DISCTPLE-1: INTERACTIVE APPRENTICE SYSTEM IN WEAK THEORY FIELDS

YVES KODRATOFF
LRI, Inference et Apprentissage, Bit. 490, U.A. 410 du C.N.R.S. & University Paris-Sud, F - 91405 Orsay

OHEOROHE TCCUQ
Research Institute for Computers and Informatics, 71316, Bd. Miciurin 8-10, Sector 1, Bucharest, Romania

ABSTRACT

The paper presents an interactive approach to learning apprentice systems for weak theory domains. The approach consists of a combination of teaming by analogy and learning by generalizing instances. One main point of this approach is that it uses the explanations drawn from an example, both to reduce the version space of me rules to be learned, and to generate new examples, analogous to the given one. Another important point is that it demonstrates not only that over-generalization is harmless but also useful and necessary, when interacting with a user. It allows to use the theory of the domain, though incomplete as it is, in order to extract the missing knowledge by asking "clever" questions to its user. This paper presents a first prototypical version of DISCIPLE and its use to the design of technologies for the manufacturing of loudspeakers.

I INTRODUCTION

If Expert Systems have proven useful in many domains, their applications are limited by their inability to acquire and to update their knowledge. This problem is largely recognized as the *knowledge acquisition bottleneck* of Expert Systems (Feigenbaum 1977), (Mitchell & Al. 1985), (Kodratoff 1986), etc... Recent Machine Learning achievements ((Mitchell, Carbonell & Michalski 1985) offer new solutions to the knowledge acquisition problem and open a new area in the evolution of Expert Systems, that is, Expert Systems able of automatic knowledge acquisition and learning, such as Learning Apprentice Systems (LAS). A LAS is an interactive knowledge-based consultant that directly assimilates new knowledge by observing, analyzing and questioning about the problem solving steps contributed by their users through their normal use of the system. The user gives to the system a problem to solve and the expert sub-system starts solving this problem by showing the user all the problem solving steps. The user may agree or reject them. Therefore, in its Expert System mode, a LAS may encounter two situations.

Either the current problem-solving step (which we shall further call partial solution) is accepted by the user. Then, the current state of the knowledge base is judged as satisfactory, and no learning will take place.

Or it is unable to propose any partial solution (or the solution it proposes is rejected by the user). Then, the user is compelled to give his own solution.

Once this solution is given, a learning process will take place. The LAS will try to learn a general rule so that, when faced with problems *similar* to the current one (which it has been unable to solve), it will become able to propose a solution *simitar* to the solution given by the user to the current problem. We are developing a LAS, called DISCIPLE, specialized for weak theory domains. In this paper we describe the learning mechanisms of DISCIPLE. To this purpose we use examples from Technology Design. The next section is a brief description of this domain. The following sections present the learning problem and the learning method of DISCIPLE

rj A WEAK THEORY DOMAIN

We have chosen, as a first domain to test our approach to interactive LASs, the domain of designing technologies for the manufacturing of loudspeakers. Before presenting in more details this domain we stress two of its important features. Firstly, the domain is usually too complex for an autonomous system. Secondly, small improvements in technology have important outcomes since a technology is usually used for a large number of products. Therefore the *best* solution is searched. A consequence of these features is that such a domain is most appropriately handled with an interactive system as the expert (consultant) sub-system of a LAS, where the user and the system cooperate in finding the best solution to the current problem.

Technology Design might well be viewed as successive decompositions of complex operations into simpler ones, and successive specializations of these simpler operations by choosing tools, materials or verifiers, which are in turn successively specialized. To design a technology, DISCIPLE needs some knowledge about the components of the loudspeakers, about the technological solutions for the manufacturing of loudspeakers, about the tools and the materials one can use to manufacture loudspeakers. All this knowledge constitutes the domain theory. This domain theory is inherently incomplete since we can not suppose that DISCIPLE knows all the objects of the domain, all the properties of a given object, all the actions that can be performed for manufacturing loudspeakers, all the properties of the known actions (preconditions, effects), all the ways of decomposing or specializing a given action, etc...

III THE LEARNING PROBLEM

In the domain we have chosen, DISCIPLE acts as an aid to a technology designer. The problem to be solved is that of planning the manufacturing of a certain loudspeaker. The solution to this problem is a plan of actions for manufacturing the loudspeaker.

The problem-solving paradigm is problem-reduction. That is, DISCIPLE will successively decompose an action into simpler actions or specialize an action to a better defined one. In this way, DISCIPLE will build a problem-solving tree. This process continues untill the leaves of this tree are elementary actions. They represent the solution to the original problem (the top of the problem-solving tree).

