THE REFORMULATION APPROACH TO BUILDING EXPERT SYSTEMS*

William S. Mark"
MIT Laboratory for Computer Science

Abstract

Many of the tasks that go into the building of an expert system — collecting the expert knowledge, setting it up for efficient problem-solving, providing mechanisms for acquisition and explanation — are structured by the choice of organization for the system's knowledge base. This paper discusses one such organization and the implementation approach it entails. This approach has been used to produce a business consultant program.

Introduction

Recently, a considerable amount of research has been directed toward the development of programs which provide problem-solving expertise to the user ([Brown, et al. 1, [Buchanan, et al.], [Davis], [Hart], [Martin], [Pople], [Rubin], [Shortliffe], [Sussman]). The methods for building these programs differ mainly in the way they trade off the ease with which knowledge can be acquired and explained against the efficiency with which it can be used to solve problems. The approach taken here explores a different trade-off: concentrating on a representation for the expert knowledge which is at the same time easy to build and efficient to run, at the cost of increasing the complexity of the process which applies this expert knowledge to any particular problem.

All approaches to building expert systems are essentially mechanisms for applying general knowledge about a class of problems to specific examples of that class. In existing systems, particularization of knowledge is part of problem-solving. For example, in rule-based systems ([Davis], [Shortliffe]), the process of deciding whether an expert rule is applicable to a problem is part of the process which uses that rule to help solve the problem. When some state of knowledge about the problem matches the pattern of a rule, the rule Is automatically applied. This creates a new state which will be used for further rule application — a process which continues until a desired solution state is reached. Note that this method requires that the same knowledge representation be good for solving problems and for enabling the program to see when knowledge is applicable. Frame systems [Rubin] have a similar property.

*This research was supported by the Advanced Research Project Agency of the Department of Defense, and was monitored by the Office of Naval Research under Contract Number N00014-75-C-0661.

Current Address
Computer Science Department
Research Laboratories
General Motors Technical Center
barren, Michigan 48090

In the approach presented here, application of knowledge is separated from problem-solving. Knowledge representations and the problem-solving procedures which use them are tailored specifically to solving problems in the domain of interest. This allows them to be easy to build and explain, because the expert's concepts and procedures can retain their real world interrelationship within the program. At the same time, this problem-solving knowledge can be efficient to use because the system builder can take advantage of algorithms which are specifically designed for those interrelationships. All of this is at the cost of providing a separate, relatively sophisticated procedure to find a mapping between the specialized expert problem-solving knowledge structures and the (presumably very different) input problem structures.

This approach has been used to Implement an expert system which uses a cause-effect flow model of the firm ([Forrester], [Gorry]) to solve work force control problems presented as one page business cases [Mark],

Definition of Reformulation

The basis of the reformulation approach is the following model of expertise:

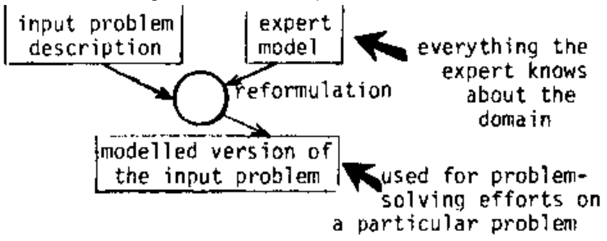
input problem description
expert model
everything the expert knows about the domain
reformulation
modelled version of the input problem
used for problem-solving efforts on a particular problem

Figure 1. The reformulation mode] of expertise

The expert knowledge is in the form of a self-contained and self-consistent model of a class of problems (and solution methods) in the domain of interest. The client describes problems in "naive" terms — i.e., in a form that is certainly not based on, and in fact may vary widely from, the expert's formalism for describing things in the domain. The expert's task is to understand what the client is saying in terms of the self-contained, self-consistent model: to reformulate the naive description into something which can be handled by the expert problem-solving methods of the model. That is, the expert creates a modelled version of the input description which preserves the special representation and consistency of the general expert model, but which is particularized to the problem at hand. The expert's specialized problem-solving procedures, which take advantage of that representation and that consistency, can then be used on the modelled version in order to solve the client's problem.

