89-1

Characterizing Machine Learning Programs: *A European Compilation*

Y. Kodratoff

MLI 89-14
81-1

CHARACTERIZING MACHINE LEARNING PROGRAMS
A European Compilation

Yves Kodratoff

CNRS & Université Paris Sud, LRI, Bldg 490, 91405 Orsay, France
George Mason University, AI Center, Fairfax, Virginia 22030, USA

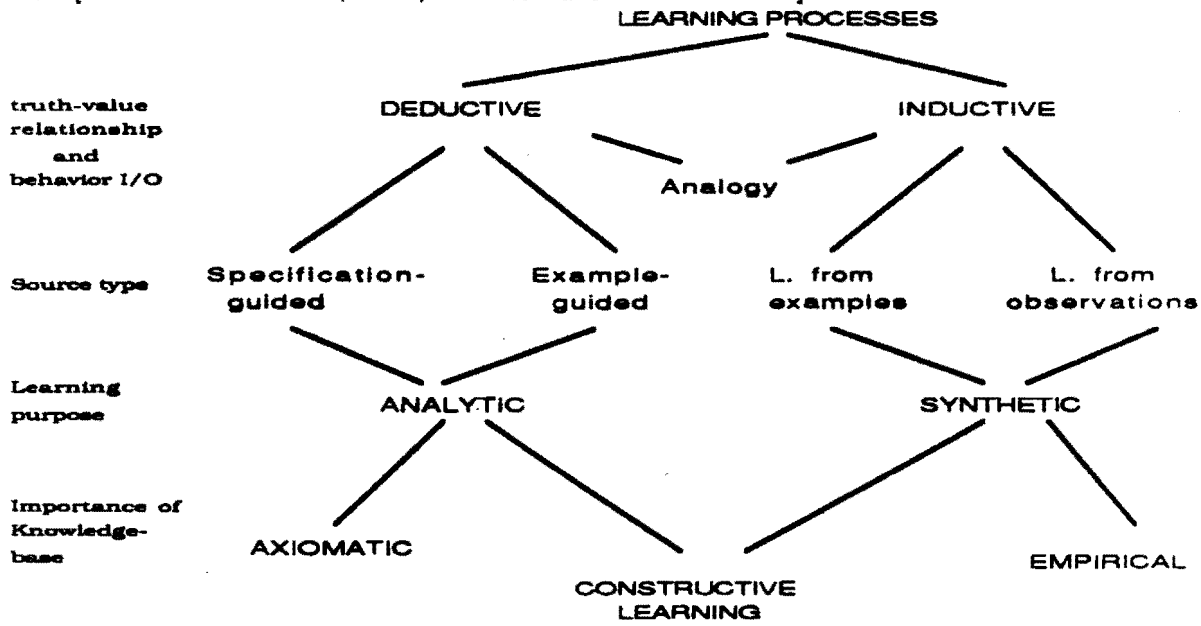
Abstract

The paper describes a number of possible characteristic features of Machine Learning (ML) programs. They have been willingly chosen to mark the Artificial Intelligence (AI) approach to ML, as illustrated for instance in [Michalski et al. 1983, 1986, Kodratoff 1988]. It follows that some of the systems that perform learning essentially by adjusting coefficients may find it difficult to fit into our schemes. On the contrary, all AI oriented programs should fit in, unless we have made mistakes with our description.
Artificial as it looks, this effort of characterization addresses two deep concerns. The first one is a clarification of the terms used, and sometimes abused, in the ML community. The second is to promote communication in the sub-field. In most cases, learning systems do learn, but it is neither clear what nor how they learn. Setting bench marks for ML is a necessity the community should be more anxious about.
Some of the systems known as knowledge acquisition systems perform as well some learning. The last section is devoted to this kind of programs.

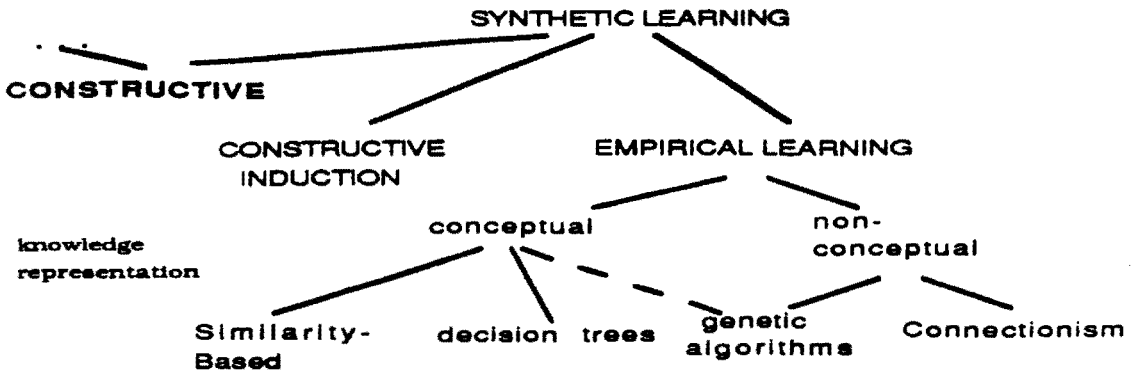
1 - INTRODUCTION

This introduction will give an overall description of the topics in ML research, and a description of what is the AI approach to ML.

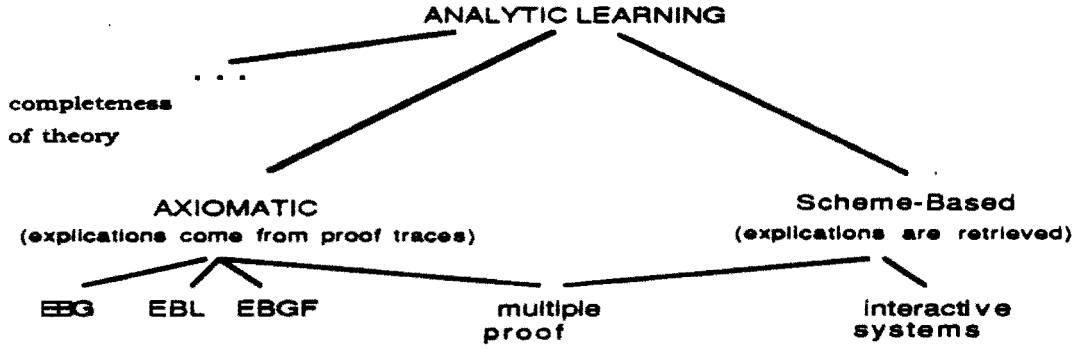
Here is a description of ML that comes from the introduction of the third volume in the series *ML: An AI Approach* [Michalski et al. 1983, 1986, Kodratoff and Michalski 1989].



Michalski's general description of ML



Details on synthetic learning



Details on analytic learning

This taxonomy gives a fairly accurate description of the different ML topics, and of their interrelationships. It shows a striking symmetry between deductive and inductive learning. Most of the early research has been done about inductive learning, in the USA as well as in Europe. The present symmetry reflects the so-called "Explanation-Based learning (EBL) revolution" that took place during the last few years mainly in the USA, under the influence of Jerry DeJong and Tom Mitchell. By consulting the tables in which the European systems are described, one notices that EBL has been catching in Europe as well.

This taxonomy is quite self-explanatory, if the terms used in it are not ambiguous. The glossary found in appendix will attempt to make them precise. Nevertheless, there are still discussions about their exact meaning, and section 3 will discuss some of the most important choices that have been made in this paper.

Section 6 will provide some information about a new field known as Knowledge Acquisition (KA), defined in the glossary. We felt it necessary to include in this compilation those of the KA systems that use ML because they are actually so few of them, and because this tendency is well represented in Europe.

1 - INTRODUCTION

The AI approach to ML relies on several concepts (induction, abduction, generalization) that are often misunderstood. We shall therefore first give a precise definition of them.

1.1 - Induction, deduction

Let us first as briefly as possible recall what induction and deduction are, more details are given in [Kodratoff 1988].

Deduction is the process by which one **infers while preserving the truth** of the data already stored. Applying the well-known modus ponens rule is truth-preserving. For instance, suppose that a theorem tells us that all green leaves are young leaves. While meeting green leaves, I can deduce that they are young.
Induction is the process by which one **infers while preserving the falsity** of the data already known. Michalski generalization rules [Michalski 1984] are falsity preserving. For instance, while meeting green leaves I can induce that I am meeting green things (that is, forgetting that they are leaves), or as well, I can induce that I am meeting young things, by using the above theorem. Suppose also that an other theorem tells us that anything green is not red, then, conversely, I cannot induce that I am meeting red things since "green" implies "not-red", that is, red = FALSE, and I must preserve this falsity during my induction.

1.2 - Knowledge Intensive Induction

Many authors link deductive learning with knowledge intensive approaches (which they actually are in existing implementations), and inductive learning with knowledge poor approaches (there exist implementations of induction that are knowledge intensive as well, for instance [Michalski 1983, Kodratoff et al. 1984, Kodratoff & Ganascia 1986]). It is clear that "pure induction" is knowledge poor. The point is that pure induction would mean induction from raw data, and raw data do not exist in reality since the world representation we get in the machine, as raw as it can be, is bound to already bear much interpretation. A clear definition of abduction will help us to define induction better, and to show the role of background knowledge in the global inductive process.

1.2.1 - What is abduction?

Technically, abduction is the process which "reverses" the implication symbol, i.e. the process by which, knowing that A fi B, one infers A, knowing that B is true. Of course, this is not truth preserving since A fi B tells that one can infer B from the knowledge of the truth of A, as classical modus ponens does. In practice, this amounts to the process by which we are able to complete a proof that just failed, by doing hypotheses that, if they were valid, would indeed lead the proof to a success.
Let us illustrate abduction by the following example drawn from [DeJong and Mooney 1986]. In this example, our aim is learning a definition of the concept of suicide Kill(x,x). Suppose then that we have been learning so far that suicide requires a weapon, as expressed by the rule

KILL(x,x) :- DEPRESSED(x), BUY(x,y), WEAPON(y).

Suppose now that we know of Mary who commits suicide. Suppose that we have the following knowledge about her.
DEPRESSED(MARY).
BUY(MARY,OBJ1).
SLEEPING-PILLS(OBJ1).
PRICE(SLEEPING-PILLS, 20).
BUY(MARY, OBJ2).
BOOK(OBJ2).
. . .
where OBJ1 and OBJ2 are variables.
We will be unable to prove KILL(MARY, MARY) because she has no weapon. Cox [Cox 1986] presents an algorithm that, in such a situation where WEAPON(y) fails, would automatically look for "weapons" and catch the first objects that have been satisfactory so far (here, those that have been satisfying BUY(x,y)), therefore abduct WEAPON(SLEEPING-PILLS) or WEAPON(BOOK) depending on which is met first. In the absence of any other kind of knowledge, it is clear that anything that has been bought might be taken as a weapon. We [Duval and Kodratoff 1989] support the fact that another abduction is also possible, namely by adding new rules in which the objects that have been satisfying BUY(x,y) are added into the clause, without hypothesizing that they are "weapons". Here, one could abduct as well

KILL(x,x) :- DEPRESSED(x), BUY(x,y), SLEEPING-PILLS(y)

or

KILL(x,x) :- DEPRESSED(x), BUY(x,y), BOOK(y).

All this shows that even in very constrained environments, many abductions are possible, all of them solving the problem of completing the failed proof. Choosing which is the best one depends clearly on the background knowledge at hand. This is why we claim that, even though an isolated abduction is knowledge poor, the whole abductive process is very much knowledge intensive.

1.3 - Generalization-particularization/bottom up-top down generalization

Tom Mitchell's version spaces [Mitchell 1982, Genesereth and Nilsson 1985, Kodratoff 1988] maintain the generalization state of an operator. They give the exact generalization state in which a descriptor used by an operator must be kept in order to optimize the problem solving efficiency. Given a set of positive and negative examples, their "version space" is the set of the consistent formulas, i.e. the set of formulas that are both complete (they recognize all the positive examples) and coherent (they recognize none of the negative examples).

When building the version space of an operator, positive examples are used "bottom-up", that is to say, by finding a formula which is more general than any of the examples, and such that each example is an instantiation of this formula (also, none of the negative examples should be an instantiation of it, but this not the point here). Conversely, the negative examples are mainly used "top-down", since they are used to particularize a formula which covers none of the negative examples (also, all the positive examples should be an instantiation of it, but this is not the point here). This should convince people that generalization (from positive examples) and particularization (from negative examples) are the two facets of the same process, known under the name of "generalization" because earlier works did not recognize enough the importance of negative examples, and the symmetry between positive and negative examples [Nicolas 1988].

There are other approaches than those inspired by the version spaces. Michalski's [Michalski and Stepp 1983, Michalski 1984] work is typical of those that progressively generalize from examples (his algorithms perform bottom up generalization). These algorithms cluster examples together, and simultaneously find a function that recognizes these clustered examples and reject all counter-examples. The progressive growth of these clusters generates more and more general recognition functions. On the contrary, all the ID3 family inspired from Quinlan [Quinlan 1983] start from a set that contains all the examples, and progressively applies descriptors that cut it into sub-sets, according to the value of the descriptor. For instance, if we start with a set containing persons with all types of eye colors, we shall form as many sub-sets as they are colors of the eyes, each containing one color. Therefore, these algorithms progressively particularize, they perform top-down generalization.

Top down or bottom up, generalization is always an inductive process. It is therefore very wrong to confuse particularization and deduction. To make this precise, let us describe in a similar way generalization and particularization. When generalizing (= bottom up generalization), the problem is inducing the common features existing between several examples. These common features characterize the concept they are all instance of. When particularizing (= top down generalization), the problem is inducing the features that make the difference between several examples. Each example is characterized by the features that are not shown by the others.