Let us suppose that, during planning the manufacturing of a loudspeak er, DISCIPLE encounters the following problem

ATTACH sectors ON chassis-membrane-assembly

for which it is unable to propose a satisfactory solution. Let us further suppose that the user indicated the following solution to DISCIPLE:

APPLY mowicoll ON sectors,
PRESS sectors ON chassis-membrane-assembly

Note that APPLY and PRESS may be actions previously unkown to the system and, in such a case, it knows nothing about them except that the are means of ATTACHing.
Now DISCIPLE knows a solution of the current problem

ATTACH sectors ON chassis-membrane-assembly
⊢ APPLY mowicoll ON sectors,
PRESS sectors ON chassis-membrane-assembly.

This solution will further be seen as an example of a general rule to be learned.

In this paper (and in the present version of DISCIPLE) we use the following generalization method. We suppose that the above example represents with fidelity the structure of the general rule to be learned. In this case, learning a general rule reduces to learn the concepts (instantiated in the example by 'sectors', 'chassis-membrane-assembly' and 'mowicoll) for which the ATTACH action can be safely decomposed into a sequence of APPLY and PRESS actions.
Therefore, the rule DISCIPLE will try to learn has the following form:

IF x. y, and z satisfy <constraints>
THEN ATTACH x ON y \- APPLY z ON x, PRESS x ON y

The learning problem addressed by DISCIPLE is therefore : Given an example rule, generalize it and find its domain of application.

IV THE LEARNING METHOD

DISCIPLE learning method involves a combination of Explanation-**Based, Similarity-Based, and Analogy-Based algorithms.**

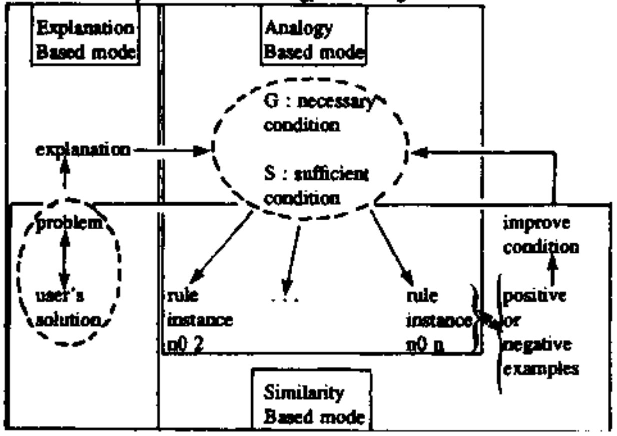

Figure 1. The learning method in DISCIPLE

The learning starts by interpreting a user's solution as an instance of a more general rule to be learned.

Firstly, in its Explanation-Based mode, DISCIPLE looks for plausible explanations of the validity of the user's solution. It is essential to the success of DISCIPLE that there should be a possible explanation in terms of the relations between the objects referred at in the example. One must nevertheless be well aware that, working in a weak theory domain, those explanations may sometimes be irrelevant (as opposed to the ones provided in well-formalized domain), and have to be validated by the user.

Secondly, DISCIPLE enters its Analogy-Based mode. The analogy relies on the concept of similarity of the explanations : two rules are analogous when they are supported by similar explanations, and two explanations are similar when they both are instances of the same (over)-generalized explanation. In its Analogy-Based mode, the work of DISCIPLE is two-fold. On the one hand it attempts to build such an over-generalization from the examples it has got. On the other hand, it consults its knowledge base in order to generate new instances of the current over-generalization. We shall use the following approximation : the explanation of the user's example will be said to be a sufficient condition to the rule application, and the over-generalization will be said to be a necessary condition to the rule application. In that way, they can be compared to the S-set and G-set of Mitchell's Version Space (Mitchell 1978).

Thirdly, DISCIPLE uses Similarity-Based Generalization from examples. Analogy does not garantee the validity of the generated conditions, which must once more be validated by the user. The generated examples rejected by the user will be treated as negative examples, and the accepted ones as positive examples. In the same way as other generalization algorithms (see, for instance (Michalski 1983, Kodratoff & Al. 1984)), positive examples will be used to generalize the set of sufficient conditions (otherwise stated : to increase the S-set). Negative ones will be used to particularize the necessary conditions. This part of DISCIPLE has not yet be fully worked out, it will not be further described here.

Finally, if it happens that the S-set and the G-set become identical, a necessary and sufficient condition has been reached , and an exact rule has been learned. This is seldom the case, especially in fields with a weak theory. In general, we shall keep the necessary and the sufficient conditions separately. We then say that we have obtained a SYMBOLIC UNCERTAIN CONDITION for the application of the rule.