As an example of this process, consider the case of a business consultant who views the firm as an interconnection of regulated flows (of goods, personnel, money, etc.). The consultant solves problems by (1) diagnosing the client's complaints as being caused by one or more of the various kinds of flow problems he or she knows about, and (2) suggesting changes in the regulation of certain flows in order to ameliorate these problems (see [Forrester] and [Corry] for examples of this basic approach). Thus, the consultant reformulates the business situation into a modelled version which is in terms of the flow model so that flow-based diagnostic and solution procedures can be used. However, the modelled version also incorporates all of the essential characteristics of the client's firm so that the diagnosis and solution make sense for that firm.

Reformulation, then, is the process of constructing the mapping between the client's naive problem description and the specialized expert's knowledge.

Reformulation as an Implementation Methodology

The importance of the reformulation model shown in Figure 1 is that it can be used to define a basic method for organizing expert knowledge in programs:

(1) The concept structures of the expert knowledge base are chosen with reference only to the dictates of the expert model. For example, for the cause-effect flow models of the business consultant, the concept structures look like

```
(CI)   (DETERMTNED-BY  (EMPLOYEES)
                (INFLUX    (ACT-ON  (PEOPLE)
                                    (HIRE)))
                (OUTFLUX   (ACT-ON  (EMPLOYES)
                                    (FIRE)))),
```

i.e., "the number of employes is determined by the number of new hires coming in and ex-employes going out," In a database system a possible concept structure would be

```
(C2)   (ACCESS PROJECT
              (CHARACTERISTICS
                     (JOB-NUMBER)
                     (JOB-TYPE))
              (OPTIONAL-CHARACTERISTICS
                     (DATE-SPECIFICATION))),
```

i.e., "in order to find a particular project, its job number and type must be specified, and some date information may be specified". The point is that in choosing structures, the expert need only be concerned with the characteristics of his expert knowledge, not with the characteristics of the system which will apply that knowledge.

(2) All of the program's knowledge about a concept is stored with that concept and nowhere else. The fact that (HIRE) is an action which operates on (PEOPLE) according to a certain policy is found under the concept (HIRE) — never as part of some problem-solving procedure that "knows about" (HIRE). Moreover, all of the policies that the program knows about for controlling hiring are found under (HIRE).

(3) Problem-solving procedures are constructed so as to use the given structures efficiently. For example, the business consultant program contains specialized procedures for tracing cause-effect relationships through flow structures represented as in (CI). The database program could have a specialized procedure for determining the best access path to objects described as in (C2).

(4) Knowledge about how a given expert concept structure applies to a particular problem is not stored with the structure. All decisions about how to apply a given piece of expert knowledge are made by the map construction process; e.g., the choice of a specific hiring policy to model (HIRE) in a particular firm, or the decision not to model (HIRE) at all, are the responsibility of the map construction process.

(5) For the purposes of map construction, expert concept structures are considered to be patterns which must be matched with input structures. The mapping procedure itself consists of transformations which may be applied to these patterns to produce new patterns which are closer in form to the input structure under consideration (see Figure 3). One kind of transformation allows the substitution of more specific patterns for more general ones. The program would use this kind of transformation to change the (HIRE) concept in (CI) into a pattern which could match a particular firm's hiring policy. Another kind of transformation can be used to delete the (HIRE) concept — and the concepts that depend on it — from (CI) to reflect the fact that no match is expected at all in a particular problem. The kinds of pattern transformations available to the program are listed and discussed below.

(6) The procedure which decides when and where to apply these transformations is controlled by (in a sense, parameterized by) information derived from the patterns that have already been successfully transformed and matched to input structures. These matched patterns are contained in the modelled version of the problem at hand (see Figure 2.). For example, . (see Figures 2 and 3), the fact that (CI) has been expanded to model a particular hiring

policy which separates employes into SALARIED and UNION will be noticed by the selection procedure and used to aid the choice of viable transformations later in the problem. Specifically, as the modelling effort continues — and where that much detail is necessary — only transformations which preserve the distinction between SALARIED and UNION will be allowed. Deciding how detailed a pattern to present at given point is also part of the transformation selection mechanism, as we will see later.