1.4 - What is the AI approach to ML

We present here five features characterizing AI, that make the AI approach to ML original. When they are used in a very strict way, that is to say when all of them are requested to be fulfilled, one defines something like the core of AI. A softer definition of AI, that will include all the existing AI systems, will be to request that some of the features given below are partly fulfilled.

First, as opposed to the "intelligent machine" definition [Barr and Feigenbaum 1981], AI refuses to built intelligent black boxes. For instance, Roger Schank points out [Schank 1986] that the chess playing machines presently available on the market are out of AI (actually, they have been built without using any AI

techniques) even though they start being real good players. The reasons why they play so well are available to their designers only, not to their users, and therefore they cannot be considered as AI. Anyhow, it would be somewhat strange to attribute to AI the success of these chess machines since they result from operational research, not from AI. We do not claim here that temporary stoppage of communication is impossible in AI, we rather point out that a definitive loss of any communication abilities is contrary to the spirit of AI. AI systems are open to their users who must understand them.

Second, AI tends to avoid using implicit knowledge. A clear illustration of this assertion is the marked preference AI shows for declarative knowledge representation, in which knowledge and the way to use it are clearly differentiated. AI tends to avoid procedural representations where knowledge and its use are merged. For instance, AI favored the birth of rather strongly declarative languages like LISP and PROLOG. Another well-known example of ES. An ES is nothing but a kind of decision tree, except that the tree has been flattened in a declarative form by writing out each rule that lead to a decision, without telling in advance in which order they must be called. This generates the so-called "conflict resolution" problem, which is the price that must be paid to declarativity. In an AI system, knowledge should be stated explicitly, preferably in a declarative way. As little as possible procedural knowledge is expressed in a programming language. Similarly, fancy ad hoc codings are very much in opposition with the spirit of AI.

Third, as opposed to what the early workers believed, a system cannot pass the above tests without already carrying a huge amount of preliminary knowledge. At least, this knowledge must carry the user's understanding of the problem. This is illustrated by the importance given to knowledge bases in AI. Knowledge is a kind of data: We do not speak of data bases because our bases have to be declarative and understandable by their users. An AI system is supposed to make heavy use of background knowledge, provided by the user in a knowledge base.

Fourth, in accordance with the need for user understandability, an AI system has to speak its user's language, instead of speaking a "language of AI", that, by the way, does not exist. This feature opposes well AI to other sciences the first concern of which is the creation of their own jargon, to be imposed on their user as part of his/her scientific formation. For instance, statistics does describe interesting natural phenomena, and does it in terms of "mean squares distances" etc ..., which is its own language, instead of expressing the knowledge in its user's language, as AI attempts (and still partly fails) to do.

Fifth, we believe (even if this is far from being accepted by all our colleagues) that AI is concerned with the definition, analysis, measurement, and comparison of explanations relative to an automatically executed reasoning session, and provided in the user's language. Our main argument follows from the recognition that explicability fulfills the needs expressed above since the concern for understandability is perfectly met by a system able to provide itself its user with explanations. Moreover, a huge amount of background knowledge is necessary to be able to find relevant explanations.

2 - EXPLANATION-BASED LEARNING (EBL) IN STRONG THEORY DOMAINS

The most rigorous EBL learns generalizations only, this is why it has been called Explanation-Based Generalization (EBG).

2.1 - EBG: An intuitive presentation

EBG is given a complete theory of the domain in which the learning takes place and a training instance of the concept to be learned. Since the system knows the complete theory, it knows a definition of the concept to be learned. The goal of the learning process is to learn a new "better" definition of this concept. To achieve this goal, EBG works as follows. Using its knowledge of the domain theory, it proves that the training instance is actually an instance of the concept learned. The proof trace is then pruned by using an operationality criterion that tells which descriptors must be left out. This amounts to telling which part of the proof is considered as being of interest. Notice the importance of this criterion on which relies all the efficiency of the process. The pruned proof trace is then generalized into an explanation structure, which contains the parts of the original clauses that have been used. As noticed in [DeJong and Mooney 1986], this amounts to keeping a part of the unifications needed to achieve the proof. A variablized version of the training instance

is then regressed [Waldinger 1977] through the explanation structure. The result of this regression is new, more operational definition of the concept.

In order to illustrate EBG, we shall use here a PROLOG version of the "SAFE-TO-STACK" example, taken from [Mitchell and al. 1986]. Several versions of EBG have been implemented as for instance in [Kedar-Cabelli and McCarty 1987, Puget 1987, Siqueira and Puget 1988]. The 'SAFE-TO-STACK example shows how to learn a more efficient rule to stack given - a definition a theory of stacking - and an example of a particular box 'BOX1' stacked on a particular table 'ENDTABLE1'. In this very particular example it happens that one needs to know the default value of the type 'TABLE'.
The first kind of information is the one relative to the specific box, 'BOX1', and the specific table 'ENDTABLE1'.

```
C1    ON(BOX1, ENDTABLE1).
C2    COLOR(BOX1, RED).
C3    COLOR(ENDTABLE1, BLUE).
C4    VOLUME(BOX1, 10).
C5    DENSITY(BOX1, 1).
C6    FRAGILE(ENDTABLE1).
C7    OWNER(ENDTABLE1, CLYDE).
C8    OWNER(BOX1, BONNIE).
C9    ISA(ENDTABLE1, TABLE).
```

The second kind of information is the one relative to the background knowledge about stacking things, also called the domain theory.

```
C10   SAFE-TO-STACK(x, y)   :-   NOT FRAGILE(y)
C11   SAFE-TO-STACK(x, y)   :-   LIGHTER(x, y)
C12   WEIGHT(x, w)          :-   VOLUME(x, v), DENSITY(x, d), w is v*d
C13   WEIGHT(TABLE, 50)     :-
C14   LIGHTER(x, y)         :-   WEIGHT(x, w1), WEIGHT(y, w2), LESS(w1, w2)
C15   LESS(x, y)           :-   x < y
```

The third kind of information is the one expressing that one can safely stack 'BOX1' on 'ENDTABLE1'. Since we want to prove that by a PROLOG refutation procedure, this knowledge will be given as question to the PROLOG interpreter. It reads

```
C16                        :-   SAFE-TO-STACK(BOX1, ENDTABLE1)
```

The proof proceeds as shown by the following trace. This trace is provided by most PROLOG interpreters, though in less readable form.

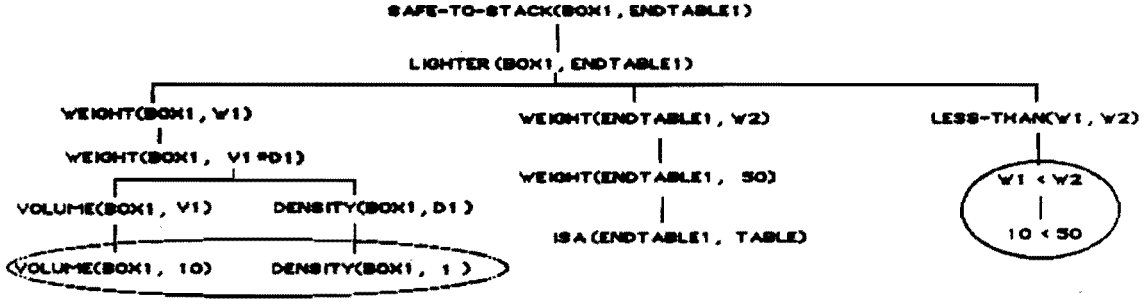


Figure 1. *Tree of the proof that "box1" can be stacked on "endtable1".*

In this trace the descriptors we have chosen to call non-operational have been circled. By pruning the non-operational descriptors and generalizing, we obtain the following explanation structure.

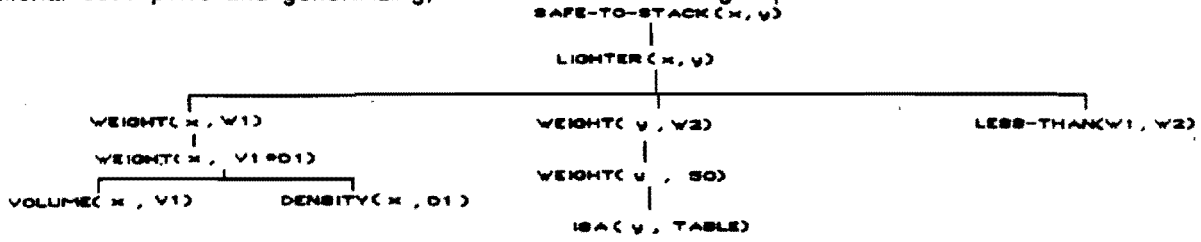


Figure 2. *Pruned tree of the proof. Non-operational descriptors have been removed.*

EBG then regresses a general goal through the explanation structure, and propagates the obtained instantiations through the whole set of regressed sub-goals. The final set of sub-goals becomes the new conditions for the achievement of the general goal. The figure below shows the explanation structure for SAFE-TO-STACK.

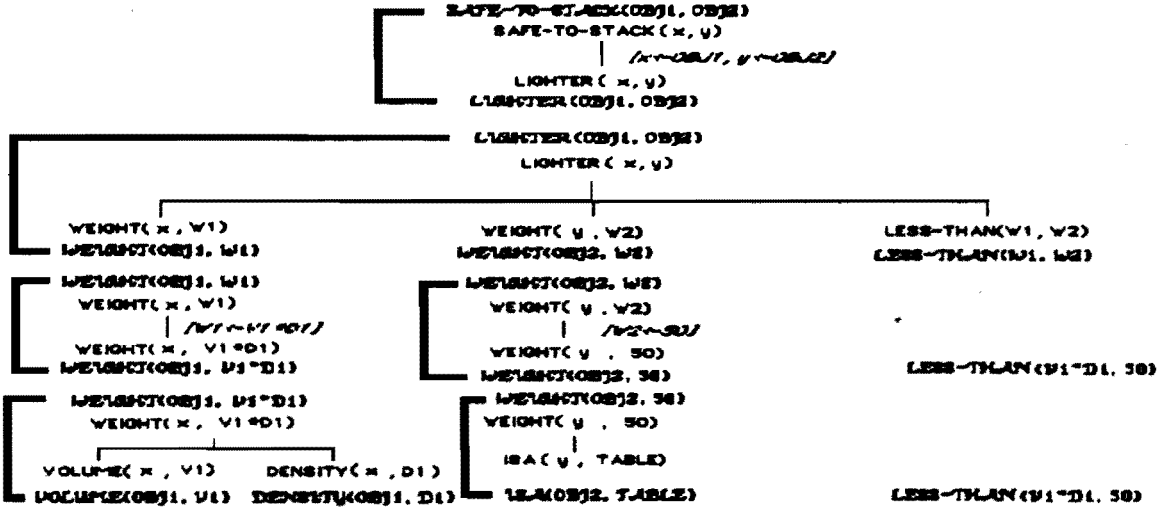


Figure 3. *Goal regression through the pruned tree of the proof.*

Around each part of the explanation structure are the goals and their regressions indicated in Venice font. At each level of regression, the substitutions are transmitted in all the sub-goals. The last line of sub-goals is the regression of 'SAFE-TO-STACK(OBJ1, OBJ2)' through the explanation structure. The rule learned is therefore

SAFE-TO-STACK(OBJ1, OBJ2) :- VOLUME(OBJ1, v1), DENSITY(OBJ1, d1), LESS-THAN(v1 * d1, 50), ISA(OBJ2, TABLE)

where v1, d1 are variables, and where the "50" comes from C13.

2.2 - EBL and the refinement of strong theories

For the management of incomplete theories, the central idea is to complete the failure proofs by an abduction mechanism [Duval and Kodratoff 1989]. This abduction process is guided by analogical reasoning about explanations and enables to discover new rules that refine the domain theory.

Remember section 1.1.1 in which we considered that some training instances led us to learn
KILL(x,x) :- DEPRESSED(x), BUY(x,y), WEAPON(y).
Recall also the story of Mary's suicide, which comes with the information
DEPRESSED(MARY).
BUY(MARY,OBJ1).
SLEEPING-PILLS(OBJ1).
PRICE(SLEEPING-PILLS, 20).
BUY(MARY, OBJ2).
BOOK(OBJ2).
We then proposed to explain Mary's suicide by adding
KILL(x,x) :- DEPRESSED(x), BUY(x,y), SLEEPING-PILLS(y)
or
KILL(x,x) :- DEPRESSED(x), BUY(x,y), BOOK(y)
to the knowledge base. These two rules have been built because, when applied to this story, they lead to a resolution tree which is analogous to the one obtained to explain the earlier stories. In that way, the theory is completed since we are now able to prove Mary's suicide as well.
We are still left with the choice between the two possible clauses. One way to choose is considering theories of different granularities that support each other. By granularity of a theory, we mean to describe the level of detail it copes with. For instance, a sociology of suicide does not need usually to take into account the details about the chemical reactions that result in the death process. Even though social details can be of much subtlety, we say that, in this context, molecular chemistry is of finer grain, and will be discarded in a first step. Suppose that our theory (theory1) has been so far of granularity 1, and that we have also a theory (theory2) of granularity 2 in which the "deeper" properties of the descriptors of theory1 are given. Theory2 will contain knowledge about the use of sleeping pills, books, and other objects that Mary may have been buying. Theory2 will have also more detailed knowledge about the way killing is performed, for example about the potential danger of all weapons. Thus, one should be able to prove that the potential danger of a book is of pure psychological nature, while the potential danger of an excess of sleeping pills is also physical. One could then come back to theory1 with an explanation of why the sleeping pills, provided they are used in excess, are suitable tools for suicide. Notice however that all this reasoning is done with uncertain arguments. For instance, eating books in excess will certainly also cause death, it is simply less probable that one ingests books in a physical manner. Supposing this kind of difficulties taken care of, theory2 can help revising theory1 by adding
KILL(x,x) :- DEPRESSED(x), BUY(x,y), SLEEPING-PILLS(y), EXCESS-INGEST(x,y)
If one is concerned only by choosing among the possible abductions, then one could also put aside all the stories uncounted for by
KILL(x,x) :- DEPRESSED(x), BUY(x,y), WEAPON(y)
and, after many of them are found, cluster those that present common features. If the stories are varied enough then "sleeping pills" should be the only common feature among several examples.

