268 - 275.

Schaffer, C. (1993). Overfitting avoidance as bias. *Machine Learning, 10*, Boston: Kluwer Academic Publishers, 153 - 178.

Theron, H. & Cloete, I. (1993). An empirical evaluation of beam search and pruning in BEXA, in *Proceedings of the Fifth International IEEE Conference on Tools for Artificial Intelligence – TAI'93*, Cambridge, Massachusettes, 8 - 11 November.

Theron, H. (1994). *Specialization by exclusion: An approach to concept learning.* Ph.D. dissertation, Department of Computer Science, University of Stellenbosch, Stellenbosch, South Africa, March 1994.

Wells, M.B. (1971). *Elements of combinatorial computing.* New York: Pergamon Press.

Wirth,. J. & Catlett, J. (1988). Experiments on the costs and benefits of windowing in ID3. *Proceedings of the 5th International Workshop on Machine Learning*. Los Altos, CA: Morgan Kaufmann, 87 - 99.

Received February 4, 1994
Accepted August 2, 1995
Final Manuscript September 7, 1995