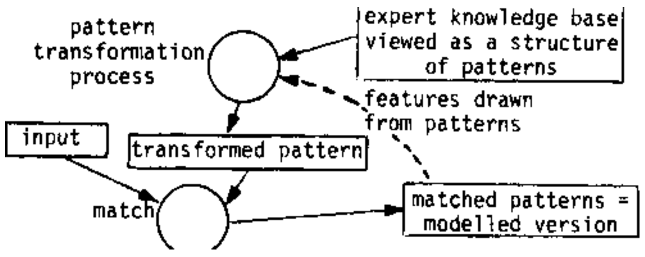
pattern transformation process
expert knowledge base viewed as a structure of patterns
features drawn from patterns
input
transformed pattern
match
matched patterns = modelled version

Figure 2. Reformulation as implementation

In summary, a reformulation system works by mapping input problem descriptions into the concepts of its expert model. It does this via a kind of pattern-matching in which the expert concept structures, viewed as patterns, are put through a series of transformations in order to make them closer to the input forms. The particular transformations and the patterns to which they are applied are chosen by a selection procedure which is parameterized by information drawn from already matched patterns.

Advantages of this Approach

Before going into the details of how this methodology works, I will discuss its advantages as an approach to building expert systems. First of all, notice that the system's mode of operation is to try to make everything it is given into something that can be handled by its particular expert model. Although this may seem a little severe, it is really just an acceptance of reality, and perhaps an advantage. In the foreseeable future, expert systems will only be able to do a few things, will know relatively little about relatively restricted domains. If the system cannot map an input problem (a request to a database system, a business consulting situation) into one of the things It knows about, it will have to give up on the problem. Therefore, it is reasonable to construct systems which work by trying to channel all input into the few things they know how to do.

This is the reality — the advantage is that this allows the system to always work from its knowledge base of comparatively few, well-formed expert concept structures. Only transformations which originate from these structures can constitute legal mappings. In the well-structured models assumed here, concepts will be basically canonical: they will not overlap unnecessarily. The possible ways to manipulate concept structures which result in new legal concept structures will be few and well understood. Thus, at any point in the mapping effort, the transformation selection procedure will have a very restricted set of root structures from which to begin transformation chains. Since relatively few transformations result in new legal structures, the set of possible transformations at any given time will also be restricted. Furthermore, the system does not depend on the natural parsimony of the model; it provides its own restriction mechanism based on information gathered from the mapping effort so far.

This restriction mechanism is a key feature of the reformulation approach. It is used to make pattern-matching efficient. Remember that in this approach patterns are equated with concept structures, not with subgoals as in other systems. At each stage in the pattern transformation series, the program can look at the concept structures it has already matched in order to see what features of the problem restrict the choice of transformations that can be applied. We will see how this restriction mechanism works in the next section. T mention it now because it is an advantage of the approach: it provides a mechanism for using search limiting constraints at each stage of the pattern-matching process. This is quite distinct from approaches used in other expert systems, which rely on extensive search or sophisticated failure backup schemes in their pattern-matching effort. In reformulation, low yield searching is greatly reduced; backup is limited to rare garden-path situations (see [Mark]).

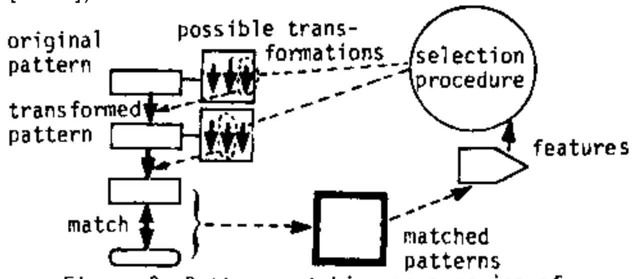

Figure 3. Pattern-matching as a series of transformations

Another advantage of the reformulation approach is that the information attached to each concept structure localizes all of the system's knowledge that is relevant to that structure. This attached knowledge is not cluttered with control structure information. Therefore, the only way in which two concept structures can be interdependent is if one mentions the other at top level. For example, in (CI) we can see that the concept (EMPLOYES) depends on the INFLUX and OUTFLUX structures in a way defined by DETERMINED-BY. We can be confident that no other interdependency information is hidden away in a separate procedure or under some other unmentioned concept.

Finally, the control mechanism of steps (5) and (6) above allows the system to deal with a number of issues which arise whenever a realistic amount of expert knowledge must be applied to a problem. For any given problem, only parts of the expert knowledge base will be relevant.