2.3 - Storing chunks of knowledge

After a problem solving session the trace of the solution obtained, or part of it, can be kept. Since such solutions are then stored in the knowledge base, it can be used to obtain immediate solution in further applications. At this stage, the problem consists of keeping only the most interesting chunks.
In practice, many other problems are met when the chunks are simply piles together, and that the system starts crumbling under its own weight. One then has to recognize when two chunks are equivalent or can be generalized into a more efficient one, and to organize the chunks into "chunks of chunks" which amounts to use meta-knowledge to sort them.
The system SOAR [Laird et al. 1986, 1987] which implements some of these ideas is currently under test in the industrial environment and one should soon learn about its achievements.

3 - EXPLANATION-BASED LEARNING IN SOFT THEORY DOMAINS

Three different approaches have extracted explanations from soft deep theories (by soft, we mean : incomplete and/or incoherent and/or intractable).
One is illustrated by the system PROTOS [Bareiss, Porter, and Wier 1989], the other one by the system DISCIPLE [Kodratoff and Tecuci 1987a, 1987b], and the third one by Shank's theory of XPs [Schank 1987].

3.1 - Definition of an explanation for PROTOS

In PROTOS, the production rules are associated to elementary explanations, that describe what kind of relation takes place between the antecedent and the consequent of each production rule. There are six such elementary explanations. Among them, one finds the classical ISA (generality relationship), and PART-OF (part to whole mappings), but also ENABLES telling that a feature enable the function of an object (e.g., wings enable flight).
An explanation is a combination of such elementary explanations as given by a path in the relationship frame among objects.

For example, one can say that the relation between engine and car is PART-OF. But a better explanation can be provided by considering a less direct path : the engine ENABLES the movement which is the FUNCTION of vehicles and car ISA vehicle. This knowledge is part of the training provided by the teacher to underline the explanations that are significant.
In the above example, suppose that the system is taught that the second path provides a good explanation, and not the first path. Then, in the future, when trying to detect if some unknown x is an engine, it will not try to prove that x is a part of an engine but it will try to see if x can enable the movement of some vehicle.

In PROTOS a good explanation is a path in the frame of the relationships among objects, the path provided by the user.

3.2 - Definition of an explanation for DISCIPLE

DISCIPLE is an interactive Learning Apprentice System for weak theory fields. In DISCIPLE, the learning process starts with an example of a problem solving step as, for instance, the following rule for the construction of loudspeakers.

Example1

ATTACH sectors ON chassis-membrane-assembly ⊢—
APPLY mowicoll ON sectors
PRESS sectors ON chassis-membrane-assembly

where A ⊢— B means : "in order to achieve action A, perform actions B", and where "mowicoll" is a kind of glue, and "sectors" and "chassis-membrane-assembly" are parts of a loudspeaker. This is interpreted as an example of a general rule indicating a way of performing the ATTACH action:

General Rule1

ATTACH x ON y ⊢—
APPLY z ON x
PRESS x ON y

Being completely instantiated, example1 implicitly contains the proof of its validity in the supposedly known properties of sectors, chassis-membrane-assembly, and mowicoll. On the contrary, since we have no knowledge of the possible properties of the variables found into General Rule1, it does not contain any implicit explanation.
Nevertheless, it is essential to obtain this knowledge, because it tells us when it is valid to perform an ATTACH operation as a sequence of APPLY and PRESS. In order to achieve this goal, the system over-generalizes the examples (by transforming Example1 in General Rule1, for instance) and applies this over-

generalization to the data basis of the user in order to find instances of the over-generalization. These instances are proposed to the user as tentative rules. When the user validates such a system-produced rule, we say that the system is provided with a positive example of its over-generalization. When the user rejects such a rule, we speak of a negative example. This set of positive and negative examples is used by a classical generalizer such as AGAPE [Kodratoff and al. 1984] to produce a more refined generalization. Finally, from these interactions with the user, DISCIPLE may find what is the correct generalization of the proofs of the examples. This gives a correct set of conditions for the application of the generalized rule, therefore a good explanation.

For instance, the final rule DISCIPLE will learn, from Example1 and further interactions with the user, is the following.

General Rule2

IF
z ISA adhesive
z TYPE pure
z GLUES x
z GLUES y
THEN
ATTACH x ON y ⊢
APPLY z ON x
PRESS x ON y

Since the structure of the rule is precisely the structure of the example, what is to be learned is just the explanation of the rule.

An explanation in DISCIPLE is a set of conditions that authorizes the application of a general rule. For instance, the conditions of General Rule2 are an explanation of General Rule1. It is always obtained as the generalization of the particular proofs of the example rules provided to the system.

The definition of an explanation is therefore very similar to the one of section 2, the difference is in the generalization process that is used. DISCIPLE is just in the middle between strong-theory-EBL and PROTOS. In strong-theory-EBL the explanations are provided by the system to the user, in PROTOS they are provided by the user to the system, in DISCIPLE they are obtained as the final result of a consultation of the user with the system.

3.3 - Pre-stored explanations : Shank's XPs

Roger Schank has recently proposed his own vision of Machine Learning and AI, of which creativity is the central theme [Schank 1987]. Nevertheless, in this theory, the essential creative information is constituted by a set of Explanation Patterns (XPs). These XPs are pre-stored explanations that can be applied in a variety of situations. Scripts [Schank and Abelson 1977] are classical in AI, they contain pre-stored information about standard situations. Let us define an XP as being a script for an explanation.
An XP is made of four parts

- an index that allows to get at the XP,
- a set of states of the world under which the XP can expect to be activated,
- a scenario which is a causal chain of states and events,
- the resultant state following from the scenario application.

For instance, in the case of the unexpected death of a promising racehorse named Swale, Schank [Schank 1986] proposes several XPs containing the following scenarios :

"Early death comes from being malnourished as a youth"
"High living brings early death"
"An inactive mind can cause the body to suffer"
"High pressure jobs cause heart attacks"

...

The first scenario leads to ask wether Swale underwent bad treatments during its youth, etc ...
In the case of PROTOS and DISCIPLE, the problem could be viewed as the obtainment of new XPs. In this case, the explanations are pre-recorded and the problems are

- the retrieval of the relevant ones from a certainly huge amount of them,
- the application through some analogical instantiation process to the particular case under consideration
- their modifications to fit into unexpected situations (tweaking them, as Schank puts it).

In practice, one can guess some kind of XPs will have to be given beforehand to the system, but also that no system will be able to run a real life problem without a mechanism allowing it to generate its own explanations.

4 - ANALOGICAL REASONING

First, let us briefly recall what analogical reasoning is. For more details, see for instance [Chouraqui 1985], [Davies 1987] or [Kodratoff 1988]. Let us suppose that we dispose of a piece of information, called the source S (which can be viewed as a knowledge base) and that this information can be put into the form of a doublet (A, B) in which B depends (often causally) on A. Let us suppose now that another piece of information, called the target T, can be put into the same form (A', B') with the requirement that there exists some resemblance between A and A'.

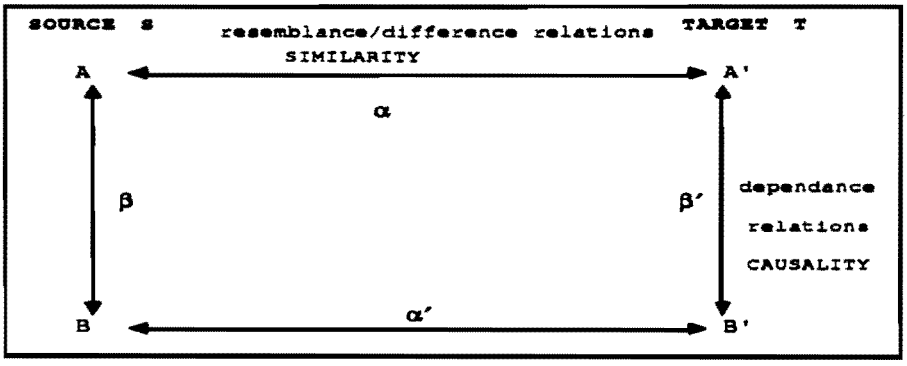


Figure 4. *The general scheme of analogy.*

This scheme can be used in different ways to define analogical reasoning. In most cases, we consider that we know more or less precisely A, B, A', α and β and that we try to invent B' or to justify its plausibility.
Let us illustrate a different use of analogy [Winston 1982] in which the result of the analogy is finding the causality relation ß allowing an analogy to be drawn. Consider the following text.

Let 'A = Earth has an atmosphere of 1013 milli-bars (MB) containing oxygen', 'B = Earth is inhabited by humans', 'A' = Mars has an atmosphere of 5 MB containing oxygen', 'α = both Earth and Mars have an atmosphere containing oxygen, but the atmospheric pressure is 1013 on Earth, and 5 on Mars'.

The analogy problem is to complete the diagram in Figure 5, in which β and β' are missing

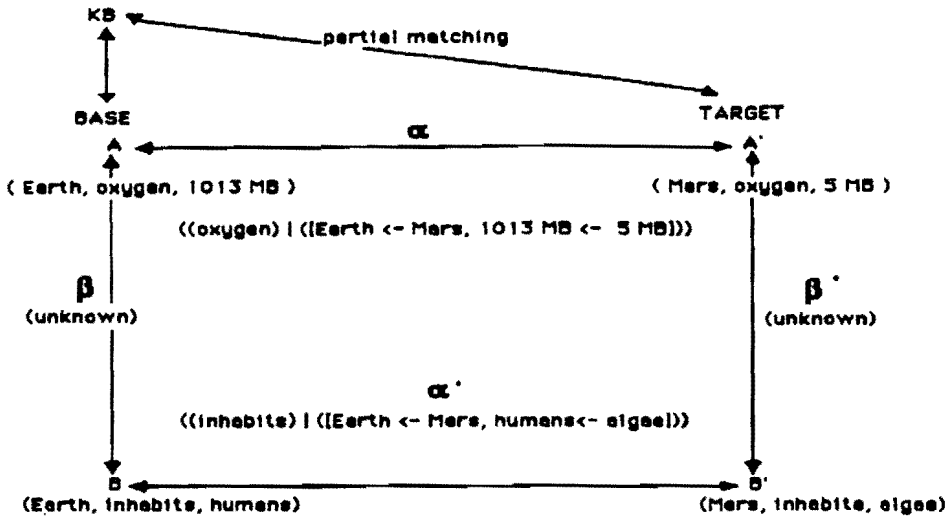


Figure 5. *An application of the general scheme to a specific analogy.*

In that case, we need, to be realistic, a knowledge base from which A can be extracted. Let us therefore suppose that we have access to a huge knowledge base KB about earth-life, in which the A above is contained together with a lot of information about life conditions on Earth.
Suppose that we know for sure that A' = *The atmosphere of Mars contains oxygen and its atmospheric pressure is 5MB*. Suppose also that we are asking the question: Is there any reason for us to believe that some algae live on Mars? Therefore, B' is assumed to be *algae live on Mars*. Now, the process of analogy will consist of finding some information within KB that is relevant to A'. In this simple example, we can easily find that A = *Earth atmosphere contains oxygen and its pressure is 1013MB*, which obviously a very good match for A'. In a more realistic example one can imagine that finding A within KB is a tedious process. Now that A has been found, we try to determine B by finding in KB "something" that lives on Earth and has a causality relationship with oxygen and atmospheric pressure. Suppose, for example, that there is a causality relationship between oxygen, atmospheric pressure, and the presence of humans on Earth. Thus β is discovered to be *humans need oxygen and a 1013MB atmospheric pressure to live*, and B is discovered as *humans live on Earth*. Now, since we are assuming that β = β', we can also claim that algae live on Mars because they need oxygen (as humans do on Earth), and they need a 5MB atmospheric pressure (unlike the humans that need a 1013MB pressure).

Notice that there exists several other approaches to analogy that we did not study here. The interested reader should refer to [Michalski et al. 1983, 1986, Priedetis 1988, Kodratoff 1988]

4 - INDUCTIVE LEARNING: ID3, INDUCE, STRUCTURAL MATCHING

Let us present here three induction methods ordered by the amount of background knowledge they rely on. ID3 is typically knowledge poor. Nevertheless, all the work done on ID3 from the original version [Quinlan 1983] to its recent improvements (see for instance [Bratko and Lavrac 1987]) are centered on understandability rather than on efficiency, which, by the way, gives to these works their deep originality as well as their belonging to AI. Surprisingly enough, it seems that most improvements in understandability have been followed by efficiency increase as well [Bratko 1988].

4.1 - ID3

4.1.1 - Stating the problem

The aim of this method is as follows. Given a set of descriptors, of examples and of concepts the examples belong to, find the most efficient way to classify all the examples under the concept they belong to, i.e. find the most efficient way to "recognize" the examples. The method relies on information theory, it measures the amount of information associated with each descriptor, and chooses the most informative one. By applying descriptors in succession, a decision tree is built that will recognize the examples. As opposed to similar numerical techniques, ID3 preserves the understandability of its results by avoiding introducing linear combinations of descriptors.