V EXPLANATION-BASED MODE

The system will try to explain why the solution indicated by the user is a good one. Since DISCIPLE does not have a complete domain theory, it is unable to find alone such a "complete" explanation. Recall, for instance, that APPLY may be an action previously unknown to the system. It does not mean that DISCIPLE is waiting for an explanation from the user, but simply that it will try to find an explanation with the user's help. More precisly, it will try to propose several partial explanations, asking the user to validate them.
The heuristic used by DISCIPLE is that the explanation has to be expressible in terms of the relations between the objects from the rule instance ('sectors', 'chassis-membrane-assembly', 'mowicoll'). While, in general, there exist many relations between two objects, it is expected that, in the world of an Expert System, only the relations relevant to the domain of expertise are present.
Therefore, DISCIPLE will look in its knowledge base for the links connecting 'sectors', 'chassis-membrane-assembly' and 'mowicoll'. They are illustrated in the following netoork:

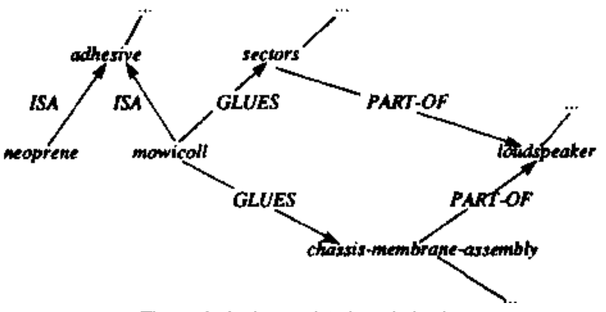

Figure 2. An incomplete knowledge base

Some of these links may be relevant for the rule to be learned when they are plausible pieces of explanation. Since DISCIPLE does not have the necessary knowledge to make the difference between the relevant and the irrelevant links, it will have to rely on the user, by asking questions which are issued from a straithforward analysis of these links: DISCIPLE will initiate the following dialogue (where the user's answers are put between *).

Is it relevant for the solution that:
mowicoll GLUES sectors ? * yes *
mowicoll GLUES chassis-membrane-assembly ? * yes *
sectors PART-OF loudspeaker ? * no*
chassis-membrane-assembly PART-OF loudspeaker ? * no *

Making use of this new knowledge, DISCIPLE will find an "explanation" represented by the following network.

$$\begin{tikzcd} & \textit{sectors} \\ \textit{mowicoll} \arrow[ur, "GLUES"] \arrow[dr, "GLUES"'] & \\ & \textit{chassis-membrane-assembly} \end{tikzcd}$$

DISCIPLE will have to consider all such networks as candidates for explanation. This process may become somewhat cumbersome. Nevertheless, it is much easier for the user to agree or deny a partial explanation, than to give oneself an explanation, which should also be understandable for the system.

VI ANALOGY-BASED MODE

Two situations are said to be analogous when a mapping can be done between the causal networks of these two situations (Wiaston 1980, Kedar-Cabelli 1985). The above network will be seen as a causal network of the decomposition in example rule. A heuristic used by DISCIPLE to find similar explanations is: two explanations are similar when the edges of their networks are indexed by the same values.
For instance, the following network is similar with the above one,

$$\begin{tikzcd} & \textit{centering-device} \\ \textit{neoprene} \arrow[ur, "GLUES"] \arrow[dr, "GLUES"'] & \\ & \textit{chassis-assembly} \end{tikzcd}$$

DISCIPLE uses the similarity between networks, and proposes instantiated rules to its user. These rules are obtained by replacing the explanation pattern of the example rule by similar ones.
For instance, DISCIPLE will propose the following new rule.

ATTACH centering-device ON chassis-assembly
h *APPLY neoprene ON centering-device,*
PRESS centering-device ON chassis-assembly

Since the analogy is never proven to be valid, the user will have to validate it One can estimate the similarity between two networks, or concepts, by computing their "best generalization" (Kodratoff & Tecuci 1986b). The less general is their generalization, the best their similarity. In DISCIPLE, we slightly modified the approach referred at in (Kodratoff & Tecuci 1986b). Instead of starting a costly generalization algorithm of the AGAPE kind (Kodratoff & Al. 1984), DISCIPLE rather over-generalizes the explanation it disposes of, and states that any two instances of this over-generalization are similar. In DISCIPLE'S present implementation state, this over-generalization is very elementary : turn constants into variables by giving the same variable name to all the occurences of a same constant. It is part of our planned improvements to DISCIPLE to refine this definition. We are quite aware that such a generalization may not be a necessary condition. This is part of the approximations of the present system to accept this lack of precision.
It follows that the over-generalization of the above networks is

$$\begin{tikzcd} & x \\ z \arrow[ur, "GLUES"] \arrow[dr, "GLUES"'] & \\ & y \end{tikzcd}$$