Furthermore, in many domains the level of detail at which these parts are relevant will vary from problem to problem. For example, in the business consultant system, the labor sector of the firm is usually modelled in detail, while the production sector is merely sketched. However, if details of the production process (quality control problems, seasonality, etc.) directly directly affect labor needs in a particular problem, the production sector must be modelled much more thoroughly.

The problem description alone cannot determine what the expert should model: some parts of the client's recital may be irrelevant to what the expert needs to know. The expert program mist therefore sift through the input to find the things it needs. it must ask the client questions about things it needs but cannot find.

In the reformulation approach, the mapping mechanism can be used to strike a balance between what the program wants to know and what the client is trying to tell it. On the one hand, each mapping is firmly rooted in the expert model, since it must consist of a chain of transformations which begins with an original expert concept structure. On the other hand, the mapping mechanism is specialized to each individual problem, because the transformation selection procedure is controlled by information taken from the program's current model of that problem (see Figure 3).

To achieve this balance, the mapping mechanism is used in the following way on each problem:

(1) The client's basic symptom is defined by matching a symptom pattern in the model. E.g., input of the form "The Dominion Co. often hires or lays off workforce ..." maps into the symptom.

```
(C3) (STATE-OF (NUMBER-OF WORKFORCE))
               (FLUCTUATING)).
```

(2) That symptom is used to select those parts of the expert model which are relevant to the solution of the client's problem. E.g., symptom (C3) is attached to a part of the +model which deals with damping problems in workforce flows. When (C3) is matched, that part will be selected as relevant.

(3) Feedback from the modelled version of the problem is used to see how those relevant parts should be applied to the particular problem at hand. E. g., the fact that the Dominion Co.'s hiring policy contains no long delays would restrict the class of damping problems that are possible in this firm.

Note that (1) uses the problem description and the expert model to define the <u>expert's</u> view of the problem, (2) makes sure that all transformations must be rooted in the expert's idea of what is relevant, and (3) uses the client's description to ensure that the model is correctly particularized to the problem. Level of detail considerations are handled by starting each part selected in (2) with a default level which can be altered by information from (3).

The next section describes in more detail the way in which the mapping mechanism effects this system behavior. Tn particular, it discusses the kinds of transformations that can be applied to expert patterns and the kinds of feedback information that can be used to choose these transformations•

<u>Details of the Map Construction Process</u>

Concept structures, or patterns, are made up of two kinds of elements: <u>concepts</u>, like (PEOPLE) or (JOB-NUMBER), and <u>functional concepts</u>, like DETERMINED-BY, INFLUX, OUTFLUX, and ACT-ON in (CI), and ACCESS, CHARACTERISTICS, and OPTTONAL-CHARACTERTSTTCS in (C2). A concept is anything that the expert chooses to name as a coherently describable entity. A functional concept groups concepts to convey a special meaning to the expert model (like a flow or an access specification). Functional concepts are not matched against the input. They do not represent the expert's point of view of something the client might say, but rather an internal view of how concepts are structured. There is no restriction on concepts or functional concepts (except that they must be identified as such), since the intent is that they be chosen to be convenient for the concept structures of any given expert model.

The transformations that can be applied to patterns represented in this form are:

<u>structural transformations</u>: Patterns can be rearranged according to rules attached to functional concepts. This usually means that a new functional concept will be involved. E.

```
(DETERMINED-BY a          (FLOW b a c)
      (INFLUX b)
      (OUTFLUX c))

(SHOW-USER e f g)         (DISPLAY-PLOT 3-D
                            (X-UNIT e)
                            (Y-UNIT f)
                            (Z-UNIT g)).
```

The idea is to express different points of view of the same set of concepts: sometimes it is important to know that a_, b^, and <u>c</u> are part of the same flow, and sometimes it i6 important to know that a_ is determined by b^ and c_; sometimes the system only needs to know that the user has seen e_, £_, and £, and other times it must know the exact display format.