Example. Suppose that we start with 2 concepts (A and B) described by 3 descriptors (size, nationality, family) that can take the values (small, large) for the descriptor size, (French, German, Italian) for nationality, and (married, single) for family.
Suppose that the concepts are illustrated by the examples in table 1.

Table 1.

CONCEPT A			CONCEPT B		
size	nationality	family	size	nationality	family
small	German	single	small	Italian	single
large	French	single	large	German	married
large	German	single	large	Italian	single
			large	Italian	married
			small	German	married

ID3 will find an optimized decision tree to recognize concept A from concept B.

4.1.2 - Numerical optimization

For each descriptor that has not yet been used, the disorder left after applying a descriptor in the decision tree is computed. The one that leaves the less disorder is chosen as next node of the tree. The process stops when each leaf of the tree contains examples of one concept only. The disorder is computed after the classical entropy formula

$$E = - \sum f_s \log_2 f_s$$

where f_s is the relative frequency of class s.

In our example the initial disorder is given by

$$E0 = -(3/8 \log_2 3/8 + 5/8 \log_2 5/8) = 0.954$$

Let us now compute the information gain of applying each descriptor.
For instance, the information value of *nationality* is obtained by analyzing the result of classifying the examples by nationality and computing the disorder after applying *nationality*. One has

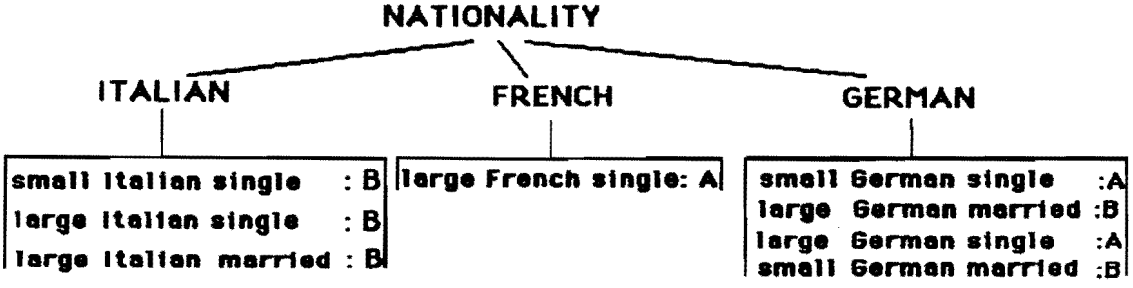


Figure 7. *Result of the application of descriptor "NATIONALITY".*

The disorder in the cluster *Italian* is: $-(3/3 \log_2 3/3) = 0$, in the cluster *French*, it is $-(1/1 \log_2 1/1) = 0$, and in the cluster *German*: $-(2/4 \log_2 2/4 + 2/4 \log_2 2/4) = 1$. Therefore, the disorder left after applying *nationality* is

$$E1 = (3/8 * 0) + (1/8 * 0) + (4/8 * 1) = 0.5$$

The gain in information is E0 - E1 = 0.454.
The same computation is done for, say, *family*. Applying *family*, we get

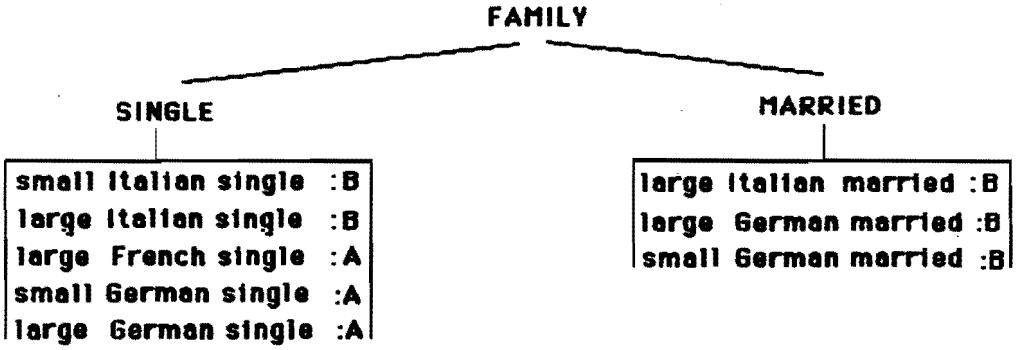


Figure 8. *Result of the application of descriptor "FAMILY".*

The disorder after applying *family*, in the cluster *single*: $-(2/5 \log_2 2/5 + 3/5 \log_2 3/5) = 0.971$, and in the cluster *married*: $-(3/3 \log_2 3/3) = 0$. The disorder left after applying *family* is therefore

$$E'1 = 5/8 * 0.971 + 3/8 * 0 = 0.607$$

and the associated gain in information is E0 - E'1 = 0.347. The gain is less than when applying *nationality*, which should therefore preferred to *family*.
In order to complete our example and show a decision tree, let us now compute the information value of *nationality* followed by *family*.
Applying them in order leads to

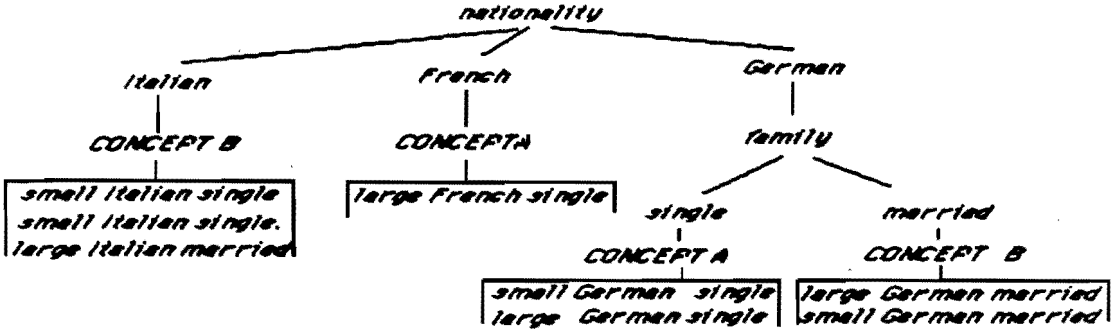


Figure 9. *Result of the application of descriptors "NATIONALITY" followed by "FAMILY".*

in which all leaves contain example of a single concept. This is therefore a complete decision tree allowing us to decide wether an example belongs to concept A or to concept B.

From the initial examples, we knew that applying the three descriptors in any order was indeed a decision procedure, we have now learned that applying *nationality* and *family* is enough, and that we optimize the search speed by applying them in that order.

4.1.3 - Improvements on ID3

All improvement performed are relative to a better understandability of the results. For instance, in some cases it seemed that preferring binary trees branching on their right only would improve understandability [Arbab and Michie 1985]. Other notable improvements are

- pruning the tree (see [Bratko and Lavrac 1987]),
- transforming the tree into decision rules [Corlett 1983]
- using background knowledge [Kodratoff et al. 1987, Nunez 1988]

4.2 - INDUCE

4.2.1 - **Methodology**

INDUCE builds a function R that recognizes positive examples and rejects negative ones, i.e. a complete and coherent function. In order to achieve this goal, it starts by building the most general recognition function R_i which recognizes the example e_i and rejects the set of examples $\{e_j\}$. Details on how to built such function will be given later.

Let POS be the set of positive examples, and NEG be the set of negative examples, let e_i be the current member of POS. The algorithm to compute R is as follows
First, as we said, suppose that we have been able to compute R_i, the recognition function that recognizes e_i (and some others of POS), and rejects all instances of NEG. If it happens that R_i recognizes all of POS, we have obtained a complete and consistent recognition function, $R = R_i$. If not, choose an other example e_j, and compute R_j. Then, $R_i \vee R_j$ (where $\vee$ is the logical disjunction) will improve on e_k, it is consistent by construction. Go on as long $R_i \vee ... \vee R_n$ is not complete.
Example. Suppose that we have ten examples described by means of four descriptors and their values, as shown in table 2.

Table 2.

	X1	X2	X3	X4
e1	light	boy	baby	red
e2	light	girl	baby	blond
e3	light	boy	child	chestnut
e4	medium	woman	teenager	chestnut
e5	medium	boy	child	red
e6	heavy	girl	child	blond
e7	heavy	man	teenager	red
e8	heavy	woman	teenager	chestnut
e9	heavy	man	adult	blond
e10	heavy	woman	adult	chestnut

The background knowledge is expressed by the properties of the descriptors. They can be linear like X1, linear descriptor describing the values of weight. This will be expressed by the formula {light < medium-weight < heavy}.
Some descriptors can also be structured like X2 and X3.
The background knowledge relative to X2 may be given by the following taxonomies of generality.

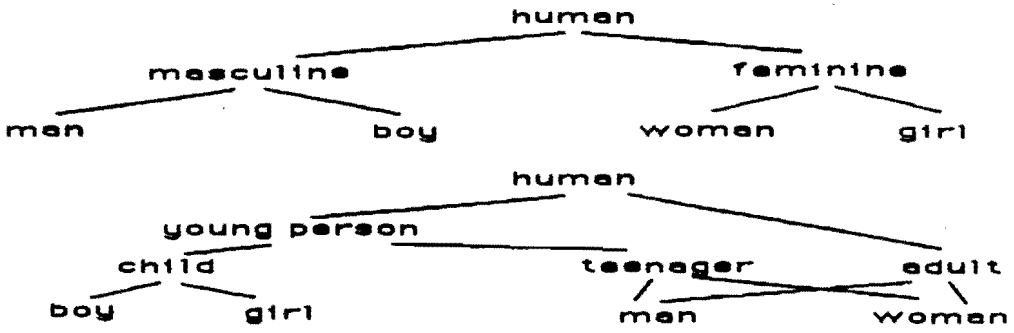


Figure 10. *"Taxonomies" of generality between the descriptors.*

It can also happen that no particular knowledge is available for a descriptor, like X4 which is a nominal descriptor. This knowledge is expressed by describing the set of values as an unstructured set: {blond, red, chestnut}.

4.2.2 - **Building R_i**

Definition 1.

Let us define a recognition function R as being described by

$$R = (Xi = Vi) \& \ldots \& (Xk = Vk)$$

where the Xj's are descriptors and the Vj's are the values of these descriptors or disjunctions of possible values.

For example, follow four different recognition functions:

- X2 = man,
- X2 = man ∨ woman ∨ boy,
- X3 = [baby .. teenager],
- (X2 = man) & (X3 = [baby .. teenager]).

Definition 2: Subset recognized by a recognition function.

This is the subset of examples the descriptor of which show the same value as the recognition function.
For instance, the recognition function R1: X2 = man ∨ woman ∨ boy recognizes the subset {e1, e3, e4, e5, e7, e8, e9, e10}. The function R2: (X1 = heavy ∨ light) & (X4 = red ∨ chestnut) recognizes {e1, e3, e7, e8, e10}.

Creation of a recognition function that recognizes e_i and rejects e_j: $G(e_i/e_j)$.

In the beginning, $G(e_i/e_j)$ is empty. Then, if the descriptor Xi has same value in e_i and in e_j, then Xi does not occur in $G(e_i/e_j)$. If the descriptor Xi is different in e_i and e_j, let Vj be the value of Xi in e_j, then add the disjunct ∨ (Xi ≠ Vj) to $G(e_i/e_j)$.
Example. e1 and e2 differ by X2 and X4. In e2: X2 = girl and X4 = blond. It follows that G(e1/e2) = (X2 ≠ girl) ∨ (X4 ≠ blond). Similarly, one sees that G(e1/e4) = (X1 ≠ medium) ∨ (X2 ≠ woman) ∨ (X3 ≠ teenager) ∨ (X4 ≠ chestnut).

Comparison of an example and a set

We are now able to compute the most general recognition function which recognizes the example e_i and rejects the set of examples $\{e_j\}$. Let us note it by $G(e_i/\{e_j\})$. This is what we have been calling R_i above.

For each e_k in $\{e_j\}$, calculate $G(e_i/e_k)$.
The conjunction of all the G obtained is $G(e_i/\{e_j\})$:

$$R_i = G(e_i/\{e_j\}) = \&_{e_k}\ G(e_i/e_k),\ e_k \in \{e_j\}$$

Example.
$G(e1/\{e2, e4\}) = G(e1/e2)$ & $G(e1/e4)$ = ((X2 ≠ girl) ∨ (X4 ≠ blond)) & ((X1 ≠ medium) ∨ (X2 ≠ woman) ∨ (X3 ≠ teenager) ∨ (X4 ≠ chestnut)) which is then put in normal form.

4.2.3 - Improving the recognition functions

These improvements are performed by using numerical information and also background knowledge. A typical symbolic improvements minimize the number of disjuncts, improve the simplicity of the description, replacement of disjunctions by interval when the descriptors are linear.

4.3 - Structural Matching

4.3.1 - Definition of Structural Matching (SM)

Two formulas structurally match if they are identical except for the constants and the variables that instantiate their predicates.

More formally: Let E1 and E2 be two formulae, E1 structurally matches E2 iff there exists a formula C and two substitutions s_1, s_2 such that:

1-$s_1 \circ C$ = E1 and $s_2 \circ C$ = E2.

2-s_1 and s_2 never substitute a variable by a formula or a function.

where $\circ$ denotes the application of a substitution to a formula.
It must be understood that SM may be difficult up to undecidable. Nevertheless, in most cases, one can use the information coming from the other examples, in order to know how to orientate the proofs necessary to the application of this definition (for details, see [Vrain 1987]). Even if SM fails (which may often occur), the effects of the attempt to put into SM may still be interesting. We say that two formulae have been SMatched when every possible property has been used in order to put them into SM. If the SM is a success, then SMatching is identical to putting into SM. Otherwise, SMatching keeps the best possible result in the direction of matching formulae.