which will also be used as a necessary condition for the application of the rule.
Any example DISCIPLE will generate in the Analogy-Based mode will have an explanation which is an instance of the over-generalized expression.
At this point the learned rule has the following form

```
IF   {necessary condition}  ((z GLUES x)&(z GLUES y))
     {sufficient condition}  ((x ISA sector)&(y ISA chassis-membrane-
                              assembly)&(z ISA mowicoll)&
                              (z GLUES x)&(z GLUES y))
THEN    ATTACH x ON y   ⊢   APPLY z ON x,  PRESS x ON y.
```

which is an intermediary form of the rule during its learning. One remarks that the necessary condition can always be deduced from the sufficient one.
In the Analogy-Based mode, DISCIPLE will generate instances of the intermediate rule, such that these instance will satisfy the necessary condition (and that will not satisfy the sufficient condition).

VII DISCUSSION AND CONCLUSIONS

In the previous sections we have presented in some detail the learning mechanisms of DISCIPLE. It shares, of course, many features with LEAP (Mitchell & Al. 1985), both relying on the same design principles: the interactive nature of problem solving, the association of each example to a single problem solving step, the partition of control and basic domain knowledge. On the other hand there are also important differences between LEAP and DISCIPLE. LEAP utilizes explanation-based generalization. Therefore, it produces justifiable generalization from a single example, it allows rejecting incorrect training examples, it relies on a strong domain theory (VLSI design). DISCIPLE utilizes a combination of Explanation-Based learning, learning by analogy, and Similarity-Based learning. It relies on an incomplete and/or weak domain theory, it relies on user to reject incorrect training instances, it produces justifiable generalizations from examples. DISCIPLE learns not only generalizations but also particularizations of concepts (DeJong & Mooney 1986). Also, it uses the same interface paradigm for both problem-solving and learning (it proposes solutions and the user accepts or rejects them).

There are several weaknesses of DISCIPLE, that are currently under improvement. The method of finding an explanation is not powertull enough. Other sources of knowledge are needed, as well as metarules for finding far-off explanations. The explanation-based method does not use goal regression. The use of theorems in the Analogy-Based mode is quite limited. From the practical point of view, DISCIPLE shows one more very important weakness. Its analogy mechanism works through our over-generalization process, which actually reduces to turning constants into variables. It follows that DIISCIPLE is of interest if, and mainly only if, the given rules are such that the same constants show in their conditions and in their actions.

DISCIPLE-1 has been implemented in LELISP (Chailloux 1985) and we are running it on VAX-750, and MACINTOSH computers. Implementations on SUN and Explorer stations are under way.

REFERENCES

1 Chailloux J., "LE-LISP de 11NRIA, Le Manuel de reference", I.N.R.I.A., Rocquencourt, France, 1985.
2 DeJong G., Mooney R., "Explanation-Based Learning: An Alternative View", Machine Uarning 12 (1986) 145-176.
3 Feigenbaum E., "The Art of Artificial Intelligence" In Proc. UCAI-77, MIT, Cambridge, Massachusetts, 1977, pp. 1015-1029.
4 Kedar-Cabelli S. "Purpose-Directed Analogy", Research Report ML-TR-1, Rutgers University. 1985.
5 Kodratoff Y., Ganascia J.G., Clavieras B., Bollinger T., Tecuci G., "Careful Generalization for Concept Learning" In Proc. ECAI-84, Pisa 1984, pp. 483-492. Also In Advances in Artificial Intelligence T. O'Shea editor, pp. 229-238, North-Holland Amsterdam 1985.
6 Kodratoff Y., "Learning Expert Knowledge and Theorem Proving" In Proc. GWAI-86, Springer-Verlag 1986, pp. 164-179.
7 Kodratoff Y., Tecuci G, "DISCIPLE: An Interactive Approach to Learning Apprentice Systems", LRI Research Report 293, Orsay, France, Aug. 1986.
8 Kodratoff Y., Tecuci G., "Conceptual Distance-Based Learning", LRI Research Report 299, Orsay, France, Sept. 1986.
9 Michalski R., "A Theory and a Methodology of Inductive Learning" Artificial Intelligence 20 (1983) 111-161.
10 Mitchell T.M., Carbonell J.G., Michalski R.S. (eds). Machine Learning: A Guide to Current Research, Kluwer Academic Publishers, 1985.
11 Mitchell T.M., "Version Spaces: An Approach to Concept Learning", PhD thesis. Department of Electrical Engineering, Stanford University, 1978.
12 Mitchell T., Mahadevan S.. Steinberg L., "LEAP: a Learning Apprentice System for VLSI Design" In Proc. UCAI-85, Los Angeles 1985, pp. 573-580.
13 Winston P. "Learning and Reasoning by Analogy", Com. ACM 23 (1980)689-703.