<u>elaborations</u>: Sub-patterns (including individual concepts) can be substituted for concepts within a pattern. E.g., a particular hiring policy can be substituted for (HIRE) in (CI); ((MONTH-NAME x)(DAY-NUMBER y)(FOUR-DIGIT-YEAR z)) can be substituted for (DATE-SPECIFICATIONS) in (C2).

deletions: Sub-patterns can be removed from the pattern according to rules attached to functional concepts. E.g.,

```
(OUTFLUX (ACT-ON (EMPLOYES)
                 (FIRE)))
```

can be deleted from (CI) if necessary because DETERMINEIVBY is still well-formed with only one argument.

instantiations; Client-level descriptors can be substituted for concepts or sub-patterns. This corresponds to pattern-matching in the usual sense. E.g., "fire", "dismiss", "lay-off", etc. may be matched with (FIRE); a certain demain of Integers can be matched with (JOB-NUMBER).

These possible transformations make up the kinds of knowledge which can be attached to concept structures. In the case of a concept, attached knowledge consists of the possible elaborations for that concept (like the hiring policies under (HIRE)) and possibly a match-list of the possible user descriptors that the concept can be associated with ("hire", "employ", "take on"). For a functional concept, this knowledge consists of structural transformation rules and rules defining sub-structures that can be deleted (and still allow the pattern, I.e., concept structure, to make sense).

In addition, concepts can have arbitrary properties which can be accessed and used by the model's problem-solving procedures. The system builder can make the application of any of these properties part of the map construction process by providing an explicit property access function as one of the functional concepts. In this case, the value of the property must be a valid concept structure. For example, (NAME-OF (EMPLOYE)) in a pattern implies that the value of the EMPLOYE'S NAME property can be substituted for that construct. A built-in CLASS-MEMBERSHIP property is provided by the system for relating individual concepts (e.g., (EMPLOYE) is a member of the (PEOPLE) class). CLASS-MEMBERSHIP information is used by the map construction process.

It Is important to understand that no applicability information goes along with the knowledge attached to individual concepts. Yet it is quite clear even from the few examples given here that some part of the system must decide on applicability: a particular hiring policy is not always a good model of (HIRE), "lay-off" cannot always be substituted for (FIRE), a DETERMINED-BY cannot always be changed Into a FLOW without fatal loss of information. The choice of what to use when is made by the transformation selection procedure.

The procedure itself is basically just a general function which knows how to look up transformations and apply them to arbitrary list structures. The whole philosophy of reformulation is to take advantage of as much information as possible in this procedure to choose the right transformation without extensive trial and error search. As I said earlier, the procedure relies on the structure of the model Itself to represent what the expert needs to know to solve the problem. In a well-structured model, strong expectations of what the expert needs to know constrain the choice of possible transformations to some extent. However, the procedure must have additional information in order to narrow down the choice to what is appropriate for the particular problem at hand. This information is what is drawn from the patterns that have already been matched. The information is in the form of constraints that can be divided into three categories:

(1) Constraints of the whole problem on the transformations that can be applied to each pattern;

(2) Constraints of a pattern on the transformations that can be applied to the individual elements of that pattern;

(3) Constraints within each pattern element on the transformations that can be applied to it.

In the existing business consultant program, the first category is compressed into a single piece of information with a single use. The piece of information is the program's view of what the client's symptom is; its only use Is to define the group of patterns that are applicable to that symptom. Thus, the mapping between the Dominion Co. case and the symptom concept (C3) represents the program's view of Dominion's symptom. The fact that (C3) is attached to a group of patterns which deal with damping problems defines that group of patterns to be applicable to the Dominion case.

Note that in the business consultant program, the system does work to establish a view of the symptom, i.e,, to construct the mapping to a symptom pattern. The program's view may even end up being rather different from the client's own. In other problem areas, the expert system may not have such a strong model of symptoms. For example, in a database system, the client's symptom corresponds to the overall reason why the client is using the system during any particular session (e.g., "to find out If there are any workload peaks", "to see where our parts inventory is going"). Here the system would probably <u>not</u> have the facility for deducing this reason. The client would have to provide this information (via a statement like "I'm trying to do x.") in order to make it available. The importance of this symptom or statement of intent information is that It provides a top level restriction on the patterns which have to be considered relevant. It is especially important because it is used at a time the very beginning of the problem-solving effort — when none of the other problem-based constraint information is available.