4.3.2 - A simple example of successful SM

Let us consider the following two examples. E1 represents A piled on B, and E2 represents C and D, as shown in figure 11.

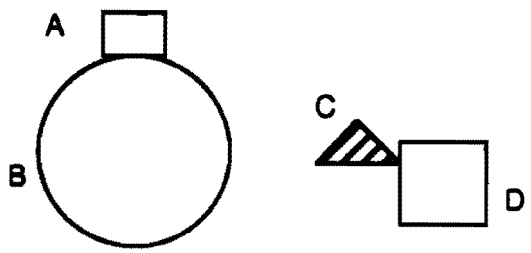


Figure 11. *E1 is the left part of the figure, E2 is its right part.*

The examples can be described by the following formulas
E1 = SQUARE(A) & SMALL(A) & UPON (A, B) & CIRCLE(B) & BIG(B) & UNDER(B, A)
E2 = TRIANGLE(C) & SMALL(C) & LEFT_SIDE(C, D) & SQUARE(D) & SMALL(D) & RIGHT_SIDE(D, C)

Let us suppose that the following hierarchies are provided to the system

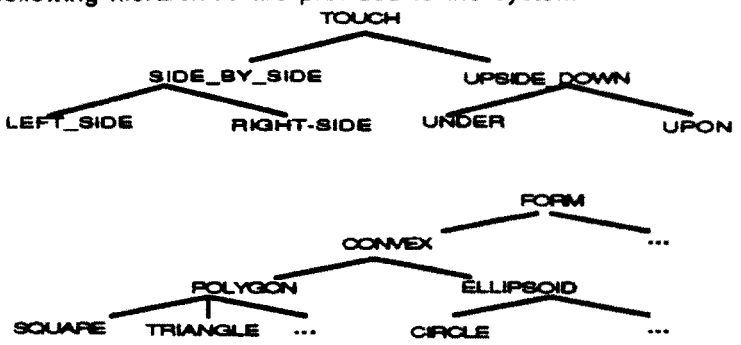


Figure 12. *Taxonomies of generality relationships among the descriptors.*

These taxonomies represent our semantical knowledge about the micro-world in which learning is taking place. The SM of E1 and E2 proceeds by transforming them into equivalent formulas E'1 and E'2, such that E'1 is equivalent to E1, and E'2 is equivalent to E2 in this micro-world (i.e., taking into account its semantics). When the process is completed, E'1 and E'2 are made of two parts.
- one is a variablized version of E1 and E2, called the body of the SMatched formulae. When SM succeeds, the bodies of E'1 and E'2 are identical.
- the other part, called the bindings (of the variables), gives all the conditions necessary for the body of each Ei' to be identical to the corresponding Ei.
The process starts as follows. Suppose that we know that the descriptors of the taxonomy 'FORM' are "more interesting" than those of the taxonomy 'TOUCH'. We will therefore start attempting to match the 'SQUARE' and 'CIRCLE' of E1 with the 'SQUARE' and 'TRIANGLE' of E2. We replace 'A' in E1 and 'D' in E2 by the pseudo-variable 'x', and keep memory of this value in the bindings. This gives (note that the list of bindings are between brackets):

E'1 = SQUARE(x) & SMALL(x) & UPON (x, B) & CIRCLE(B) & BIG(B) & UNDER(B,x) & [(x=A)]
E'2 = SQUARE(x) & SMALL(x) & RIGHT_SIDE(x, C) & TRIANGLE(C) & SMALL(C) &
LEFT_SIDE(C, x) & [(x=D)]

The next step uses the taxonomies by matching the 'CIRCLE' of E1 to the 'TRIANGLE' of E2. One obtains

E"1 = SQUARE(x) & SMALL(x) & UPON (x, y) & CONVEX(y) & BIG(y) & UNDER(y,x) & [(x=A)
& (y=B)]
E"2 = SQUARE(x) & SMALL(x) & RIGHT_SIDE(x, y) & CONVEX(y) & SMALL(y) &
LEFT_SIDE(y, x) & [(x=D) & (y=C)]

This example shows well that, once this SM step has been performed, the generalization step itself becomes trivial: we keep in the generalization all the bindings common to the SMatched formulas and drop all those not in common. In other words, this SM technique allows to reduce the well-known generalization rules [Michalski 1983, 1984] just to the "dropping condition rule" which becomes legal on SMatched formulas. All the induction power is in the dropping condition rule, all other rules are purely deductive. We must confess that formal proof of the above statement is still under research.

Since 'BIG' and 'SMALL' do not belong to a common taxonomy in our example, they will be dropped, and since 'x' and 'y' have always different values this will be noted in the final generalization which is

Eg = SQUARE(x) & SMALL(x) & TOUCH(x, y) & CONVEX(y) & TOUCH(y,x) & [(x≠y]

In this example, one feels that one could use theorems of the like $\forall x,y$ [TOUCH(x,y)$\Leftrightarrow$TOUCH(y,x)] in order to improve the generalization.

4.3.3 - Using theorems to improve generalization

It can be easily guessed that using theorems can lead to many difficulties, since one enters the realm of Theorem Proving, which is well-known for being a good source of yet unsolved problems. In the case of SM, one is driven by the need to put the examples into a similar form, and the usual difficulties of Theorem Proving are somewhat eased. We cannot formally prove this point, but the following example, taken from [Vrain 1987] can at least illustrate our claim.

Starting from two examples that have no common predicates, we show that they nevertheless have a common generalization, found by using theorems that link the predicates.

Let the examples be

E1 = MAMMALIAN(A) & BRED_ANIMAL(A)

E2 = TAME(B) & VIVIPAROUS(B)

to which the following theorems are joined

R1: $\forall x$ [MAMMALIAN(x) & BRED_ANIMAL(x) $\Rightarrow$ TAME(x)]

R2: $\forall x$ [TAME(x) & VIVIPAROUS(x) $\Rightarrow$ MAMMALIAN(x)]

R3: $\forall x$ [TAME(x) $\Rightarrow$ HARMLESS(x)]

where $\forall$ is the universal quantification, & the logical conjunction, and $\Rightarrow$ the logical implication.

The first step of SM is here trivial: we replace the constants by a variable x, and obtain the equivalent examples:

E'1 = MAMMALIAN(x) & BRED_ANIMAL(x) & [(x= A)]

E'2 = TAME(x) & VIVIPAROUS(x) [(x= B)]

Since the predicates have no common occurrence, we consider the first (this ordering is not significant, and just follows the one in which the examples are given) predicate of E'1: MAMMALIAN. We see that we can deduce this predicate from E2, using the rule R2. We get:

E"1 = MAMMALIAN*(x) & BRED_ANIMAL(x) & [(x= A)]

E"2 = TAME(x) & VIVIPAROUS(x) & MAMMALIAN**(x) & [(x= B)]

The MAMMALIAN of E'1 has been treated, this why it is marked by an * in E"1. The one of E"2 is derived from the use of theorems, this is why it is marked by **.

Using again the order in which the examples are given, the next non-marked predicate is BRED_ANIMAL.

- No rule can be applied to E"2 to make explicit the presence of BRED_ANIMAL in it.
- Nevertheless, we remark that applying the rule R1 to E"1 uses the predicate concerned: BRED_ANIMAL. Checking the effect of this application, we see that it generates the atomic formula TAME(x) and that there is an occurrence of x in E"2 which matches this occurrence. Therefore, we conclude that we must apply R1 to E"1.

One obtains

E'"-1 = MAMMALIAN*(x) & BRED_ANIMAL*(x) & TAME**(x) & [(x= A)]

E'"-2 = TAME*(x) & VIVIPAROUS(x) & MAMMALIAN**(x) & [(x= B)]

Now, the only unmatched predicate is VIVIPAROUS in E"2.

- No rules can be applied to E"'1 to make its presence explicit.
- The only rule which can be applied in E"'2, relative to VIVIPAROUS is R1. But, it would introduce the atomic formula MAMMALIAN(x), which is already matched since its instances are starred.

No other rule can be applied, we star the predicate VIVIPAROUS to remember that it has already been dealt with, obtaining:

E""1 = MAMMALIAN*(x) & BRED_ANIMAL*(x) & TAME**(x) & [(x=A)]

E""2 = TAME*(x) & VIVIPAROUS*(x) & MAMMALIAN**(x) & [(x= B)]

All possible occurrences have been dealt with, a complete SM is not possible, therefore the SMatching operation stops here.

Now, the generalization step is trivial: one drops the non-common occurrences, obtaining the generalization

G = TAME(x) & MAMMALIAN(x)

This example shows well how potential infinite proof loops can be easily avoided, simply because they do not improve the SMatching state of the examples. More generally, one can use theorem proving techniques in order to improve the degree of similarity detected among the examples. Such a system is under development in our group [Vrain 1987]. It is not the concatenation of a classical theorem prover and of generalization algorithms, but instead is rigorously adapted to the kind of proofs required by Machine Learning. As an instance of its peculiarity (and of its incompleteness), it will not allow to use twice the same theorem during a given derivation.

5 - THE QUESTIONNAIRE

Answer NR when you think that the question is Not Relevant to your system. When you do not answer a question, NA (No Answer) will be automatically added. Concepts used in this questionnaire are defined in the glossary found in appendix.

3.1 - Background knowledge
Knowledge intensive/knowledge poor
Representation of background knowledge

3.2 - Meta-knowledge
Have you ways to express meta-knowledge? yes/no
If yes describe

3.3 - Inputs
Representation of inputs (make precise their logic (0th, 1st order, ...), their semantic (what do their represent: Rules, conditions, ...?), the language you use (give an example if necessary)

3.4 - Outputs
Representation of output (make precise their logic (0th, 1st order, ...), their semantic (what do their represent: Rules, conditions, ...?), the language you use (give an example if necessary)

3.5 - Examples
You give positive only/negative only/positive + negative examples?
Is the system able to generate itself new examples? yes/no

3.6 - Incrementality
Describe how incrementality is achieved, if yes/no

3.7 - Oracle
Do you use an oracle, a professor, or any other human interaction? yes/no
If yes, describe the interaction
If yes, how the oracle gives information to the system?

3.8 - Learning engine
Learning engine: induction/deduction/evaluation/merging of ...

3.9 - Algorithm
Give a short description (4 lines max) of the algorithm that does the learning (answer "NR" if you cannot do it in less than 4 lines). These descriptions will be given aside of the tables below.

3.10 - What is learned?
What is given to, learned by the system?

3.11 - Pre-clustering

Examples ordered prior to learning/system accept unordered examples
System creates clusters of examples? yes/no

3.12 - Quality criterion
What are your own criteria to judge if your program has been behaving "well" or "bad"?

3.13 - Stability
Is your program stable under learning twice the same data (in a different order, for instance)? yes/no
If no describe

3.14 - Bias
Do you have an idea of the way you introduce preference bias? yes/no
If yes describe

3.15 - Theoretical model
Do you have a theoretical model relative to which you can validate your learning? yes/no
If yes describe

3.16 - Complexity
Can you evaluate the complexity of your algorithm as a function of the number n of examples, and their size s?

3.17 - Constructive
Is your system constructive in the meanings 1/2/3 that follow?

1 - it uses its own learned knowledge to improve its performance,
2 - it combines induction and deduction
3 - it uses explanations to validate the newly acquired knowledge, and using it again.

6 - ANSWERS

NAME OF SYSTEM	INSTITUTION	CONTACT PERSON(S)
ID3-TI	TURING INSTITUTE, Glasgow	T. Niblett
CN2	TURING INSTITUTE, Glasgow	P. Clark, T. Niblett
CIGOL	TURING INSTITUTE, Glasgow	S. Muggleton
ASSISTANT	Josef Stefan Inst., Ljubljana	B. Cestnik, I. Kononenko, I. Bratko
GINESYS	Josef Stefan Inst., Ljubljana	M. Gams
LogArt	Josef Stefan Inst., Ljubljana	B. Cestnik
NINA	Nixdorf, Paderborn	A. Ludwig, R Reppenhagen
DUCE	TURING INSTITUTE, Glasgow	S. Muggleton
ALEXIS	Telfonica Invest., Madrid	M. Nunez
SICLA	INRIA, Paris	E. Diday
MAKEY	Univ. Paris 6	J. Lebbe, R. Vignes
CONNECTIONISM (CONNEC)	(version of) EIHE, Paris	F. Fogelman
INSTIL	ESPRIT consortium, Europe	M. Manago
INDUCE	(version of) BRITISH AEROSPACE, Bristol	D. Hutber
ML-SMART	D. Informatica Univ. Torino	A. Giordana
ORLON	D. Informatica Univ. Torino	S. Bruno
RIGEL	D. Informatica Univ. Torino	R. Gemello
OGUST	LRI, Orsay	C. Vrain
DISCIPLE	LRI, Orsay	G. Tecuci
BLIP	TUB, Berlin	K. Morik
EBG	MANY VERSIONS RE-IMPLEMENTED	
EBGF	LRI, Orsay	J. F. Puget
LIFE	LRI, Orsay	J. F. Puget
EBG (extended)	AI dept., Edinburgh	R. Desimone
ALEX	INSERM, Paris	B. Seroussi
MACHIN	LIFIA, Grenoble	C. de Sainte Marie
ALLY	IRISA, Rennes	J. Nicolas
CHARADE	LIFIA, Paris	J. G. Ganascia
CALM	CRIM, Montpellier	J. Quinqueton, J. Sallantin
PLAGE	CRIM, Montpellier	O. Gascuel
REFINER	DCS, Aberdeen	D. Sleeman, S. Sharma
INFER	DCS, Aberdeen	D. Sleeman, I. Ellery
MORE	Univ. Coimbra, Portugal	E. Costa
CLINT	DCS, Kat. Univ. Leuven	L. de Raedt
INDE	Univ. Amsterdam	P. Terpstra
CONCLAVE	AI-Lab VUB Brussels	W. Van de Welde
KBLA for Tonal Music	Res. Inst. for AI, Vienna	G. Widmer
AR4, BR3	Dept. Elec., King's Col. London	S. Kolabas
FM	DCS, City Univ. London	L. McCluskey
ZENO 1.1	Brainware Gmbit	J. Stender
MAVERICK	Brunel Univ.	A. MacDonald
BG	King's Col. London	A. Hutchinson
SEA	Ecole Polytechnique Palaiseau France	M. Sebag, M.Schoenauer
THUMBRULE	CADEPS, VUB Brussels	H. de Garis

	Background knowledge	*Meta-knowledge*	*Inputs*	*Outputs*	*Examples*
ID3-TI	know. poor	no meta-know	0th order classes of attrib.	0th order decision trees or rules	positive+negative dont generate
CN2	know. poor	no	0th order classes of attrib.	ordered list of 0th order IF-THEN rules	positive+negative dont generate
CIGOL	Know. intensive 1st order Horn clauses with function symbols	rules for know. transformation	rules in pure PROLOG	rules in pure PROLOG	positive+negative generate
ASSIS-TANT	know. poor	no	0th order, clas. & attr., can be continuous	0th order decision trees	positive+negative dont generate
GINESYS	know. poor	no	0th order classes and attributes	0th order rules	positive+negative dont generate
LogArt	know. poor	yes, rules about the particular domain	0th order classes and attributes	0th order rules with redundancy	positive+negative
NINA	know. poor , planned to become intensive	no	0th order classes and attributes	0th order rules extensions toward 1st order	positive+negative dont generate
DUCE	know. poor	no meta-know	0th order classes	decision trees or rules New concepts	positive+negative dont generate
ALEXIS	Taxonomies, costs, demons	no	attributes	decision tree	positive+negative
SICLA	know. poor	no meta-know	0th order, numerical or qualitative attr., classes	decision trees or rules clusters	p/n/p+n dont generate
MAKEY	know. poor, uses pre-order relations between descr. given by expert	rules of dependance between properties	0th order, conjunctions of multi-valued descriptors	0th order rules decision graphs	examples are concepts
CONNEC	explicit know. poor. Architecture of the net	no meta-know	0th order Boolean matrix	Boolean matrix dont generate	positive only
INSTIL	explicit know. intensive taxonomies, theorems, default values, belief coef.	object inheritance	1st order examples + classes	1st order decision trees, rules	p/n/p+n dont generate
INDUCE	generality taxonomies	no	0/1st order conj. of descr.	concepts	positive+negative dont generate
ML-SMART	knowledge intensive: frames, rules	yes, control rules	0th order, attr. of components of examples	graph of rules	positive+negative
ORLON	know. poor	yes, control rules	0th order Description of examples	taxonomy of concepts	positive only
RIGEL	now. poor	yes, control rules	1st order Description of examples	1st order rules	positive+negative
OGUST	explicit know. intensive taxonomies, theorems	no	1st order conj. of descr.	concepts	positive+negative dont generate
DISCI PLE	semantic net + rules, incomplete theory	no	rules 0th order	rules 1st order	yes, based on example gener.
BLIP	explicit know. intensive Horn clauses	meta & meta-meta rules	1st order facts	rules 1st order + concepts	NR
EBG	Complete theory supposed known	expl: operational. criterion, impl: proving strategy	1st order Example + validity proof	rules 1st order	positive only cant work with negative examples
EBGF	Complete theory given as Horn clauses with functional terms	expl: operational. criterion, impl: proving strategy	Example NOT implied by theory	failure condition	negative only

	Background knowledge	*Meta-knowledge*	*Inputs*	*Outputs*	*Examples*
LIFE	know. intensive STRIPS-like axioms	yes, determinism of some predicates	failures of plans	invariant given as conjunctions	negative only examples generated by problem solving
EBG (ext.)	know. intensive rules of inference Existing proof tactics	proof plans	proofs in intuitionist type theory	proofs plans in 1st order sorted logic	positive only
ALEX	taxonomies of generality + descriptors complexity	implicit strategies	traces of solutions	heuristics for using operators	positive only generate examp.
MACHIN	knowledge poor. Can possibly use taxonomies of generality to be completed	certitude coef.	couples (situation, operator) certitude coef.	conditional rules + "certitudes"	positive + neg.
ALLY	Complete theory as clauses without functional terms	implicit: resolution strategies	1st order examples	1st order concepts	p/n/p+n
CHARADE	know. intensive. Horn clauses without func. Typed descriptors	Properties of the system of rules.	0th order examples as conjunctions	Approximate rules with certainty fact.	p/n/p+n not generate
CALM	Implicit. Descriptors properties	selection of formula by information	0th order examples as driven criteria	0th order, estimate of fit conjunctions	positive+negative concept/formula
REFINER	know. poor, some know. as semantic net	no	0th order classes attribute-value pairs	0th order decision criteria/ rules	positive+neg. dont generate
INFER	Know. poor	state transformation rules	0th order problem/ answers pairs	new (MAL)-rules	Not relevant
MORE	know. intensive Incomplete theory as Horn clauses	no	Horn clauses	Horn clauses	positive only
CLINT	know. intensive subsets of Horn clause logic	yes, form of language	facts = 1st order ground instances	Horn clauses	positive+negative generates examples
INDE	know. intensive subset of PROLOG clauses	type of errors type of arithmetic operator	set of PROLOG facts	PROLOG output	positive+negative dont generate
CONCLAVE	contains hierarchies and causal model	no	problems + solutions	classifications + rules	positive+negative dont generate
KBLA Tonal Music	know. intensive: rules	heuristics for induction	examples of musical pieces	rules	positive+negative generate examples
AR4, BR3	know. intensive Descriptive + procedural as PROLOG clauses	pseudo-2nd order	examples of particle reactions	particle reactions	positive only generate ex.
FM	facts and rules strips-like operators	control rules	initial & goal states	solutions + control rules	NR. Uses planning trace as +/- exam.
ZENO 1.1	know. poor	no	0th order classes of attributes	0th order rules	positive+negative generates ex.
MAVERICK	know. poor	no	independent lang. arbitrary	representation predictions in input language	stream of positive ex. only dont generate
BG	know. poor	no	context free grammar, sequence of characters	context free grammar	positive only dont generate
SEA	know. poor	no	set of (attribute, discrete values)-class	0th order rules	positive+negative dont generate
THUMB-RULE	know. poor	no	conjuncts of attr/val pairs	disjuncts of conjuncts of attr/val pairs	positive+negative dont generate

	Incrementality	*Oracle*	*Learning engine*	*What is learned?*	*Pre-clustering*	*Constructive?*
ID3-TI	no	no	Entropy optim. in each class, particularization, post-pruning	given: classes -> decision tree for the classes	no	
CN2	no	no	general-to-specific BEAM research	lists of rules for the classes	no	no
CIGOL	yes, incremental generalization of new example	yes tells if stat. are ok gives name	induction & evaluation	given: description of situations learned: relations between objects	yes	yes, 1,2,3
ASSISTANT	no	yes, force attribute selection	induction	given examples -> decision tree for the classes	yes	yes, 1
GINESYS	no	no	induction	given: classes -> rules for the classes	yes	yes, 1
LogArt	yes, by rejecting no longer valid rules	no	induction	given examples -> selected rules: 2/3 attributes, credibility	yes	yes, 1
NINA	no	no	Entropy optim. in each class, particularization	rules	no	no
DUCE	NA	yes	generalization	given classes -> concepts	yes	
ALEXIS	no	no	trade off cost/entropy optimization	given classes -> decision tree for the classes	yes	
SICLA	depends of chosen algorithms	no	distance minimization in R^n	un-clustered examples -> clustered	no	no
MAKEY	no	no	Deduction. Evaluation	concepts descriptions -> concept identif. methods	no	no
CONNEC	yes. Based on incrementality	no	backpropagation of gradient, coefficient optimization	Connections weights giving desired output for given input	yes	
INSTIL	no	no	Entropy optim. in each class + generalization	decision tree for the classes + concepts at node of dec. tree	yes	
INDUCE	no	no	aggregation of samples algorithm	rules discriminating pos/neg examples	yes	
ML-SMART	no	no	merging induction and deduction	concept discriminant descriptions	yes	yes, 2
ORLON	no	no	induction	new concepts	no	NA
RIGEL	no	no	induction	concept discriminant descriptions	no	NA
OGUST	no	possible yes	generalization	given: clustered examples, get: intentional descriptions	yes	
DISCIPLE	yes	yes, based on oracle	EBL + SBL + analogy + retrieving from semantic net	problem solving rules from example solutions	no	
BLIP	yes	no	deduction + induction from facts	new rules from facts new concepts from contradictions	no create clust.	yes, 1 + 2
EBG	yes	no	theorem proving	more efficient rule	no	
EBGF	yes	no	merging of deduction & partial evaluation, some induction	recognition of dead ends described as generalizations of counter-examples	NR	yes, 2 3 is NR

	Incrementality	Oracle	Learning engine	What is learned?	Pre-clustering	Constructive?
LIFE	yes, after each backtrack of the pb solver	no	deduction	invariant patterns	yes	yes 1 3 is NR
EBG (extd)	NR	no	precondition analysis	in: example proofs out: proof plans	NR	yes, 1&3 (potentially)
ALEX	yes	no	recognition of sequences of operators, Generalization	in: solutions, out: heuristics for using operators	no	NA
MACHIN	yes, based on incrementality	no	over-generalization followed by refinement evaluation of belief coef.	learns heuristics for using operators + builds net of operator dependencies	no	yes, 1 & 3
ALLY	yes	no	theorem proving	given: clustered examples, get: intentional descriptions	yes	
CHARADE	no	no	top-down indcution	obtains rules valid on the whole set of examples	no	NA
CALM	one step of incrementality	no	induction, evaluation statistics	gets rules in favor of, and against a concept	no	yes, 2
REFINER	yes	yes, correc of errors	induction	learns distinguishing features of concepts	no	yes, 1 & 3
INFER	no	yes	induction	new (MAL)-rules	yes	yes, 1
MORE	yes	yes, completes theory	resolution	set of beliefs describing the cognitive model of an agent	yes	NA
CLINT	yes, based on	classifies generated examples	induction + deduction	concept descriptions	no	yes, 1 & 2
INDE	yes, incremental version of AQ	yes	merging of EBL, AQ15, error propagation	refined rules, new rules	no	NA
CONC-LAVE	yes	yes, user's observations	induction, evaluation of classification rules	classifications + rules	yes	yes, 1 & 2
KBLA for Tonal Music	yes, new rules added to KB	yes teacher	deduction + induction + analogy	rules for completing counter-point pieces	yes	yes, 1 & 2 & 3
AR4, BR3	yes, belief revision	of little use	induction + deduction	find valid particle reactions, composion of elementary particles, quantum principles	no	yes, 1 & 2 & 3
FM	yes	yes supplies tasks	induction + deduction	control rules	NR	yes, 1 & 2
ZENO 1.1	no	no	induction	rules for classes	no	no
MAVE-RICK	yes, based on	no	SBL + generalization + statistics	sequence of event description -> tree of predictive rules -> sequence of predictions	no	no, 1 is planned
BG	yes	no	induction + statistical clustering	new descriptors	no	yes, 1
SEA	no	no	induction	rules	no	NR (=no)
THUMB-RULE	no	no	induction	class rules	no	no

	Quality criterion	*Stability*	*Bias*	*Theoretical model*	*Complexity*
ID3-TI	comparison with classical ES, and human perf., Tree simplicity	yes, checked by partitioning exa.	yes, user-supplied pruning parameter	Information theory, statistics	n, s
CN2	comparison of rules accuracy & simpl with other systems	yes	user selects desired statistical significance of rules	statistics	p=nb attrib/ex. O(n, p)
CIGOL	performance against pre-computed solut. Human evaluation	yes	yes, statistical tests based on complexity in time ans space	Resolution theorem proving, and information theory	can be computed
ASSISTANT	check on testing set	yes	no	Information theory	O(n*s)
GINESYS	check on testing set comparison with statistical methods	yes	yes, "impurity", "error-estimate", parameters, etc ...	Information theory, Bayes' formula, statistics	O(n), $O(s^x)$, x>1
LogArt	check on testing set	yes	no	Bayes formula without attribute independence assumption	$O(n*s^2)$
NINA	comparison with classical ES and human performance	yes	not explicit	Information theory	O(n*s)
DUCE	NA (see CIGOL)	NA	NA	NA	NA
ALEXIS	comparison with classical ES and hum. performance	yes	NA	Information theory	NA
SICLA	yes, evaluated by missing axes of inertia	yes	choice in measure of distance	Statistics	n ≤ complex. ≤ n^4
MAKEY	form of the graphs	yes	attribute pre-order quality criterion	no	evaluation: $n^2 * \log n * s$
CONNEC	check on examples	yes	values of coefficients before learning + net architecture	connectionism	NA
INSTIL	expert + user tree depth, branching factor	Costs, reliability, explanations pref. by expert	beliefs on importance & reliability of descriptors	Information theory + object oriented languages	linear in n, s^2
INDUCE	expert	yes	simplicity measures choice of seed	no	NA
ML-SMART	performance on new examples, fit in theory and compactness	yes	explicit control rules	no	not checked
ORLON	increasing coherence inside clusters	yes	explicit control rules	categorization theory	not checked
RIGEL	completeness and consistency	yes	explicit control rules	no	not checked
OGUST	obtain most particular generalization	not checked	ordering of descriptors by importance	structural matching	not checked
DISCIPLE	expert	not stable	explicit in knowl. base	no	not checked
BLIP	better structuration of domain theory	not stable, depends on meeting neg. example	meta-knowledge	theorem proving	linear with nb of of facts
EBG	fit in theory	should	choice of strategy of proofs	theorem proving	same as theo. proving
EBGF	simplicity	yes	choice of strategy of proofs, operationality criterion	negation as failure	s=nb of proofs* size of proofs linear in s
LIFE	large changes in problem solving efficiency	yes	no	mathematical induction applied to constructive domains	s=nb of proofs* size of proofs s*log(s)