The second category, pattern-level constraints, is more interesting. These constraints exert primary influence over the choice of pattern transformations and govern the "level of detail" mechanism. The form of the constraint is just the pattern as it has been matched so far, including any matched sub-patterns that are already in the modelled version. There are also a few special forms which will be discussed in a minute. The constraints are used by the restriction mechanism to govern the choice of transformations. The restriction rule is simply stated:

> If the constraint contains an element which is more restrictive than the pattern element being transformed, only the elaborations and match-list of this more restrictive element may be used as transformations.

The comparative restrictiveness of any two concept structures can be determined on the following basis:

- an instantiation of a concept is more restrictive than that concept,
- an elaboration of a concept is more restrictive than that concept,
- A member of a class is more restrictive than the class itself ((EMPLOYE) is more restrictive than (PEOPLE)),
- a property value is more restrictive than the property itself ((MARY) is more restrictive than (NAME-OF (EMPLOYE))),
- a property is more restrictive than the concept of which it is a property ((NAME-OF (EMPLOYE)) is more restrictive than (EMPLOYE)),
- a concept which is not related to another concept in one of the above ways is not more restrictive than that concept.

All of this essentially says that once it has been determined how a concept maps into the problem at hand, future structures which refer to that concept are restricted to use the same mapping. These concept structures are then used to restrict the other concept structures that refer to them, and so on, until the model of that particular problem is complete.

Note that an important side-effect of this way of proceeding is that, as the mapping effort goes on, the choice of pattern transformations becomes more and more restricted, and the matching effort becomes more and more efficient.

A number of special mapping activities which greatly enhance the modelling capability of an expert system cannot be easily incorporated into this simple restriction mechanism. These activities are therefore handled by another pattern-level constraint — the special form alluded to earlier. The special mapping activities are: MODEL, which interrupts the mapping of the current pattern and calls the mapping procedure on a new pattern in an unrestricted mode; SIMILAR, which compares the values of given properties of the current pattern with those of another pattern; and CONTRADICTORY, which is just like SIMILAR except that it looks for given differences between two patterns.

These are handled by the PURPOSE constraint. A PURPOSE is just an elaboration of the form

(PURPOSE mode pattern)

which, when selected as a transformation, interrupts the map construction procedure, handles the pattern in the mode indicated, and then returns control to the mapping process.

An example of the use of MODEL arises in the mapping of (HIRE). It is sometimes necessary to find out the details of the production sector of the firm before choosing between certain policy alternatives of (HIRE) in (CI).

(PURPOSE MODEL production-pattern)

is therefore included as an elaboration of (HIRE), causing mapping of (HIRE) to stop until production-pattern has been matched.

The other two modes are efficiency devices. Sometimes it is much easier to see if the pattern under consideration is similar (or contradictory) enough to an <u>already matched</u> pattern to obviate the need for a separate effort on that pattern. For either SIMILAR or CONTRADICTORY, if the pattern named in the PURPOSE has not already been matched, the PURPOSE is a no-op.

This checking of properties for similarity or contradiction must be handled by specialized procedures. Tt is not part of the normal mapping process. In a more general sense, PURPOSE'S provide "hooks" for any expert problem-solving procedures which the system designer wishes to build into the mapping process. For example, the business consultant program contains an additional "SHOW-CAUSE" PURPOSE which provides special cause-effect tracing through flow patterns. Such domain dependent efficiency devices can add to the effectiveness of the mapping process without impairing the aesthetics of the reformulation approach, since their Interaction with the basic process can only be through well-defined PURPOSE channels.

Note that level of detail considerations are handled by the interaction of the restriction mechanism and the MODEL PURPOSE. In the main, the level of detail at which an arbitrary pattern is matched is determined by whether its elements or elaborations of its elements have already been matched. This is handled by the restriction mechanism, which propagates the level of detail of already matched elements throughout a new pattern. If, in the midst of this process, a pattern has to be mapped at a different level of detail, that pattern can be introduced by a MODEL PURPOSE, which removes the restrictions.