	Quality criterion	*Stability*	*Bias*	*Theoretical model*	*Complexity*
EBG (extended)	empirical success in guiding future	NR proofs	choice of primitive tactics determine	theorem proving possible proof plans	not checked
ALEX	analysis of success or failure on new	not checked	measure of resemblance between two expressions	no	not checked
MACHIN	large decrease in error rate = nb of backtrck/ nb of steps in solution	no	choice of initially given legal operators	no, under research	little inf.
ALLY	fit in theory	yes	tries to avoid biasing	theorem proving	NA
CHARADE	compactness of result Expert's advice	yes	properties of the looked for set of rules	structural matching boolean lattices	n, but dep. most on s (nb descrip)
CALM	Expert's advice and appreciation	yes, in average	explicit: adjustable parameters	Inference in majority logics	$O(s*n^3)$
REFINER	expert's advice	yes	favors frequently obs. features and conjuncts	no	does not depends on n ans s
INFER	expert's advice	yes	favors simpler combinations	no	large but unknown
MORE	performance with new examples	NR	partition of the theory	no	not checked
CLINT	user's advice	no, if 2 definit. are consistent	choosing the language	identification limit proven	not checked
INDE	performance on test + correction of theory	yes	by introducing over-general rules, rather than over-specific	no	not checked
CONCLAVE	practical correctness simplicity	yes	type of abstraction	yes: pb solving + definition of learning goal & quality	not checked
KBLA for Tonal Music	improve interaction less stupid questions. Compare to theory	yes	expl: heuristics impl: language + model of pb solving	theory of counterpoint	not yet checked
AR4, BR3	compare with human performance	yes	no	no	NA
FM	performance improv.	no, qualitity of control rules dep. of tasks	simple tasks must be presented first	EBL	not checked
ZENO 1.1	performance on test data and expert's advice	yes	simplicity	depends on algorithms	not checked
MAVERICK	performance on test data, rate of convergence, info. gain	yes	concepts given in knowledge base	Information theory Automata theory Markov theory	possible to compute not done
BG	simplicity	yes	parameters in algo. statistical confidence levels	hill climbing in space of grammars	not checked
SEA	performance on test data	yes, only minor changes due to ordering	order of input and attributes	no	$O(s*n^3)$
THUMB-RULE	simplicity	not sure	form of inputs and outputs	no	n_C=nb obj. p. class N_r=av. nb. partial rules per class $O(2^{n_C N_r})$

Algorithm descriptions

ID3-TI. 1. Grow decision tree (traditional ID3 algorithm)
2. REPEAT Prune tree leaves
UNTIL Pruning termination criteria met

CN2. Procedure CN2. REPEAT: find a good rule UNTIL no more can be found.
Procedure FIND-A-RULE:
1. Start with the general rule 'everything is <class>'
2. Perform general-to-specific beam search for better specializations
3. Return the most accurate, statistically significant rule found

CIGOL. REPEAT UNTIL terminated by user
READ EXAMPLE
GENERALIZE/ADD NEW PREDICATES
TEST AGAINST ORACLE

ASSISTANT. Modified ID3 with the following additional mechanisms. Forward pruning and post-pruning to fight noise; binarisation of attributes; selection of good learning instances (optional).

GINESYS. Similar to CN2, AQ15. Basic modifications are that it constructs redundant rules, combines with Bayes rules approximation, enables parameter specifications.

LogArt. Generate all the rules with up to 2/3 attributes. Eject bad rules. Sort the remaining rules according to their credibility.

NINA. ID3 + continuous attributes, "don't care" values, optional binarisation, attribute/attribute comparisons, post-pruning.

DUCE. See Cigol.

ALEXIS. The method contains two algorithms FM and GS. FM searches through a frame-based structure of the examples and send a selected frame in form of a table to GS. GS builds a decision tree, according to an economy criterion. FM joins the decision trees and builds a macro-tree.

SICLA. Clustering looks for a set of concepts the most linked with the original descriptors. Statistical criteria are used, they are based on inertia or chi-square criteria. The process is iterative and starts from initial stars. Discriminant methods generate step by step a decision tree. They work top-to-bottom and choose the most discriminant descriptor at each step.

MAKEY. Creation of decision graphs top-to-bottom. At each step the most discriminant descriptor is chosen, guided by background and meta-knowledge. The consequences of each node built are evaluated.

CONNEC. NR.

INSTIL. ID3 algorithm written in an object oriented language so as to use background knowledge during the building of the decision tree and being able to evaluate the consequence of chosing a particular node.

INDUCE. Based on Michalski's INDUCE, with extras
1. Handling of noisy data.
2. Dynamic type taxonomies created
3. Different search heuristics used.

ML-SMART. Top-down search guided by declarative heuristics and domain theory.

ORLON. Top-down clustering.

RIGEL. Specialization and generalization steps interleaved.

OGUST. First order generalization algorithm using background knowledge to increase the degree of matching of the examples. Goal oriented theorem proving is used to discover relevant properties.

KBG. First order generalization algorithm using background knowledge to increase the degree of matching of the examples. Forward chaining, i.e. propagation of properties is used to discover relevant properties.

DISCIPLE. The system receives examples of problem solving from its user. It over-generalizes the examples and applies this over-generalization to the data basis in order to find instances of the over-generalization. These instances are proposed to the user as tentative rules. When the user validate them, we say that they are positive examples, when s/he rejects them, we speak of a negative one. This set of positive and negative examples is used by a classical generalizer to produce a more refined generalization.

APT. Same as DISCIPLE.

BLIP. Learns by hypothesis generation, then hypothesis testing; a hypothesis is a rule-schema instantiated by predicates. Test patterns attached to each rule-schema (characteristic situations) are used to find positive and negative examples in the facts, including derived facts.
A threshold and ratio is used to decide wether to reject or to accept the hypothesis as a new rule.
EBG. Prove relevance of training example to concept. Prune proof trace by "operationality criterion". Generalize pruned proof trace.
EBGF. 1. Try each possible proof of the goal using the negative example
2. For each proof, collect all the leaves (as in EBG)
3. reduce the number of literals
4. remove negations by using determinism (inductive step)
LIFE. Construction and analysis of a dependency graph describing the effects of an action on the given situation.
EBG (extended). Match object level steps against meta-level tactics. Use back-propagation to fit meta-level tactics together into proof plan.
ALEX. NR.
MACHIN. As long as there is a problem to solve or that knowledge needs evaluation
Use knowledge to choose a solving step
Evaluate result using knowledge (subjective) and, if possible, feedback on previous experience (objective).
IF subjective evaluation is negative, or impossible, or different from objective evaluation
THEN create rule (that will need evaluation)
ALLY. NA.
CHARADE. Example and description spaces are viewed as a boolean lattice. Regularities are extracted using correspondence between these two spaces. A set of rules is built by exploration of the description space, computing regularities for each description and using rule system properties (i.e meta-knowledge) in order to constrain the exploration.
CALM. Three mechanisms build logical formulas. Expansion is the generation of candidate formulas. Selection is selection of good formulas. Compression is clustering equivalent formulas.
PLAGE. Close to branch-and-bound algorithms. Breadth first and top-down search of version space.
REFINER. Identifies distinguishing features of each concepts (in differential diagnosis), and generalizes them.
INFER. Infers mal-rules to cover gaps in input student traces.
MORE. NR.
CLINT. Receive positive examples from user.
WHILE possible DO
Ask for classification of examples generated by system
IF an example is said to be positive THEN generalize
INDE. NA.
CONCLAVE. Maximally generalize on purpose, specialize when forced to.
KBLA (Tonal Music). Integration of deductive and inductive learning under the paradigm of "learning as compilation of explanations". The system builds explanations of situations and generalizes the explanation structure. Explanation steps can be deductive (as in EBG), by analogy (vs. determinations) or weaker forms of similarity. This provides a flexible integration of deductive and inductive effects.
AR4, BR3. N R
FM. From weakest precondition WP=WP(O1, O2, ..., ON, S, G) of the sequence {O1, ..., On} where G is the goal. Generalize WP. Form WP° = {WP & On's precondition}. Specialize WP° until it discriminates against alternative instantiations of On in original proof trace.
ZENO 1.1. Based on AQ, ACLS, CN2. Significant modifications follow. in ACLS there is a different handling of numerical attributes. In CN2 there are different strategies for handling noise.
MAVERICK. Sequences of descriptions at appropriate level of generality collected in a tree. Selection of predictions based on accuracy/evidential trade-off.
BG. REPEAT
SCAN TEXT (repeatedly PARSE)
ANALYSE PARSES OF TEXT

AUGMENT GRAMMAR, on basis of statistical properties of parses plus other properties.
SEA. For each example, all discriminant generalization of maximal generality are investigated. Rules are selected according to an exception threshold to optimize performance on a test-set.
THUMBRULE. Similar to Diday's "CABRO", optimized to generate class rules (disjuncts of conjuncts of attr/val pairs) with minimum length (i.e. total number of pairs).

7 - A BRIEF DISCUSSION OF THE TABLES

We intend to prepare an extensive discussion of these tables with the help of the participation of the authors of systems. Since this work is not yet completed, we shall present here a simple sketch of it, to help the reader in his/her interpretation work.

Obviously, many of these programs are inspired by approaches stemming from statistics, numeric data analysis, or information compression. Such are the numerous improvements to ID3 and AQ. The improvements which European researchers seem to be very keen on are concerned mainly with the understandability of the results, and the use of background knowledge. This is in accordance with part of the US efforts (for example Michalski's school), but also at right angles with the more technical improvements looked for in the US. For instance, windowing or incremental use of ID3 are quite popular in the US, while they draw little attention in Europe. Conversely, we know of no US knowledge based ID3, while INSTIL and NINA are two European approaches to this problem. Similarly, there are no US equivalent to the large efforts devoted by the Turing Institute and the Ljubljana school to the understandability of the results.
An other typical feature of the European school is its tendency to develop new generalization algorithms, like ALLY, BLIP, CHARADE, CLINT, CONCLAVE, OGUST while the US school seems to be more satisfied with the algorithms it has been already developing.
When coming to multi-strategy learning, the US school seems to concentrate on merging Explanation-Based Learning and empirical learning, while the European school is more versatile as shown by ALEX, using analogy and generalization, BLIP using induction in a deductive environment, and DISCIPLE merging EBL, empirical learning and analogy.
An other important difference between the US and the European approach lies in the relatively low influence in Europe of the "EBL revolution". Except the pioneering works of the Edinburgh school led by Bundy, EBL caught quite late, and little work has been done to improve EBL except by LRI's EBGF and Edinburgh's EBG.
Finally, we have the feeling that the classifier approach, together with the genetic algorithms which have been mainly developed in the US, have been scarcely catching in Europe. MACHIN and CALM can be seen (even though their authors may strongly contest our opinion) as representative of the classifier approach in that sense that they rely very much on performance evaluation to perform their learning. Their essential difference and originality lies in the amount of work done to fit with the domain representation, instead of complying to the classical strings of bit representations.

8 - DESCRIPTION OF SOME LEARNING KNOWLEDGE ACQUISITION (KA) SYSTEMS

This information comes from the proceedings of EKAW'88 [Boose, Gaines & Linster 1988]. It shows the importance given in Europe to coupling KA and ML. Most of the US works depend on friendly interfaces rather than on ML.

6.1 - Model changes

The KA uses a model of the problem in order to ask questions to its user. This model can be improved automatically or, alternatively, through interactions with the user.

6.2 - Reliability

In KA systems, the reliability of the system becomes even more stringent than in other ML systems since the user is supposed to control the acquisition phase.

6.3 - Domain

This item tells the domain of application of the KA system.

6.4 - Method

What methodology is used or integrated in the system?

6.5 - Cooperation

This describes how the user and the system interact when they meet a problem. For instance, an inductive system can perform itself easy induction steps and rely on its user when it fails.

6.6 - Source

What is the source of the knowledge, how do you "spy" your user?