Finally, the element-level constraints on pattern transformation are expressed in terms of the CLASS-MEMBERSHIP property discussed earlier. If element A has a CLASS-MEMBERSHIP property with value B, the possible instantiations of A are restricted by an existing mapping of B. (Remember that B must also be a valid concept structure.) Usually, B is simply another concept; e.g., (PEOPLE) is the value of the CLASS-MEMBERSHIP property of (EMPLOYE). But it may be more complex, as in a value like (INTEGER (MIN-VALUE) (MAX-VALUE)), an implied form of match-list. In either case, the choice of an instantiation transformation for the pattern element A is restricted in the usual way by the existing match (if any) of concept structure B. Thus, if (PEOPLE) has already been matched to (MEMBER-OF (FRED MARY JOHN)), (EMPLOYE) can only match FRED, MARY or JOHN. If (MIN-VALUE) in the CLASS-MEMBERSHIP property above has been found to be 7, the number to which that property is attached must be at least 7.

Conclusions

The reformulation methodology described here is proposed as an alternate approach to organizing expert knowledge. Tt offers some distinct advantages over other approaches in domains in which the knowledge is well structured and the input is not. By allowing the knowledge representation to be optimized for problem-solving, and by providing local hooks for all knowledge about each concept, the reformulation approach makes it relatively easy to build in expertise and add to it later. However, it does not force the system designer to abandon control of the use of that expert knowledge to a general problem-solving method which cannot tune itself to particular problems. Finally, reformulation approach provides a map construction process which can particularize the knowledge base to specific problems in more interesting ways than simple instantiation. This makes the approach useful for domains in which it is inappropriate to severely restrict the form of user input.

The primary disadvantages of the approach result from one of its advantages: by refusing to place external restrictions on the representation of expert knowledge, the reformulation approach abdicates the responsibility of providing help in representing that knowledge. That is, there is no built-in conceptual framework for the system builder to use as the basis for representing the knowledge of the problem domain (in contrast, see [Martin]). Also, the map construction apparatus required by the reformulation approach is comparatively complex and hard to build, even with more detailed versions of the guidelines sketched here (see [Mark]). Both of these problems would be greatly alleviated if existing concepts, functional concepts, and PURPOSE'S could be shared in new application domains. This issue of providing a general underlying conceptual framework, which is of course of much interest elsewhere (again, see [Martin]), is being pursued in the current applications of this approach.

References

Brown, J. S., Benton, R. R., and Bell, A. G., SOPHIE: A Sophisticated Instructional Environment for Teaching Electronic Trouble-shooting, BBN Report #2790, Bolt, Beranek, and Neuman, Cambridge, HA, March, 1974.

Buchanan, B. G., Sutherland, G., and Feigenbaum, E., "Heuristic DENDRAL: A Program for Generating Explanatory Hypotheses in Organic Chemistry," Machine Intelligence 4^ B. Meltzer and D. Michie (eds.), American Elsevier, New York, 1969, pp. 209-254.

Davis, R., Applications of Meta Level Knowledge to the Construction, Maintenance and Use of Large Knowledge Bases, (PhD Thesis), Stanford University AI Lab Memo AIM-283, July, 1976.

Forrester, J. W., Industrial Dynamics, MIT Press, Cambridge, MA, 1969.

Gorry, G. A., "The Development of Managerial Models", Sloan Management Review, Vol. 12, No. 2, Winter, 1971, pp. 1-16.

Hart, P., "Progress on a Computer Based Consultant", Advance Papers of IJCAI 4_, MIT Artificial Intelligence Laboratory, Cambridge, MA, 1975, pp. 831-841.

Mark, W. S., The Reformulation Model of Expertise, (PhD Thesis), MIT Laboratory for Computer Science TR-172, September, 1976.

Martin, W. A., "OWL", Proceedings of the NYU Symposium on Computational Linguistics, October, 1974.

Pople, H. E., Myers, J. D., Miller, R. A., "DIALOG: A Model of Diagnostic Logic for Internal Medicine," Advance Papers of IJCAI 4^. MIT Artificial Intelligence Laboratory, Cambridge, MA, 1975, pp. 848-855.

Rubin, A. D., Hypothesis Formation and Evaluation in Medical Diagnosis, (EE Thesis), MIT Artificial Intelligence Laboratory TR-316, January, 1975.

Shortliffe, E. H., MYCIN: A Rule-Based Computer Program for Advising Physicians Regarding Anti-Microbial Therapy Selection, (PhD Thesis), Stanford University AI Lab Memo AIM-251, October, 1974.

Sussman, G. J., and Stallman, R. M., "Heuristic Techniques in Computer Aided Circuit Analysis", IEEE Transactions on Circuits and Systems, February, 1976_0