6.7 - Authors

"Esfahani": L. Esfahani, F.N. Teskey, Brighton Polytechnics
"Hoppe" : H. U. Hoppe, GMD IPSI, Darmstadt
"Someren": P.P. Terpstra, M. W. van Someren, Univ. Amsterdam
"Moller": J.U. Moller, Univ. Hamburg
"Boy": G.A. Boy, N. Nuss, ONERA Toulouse
"Sebag": M. Sebag, M. Schoenauer, Ecole Polytechnique Paris
"Tecuci": Y. Kodratoff, G. Tecuci, LRI Orsay & George Mason Univ.
"Witten": D.L. Maulsby, I.H. Witten Univ. Calgary

	Model Changes	*Reliability*	*Domain*	*Method*	*Cooperation*	*Source*
Esfahani	Intermediate concepts Consistency checking + relevant restructuration	User's advice	Engineering Planning Design	Rule checking	Exp. gives rules, gets rules	Form filling Graphical notes from experts
Hoppe	New rules	User's advice	UNIX file handling	constants ->variables Recog. task patterns	monitoring the user	Dialog protocols
Someren	Deep rules → shallow ones user's + Model refined & completed	yes + solving advice	Problem EBL + SBL	trivial stat. error propagation	Exp. gives examples, gets rules + Deep model imprv.	Sets of examples
Moller	Add knowledge	Knowldge engineer advice	Independent of	Matching context descriptions	K. eng. selects proposals, gets	Texts domain models
Boy	Problem reformulation	User's adv + expert+ questions	Intelligent Tutoring Systems	Chunking	Based on coop. System must guess user's intentions	Tutoring sessions
Sebag	No model	measured by system	Failure detection Maintenance	SBL	no cooperation system gets examples, gives rules	Incomplete arrays of parameters
Tecuci	Completion of user's model	User's advice	Problem solving	Integration EBL+SBL + analogy	Based on coop. 0th order rules -> 1st order ones	Solutions of problems + questions to user
Witten	New programs	Unknown	Graphics	generalization	Based on coop. Strong metaphor of teaching	Execution traces

Acknowledgements

A first version of this questionnaire has been set up by J. G. Ganascia and N. Helft [Ganascia and Helft 1988]. An other source of information was the first meeting of the Machine Learning Toolbox ESPRIT2 project. In that meeting were present representatives of British Aerospace, CGE Marcoussis, DCS of Aberdeen Univ., INRIA Rocquencourt, INTELLISOFT, LRI of Univ. Paris Sud, Nixdorf, Univ. Paris Dauphine. The knowledge acquisition questionnaire was elaborated during the 2nd European Knowledge Acquisition Workshop in Bonn, with the help of Ian Witten, and the participants to the workshop. Many questionnaires have been filled up during EWSL-88 held at the Turing Institute. All our discussions with Ryszard Michalski concerning the structure of ML were also very helpful. Finally, a careful proof reading by M. Manago helped to improve notably this paper.

REFERENCES

[Addis 1987] Addis T. R., "A Framework for Knowledge Elicitation", Proc. 1st European Knowledge Acquisition Workshop (EKAW-87), Sept. 87, Addis, Boose & Gaines eds, Reading Univ. 1987.

[Addis 1988] Addis T. R., "A Knowledge Organisation for Abduction", unpublished draft, 1988.

[Bareiss, Porter & Kibler 1989] Bareiss E. R., Porter B. W., Wier C. C. "Protos: An Exemplar-Based Learning Apprentice", to appear in *Machine Learning: An Artificial Intelligence Approach*, Volume 3, Y. Kodratoff, R.S. Michalski (Eds.), Morgan Kaufmann 1989.

[Boose, Gaines & Linster 1988] Boose J., Gaines B., Linster M. (eds) Proceedings of EKAW'88, Bonn June 1988, GMD-Studien 143.

[Cox 1986] Cox P.T., Pietrzykowski T., Causes for Events: Their Computation and Applications. *Proceedings of the eighth International Conference on Automated Deduction.* Oxford 1986.

[DeJong & Mooney 1986] DeJong G., Mooney R., Explanation-Based Learning: An alternative View. *Machine Learning ,1*, p.145-176. Kluwer Academic Publishers.

[Diday 1987] Diday E. "The symbolic approach in Clustering and Related Methods of Data Analysis", *INRIA, 1987.*

[Duval & Kodratoff 1989] Duval B., Kodratoff Y. "A Tool for the Management of Incomplete Theories: Reasoning about explanations" to appear in *Machine Learning, Meta-Reasoning, Logic,* P. Brazdil eds, Pitman 1989.

[Ganascia and Helft 1988] Ganascia J.-G., Helft N., "Evaluation des Systèmes d'Apprentissage", *Actes Journées Françaises sur l'Apprentissage*, E. Chouraqui editór, Cassis May 1988, CNRS Marseilles 1988, pp. 3-20.

[Genesereth and Nilsson 1985] Genesereth, M. R., Nilsson N. J., *Logical foundations of Artificial Intelligence*, Morgan Kaufmann 1985.

[Hall 1986] Hall R. J., Learning by Failing to Explain. *Proceedings of AAAI-86.* p. 568-572. Philadelphia . Los Altos, Ca: Morgan Kaufmann.

[Kibler and Langley 1988] Kibler D., Langley P., "Machine Learning is an Experimental Science", *Proc. EWSL-88*, D. Sleeman editor, Pitman London 1988.

[Kodratoff 1988] Kodratoff Y. *Introduction to Machine Learning*, Pitman 1988.

[Kodratoff and Michalski 1989] Kodratoff Y., Michalski R. S. (Eds)*Machine Learning: An Artificial Intelligence Approach, Volume III*, Morgan Kaufmann 1989.

[Kodratoff and Tecuci 1987] Y. Kodratoff and G. Tecuci, "Techniques of Design and DISCIPLE Learning Apprentice" *International Journal of Expert Systems 1*, no. 1, 1987, pp. 39-66.

[Michalski et al. 1983] R.S. Michalski, J. G. Carbonell, T. M. Mitchell (Eds.), *Machine Learning: An Artificial Intelligence Approach*, Morgan Kaufmann 1983.

[Michalski et al. 1986] R.S. Michalski, J. G. Carbonell, T. M. Mitchell (Eds.), *Machine Learning: An Artificial Intelligence Approach, Volume II*, Morgan Kaufmann 1986.

APPENDIX

Abduction. Process of completing a failure reasoning. See more details in section 3.

Algorithm. Most authors provided a short description of their algorithms. They are given in section 5, after the tables.

Analogy. A way of reasoning or of learning by comparing similarities and differences between two examples, situations, or solutions. Should be based on a base (A, B), a target (A', B'). A and A' are similar, B depends causally from A. The analogy process attempts to see how much the similarities between A and A' entail similarities between either B and B' or the way B' depends causally from A'.

Axiomatic/scheme-based learning. In both cases, the learning makes use of background knowledge. They differ by the degree of rigor in this knowledge. Axiomatic learning takes place from very rigorous (i.e. coherent and complete) knowledge, while scheme-based learning can use somewhat incoherent knowledge, and certainly does use incomplete knowledge.

Background knowledge. Knowledge about the field in which the learning is taking place. The difference knowledge intensive/knowledge poor refers to the amount of available explicit background knowledge. There exists almost no real learning systems that does not use specific knowledge about the domain in which the learning is proceeding. Nevertheless, most systems encode this knowledge in the representation used, and/or in the the way the computations are performed. In these cases, we say that the knowledge is implicit, that is these systems are knowledge poor.
Section 3 claims that the use of background knowledge does not make the difference between inductive and deductive learning, as opposed to a belief we have often met (in our opinion, the learning engine (see this word) does this difference).

Bias. Biasing amounts to make choices that are not easily tracked. As a consequence, as in the case of background knowledge, implicit biasing is a current practise in ML. The first example of explicit biasing happened as choices of nodes in a taxonomy of generality. For instance, deciding if (sine OR cosine) is a valid descendant of the node "trigonometric", that is, deciding if (sine OR cosine) is a relevant concept, is such an explicit biasing of the learning process.

Bottom-up/top-down generalization. Bottom generalization performs "real" generalizations, (see this word) while top-down generalization actually performs a particularization. See more details in section 3.

Complexity. Some authors refused to evaluate the complexity of their algorithm as a function of the number n of examples, and of their size s. They asked for other definitions of the complexity, as reflected in the tables.

Constructive learning. R. S. Michalski first defined constructive learning as combining induction and deduction (meaning 2 in the tables). We proposed other the two other definitions in order to test wether they were entailed by each other in practise since, from an intuitive point of view, "constructive" may also mean that the system uses its own learned knowledge to improve its performance. We suggest a third possible definition relying on that it uses explanations to validate the newly acquired knowledge, and using it again.

Cooperation. Describes the cooperation between a system and its human user or teacher. For instance, an inductive system can perform itself easy induction steps and rely on its user when it fails.

Domain. Describes the domain of application of the system.

Explanation-Based Generalization (EBG): given complete knowledge of a domain and one example of correct behavior, EBG will *prove* the example correctness, analyze the trace of this proof, and *generalize* it just enough to respect its sufficiency. The analysis is done in terms of a so-called *criterion of operationality* describing on which operators one has to focus one's attention. Notice that EBG does not perform second order generalization, i.e. it does not generalizes the operators themselves but only the expressions to which they apply.

Explanation-Based Learning (EBL): This approach to ML analyses one positive instance and keeps its significant features. It uses formal knowledge to prove the instance validity as in Explanation-Based Generalization but also uses informal knowledge, given as schemes of a semantic net, in order to determine the level of generality at which the operators themselves should be used.

Explanation-Based Generalization by Failure (EBGF): an EBG in which a complete theory of the possible failures has been included. When a failure is detected, the theory about it is used to recover from it.

Empirical Learning from Examples: often called "Similarity-Based Learning" because its goal is the detection of similarities between sets of positive examples (in order to find their complete Description) and between sets of negative examples (the negation of these last similarities constitute Coherent Descriptions of the positive examples). Keep in mind that inductive hypotheses thus drawn should be as much *reasonable* as possible. This reasonableness follows from background knowledge added to the examples. It follows that, contrary to a too much spread belief, the difference between Analytical Learning and Empirical Learning does not stem out of the quantity of background knowledge they use but from the way they use it. Analytical Learning uses it for the detection of intra-example properties while Empirical Learning uses it for the detection of inter-example properties.

Empirical Learning from Observation: similar to Empirical Learning from Examples except that the examples are provided by the system itself. It may happen that a strong example generator may then compensate for a weaker system of Empirical Learning. For instance, it is often the case that background knowledge is left implicit in the examples generator, and the empirical learning proceeds without it.

Empirical learning. Inductive learning performed without using background knowledge. A typical example of it follows. You see Bill drinking gin and water, he gets drunk. You see Peter drinking scotch and water, he gets drunk. You conclude that when "x" drinks water (the common pre-condition in both stories) then "x" gets drunk, forgetting about the knowledge dependant relation between gin and scotch, i.e. that they are alcoholic beverages.
With all its imperfections, empirical learning can be very useful as a quick first "glance" at possible relationships.

Examples. Systems learn often from examples. When these examples illustrate a behavior described as correct by the professor, they are called positive examples. When they illustrate a behavior described as incorrect by the professor, they are called negative examples.
When the system is able to generate itself new examples, they can be either positive and negative. Experimentation and/or expert interrogation will classify them.

Generalization: Extending the scope of a concept description to include more instances (the opposite of specialization). This term is sometimes used as a noun, synonymous with concept description. Provides an intentional description of a concept, as opposed to conceptual clustering that provides an extensional description of a concept.

Incrementality. Induction systems often learn from one given set of examples. Adding a new example asks for starting over again from scratch. Incrementality is achieved when the system is able to evolve smoothly when new examples are added.

Inductive learning of decision trees. Known as "ID3" from its first implementation in the ML sub-field. Builds a decision tree on criteria that optimize both the entropy and some understandability of the tree itself.

Inputs. The inputs of a ML program may take many different forms, like attributes pairs, example descriptions etc ... and can represent many different kinds of knowledge, like rules, conditions, etc ...

Knowledge acquisition (KA). Body of techniques intending to help an expert to express his/her knowledge under a form usable by a system expert. It goes from a simple window management to elaborated interrogation forms. Most KA systems do not use machine learning at all since their goal is collecting information from a human expert, but not asking the machine itself to acquire knowledge. In other words, one can say the KA systems are intended to perform clever rote learning. Nevertheless, it is clear that if some learning is performed by the system itself then it can avoid asking tedious questions to the user, and/or it can ask real clever questions. Therefore, ML is becoming a strong component of KA, without being merged in it.

Knowledge Intensive Induction. Induction making explicit use of background knowledge, see more explanations in section 3.

Knowledge Intensive/knowledge poor. See Background knowledge.

Learning engine. Inductive learning makes use of induction as its main learning engine, while deductive learning (and explanation)-based learning also) make use of deduction. In section 3, we explain that in many cases, evaluation of the program performances is a learning engine as well.

Learning from examples/observations. Since observations can always be seen as kind of examples, the only real difference between these two kinds of learning lie in the way examples are presented. If they are already clustered, and the concept to learn is already described (i.e. one learns a new description) then one speaks of learning from examples. One speaks of learning from observations in all other cases. In the last case, the system must be able to create its own clusters of examples.

Meta-knowledge. Meta-knowledge contains rules that state properties of the knowledge itself. For instance, if knowledge is made of rules, then meta-rules state relations among the rules, or strategies for using the rules.

Model changes The KA uses a model of the problem in order to ask questions to its user. This model can be improved automatically or, alternatively, through interactions with the user.

Oracle. An oracle is a the answer of a human interacting with the system by answering selected questions asked by the system. The most human dependant systems are called interactive systems. One usually speaks of oracles when the interaction is minimal in some sense.

Pre-clustering. Examples can be ordered prior to learning, or, alternately, the system may accept unordered examples. In the tables, a "yes" means that the relevant system needs pre-clustering, while a "no" means that the system is able to accept unordered examples.

Quality criterion. They are the criteria to judge if a program has been behaving "well" or "badly". Such criteria, or some of them, can be internal to the system that will decide itself of the quality of its results. For instance, the depth of a decision tree can be such an internal criterion. It also often happens that external criteria are needed, ranging from an evaluation of an expert to a testing against new data.

Stability. It is not always the case that a program stable under learning twice the same data, in a different order, for instance. A learning program may well be very reliable even though it is not perfectly stable. Knowing its range and type of instability is part of the characterization of a ML program.

What is learned? One could believe that knowing the input and the output of a ML program is enough to know what is given to, and then learned by the system. This belief is wrong since a learning program can perfectly well have the same kind of inputs and outputs, and nevertheless perform learning by improving the inputs in some sense.