Comp. & Maths. with Appls. Vol. 11, No. 5, pp. 449–456, 1985
Printed in Great Britain.

0097-4943/85 $3.00 + .00

CSRL: A LANGUAGE FOR EXPERT SYSTEMS FOR DIAGNOSIS†

TOM BYLANDER, SANJAY MITTAL‡ and B. CHANDRASEKARAN
Artificial Intelligence Group, Department of Computer and Information Science,
The Ohio State University, Columbus, OH 43210, U.S.A.

(*Received September* 1984)

Abstract—We present CSRL (Conceptual Structures Representation Language) as a language to facilitate the development of expert diagnosis systems based on a paradigm of "cooperating diagnostic specialists." In our approach diagnostic reasoning is one of several generic tasks, each of which calls for a particular organizational and problem-solving structure. A diagnostic structure is composed of a collection of specialists, each of which corresponds to a potential hypothesis about the current case. They are organized as a classification or diagnostic hierarchy, e.g. a classification of diseases. A top-down strategy called *establish–refine* is used, in which either a specialist establishes and then refines itself, or the specialist rejects itself, pruning the hierarchy that it heads. CSRL is a language for representing the specialists of a diagnostic hierarchy and the diagnostic knowledge within them. The diagnostic knowledge is encoded at various levels of abstractions: message procedures, which describe the specialist's behavior in response to messages from other specialists; knowledge groups, which determine how data relate to features of the hypothesis; and rule-like knowledge, which is contained within knowledge groups.

1. INTRODUCTION

Many kinds of problem-solving for expert systems have been proposed within the AI community. Whatever the approach, there is a need to acquire the knowledge in a given domain and implement it in the spirit of the problem-solving paradigm. Reducing the time to implement a system usually involves the creation of a high-level language that reflects the intended method of problem-solving. For example, EMYCIN[1] was created for building systems based on MYCIN-like problem-solving[2]. Such languages are also intended to speed up the knowledge acquisition process by allowing domain experts to input knowledge in a form close to their conceptual level. Another goal is to make it easier to enforce consistency between the expert's knowledge and its implementation.

CSRL (Conceptual Structures Representation Language) is a language for implementing expert diagnostic systems that are based on our approach to diagnostic problem-solving. This approach is an outgrowth of our group's experience with MDX, a medical diagnostic program[3], and with applying MDX-like problem-solving to other medical and nonmedical domains. CSRL facilitates the development of diagnostic systems by supporting constructs that represent diagnostic knowledge at appropriate levels of abstraction.

First, we will overview the relationship of CSRL to our overall theory of problem-solving types and the diagnostic problem-solving that underlies CSRL. We then present CSRL, illustrating how its constructs are used to encode diagnostic knowledge. Two expert systems under development in our laboratory, which use CSRL, are then briefly described. Based on our experience with these systems, we point out where improvements in CSRL are needed.

2. CLASSIFICATORY DIAGNOSIS

The central problem-solving of diagnosis, in our view, is classificatory activity. This is a specific type of problem-solving in our approach, meaning that a special kind of organization and special strategies are strongly associated with performing expert diagnosis. In this section we will briefly review the theory of problem-solving types, as presented by Chandrasekaran[4], and the structure and strategies of the diagnostic task[5].

†This an expanded version of a paper of the same title presented at the 1983 International Joint Conference on Artificial Intelligence.

‡Currently at Knowledge Systems Area, Xerox PARC, 3333 Coyote Hill Rd., Palo Alto, CA 94304, U.S.A.

2.1 *Types of problem-solving*

We propose that expert problem-solving is composed of a collection of different problem-solving abilities. The AI group at Ohio State has been working at identifying well-defined types of problem-solving (called generic tasks), one of which is classificatory diagnosis. (For the purposes of this discussion we will use "diagnosis" in place of "classificatory diagnosis" with the understanding that the complete diagnostic process includes other elements as well.) Other examples include knowledge-directed data retrieval, consequence finding, and a restricted form of design.

Each generic task calls for a particular organizational and problem-solving structure. Given a specific kind of task to perform, the idea is that specific ways to organize and use knowledge are ideally suited for that task.

Even when the specification of a problem is reduced to a given task within a given domain, the amount of knowledge that is needed can still be enormous (e.g. diagnosis in medicine). In our approach the knowledge structure for a given task and domain is composed of *specialists*, each of which specialize in different concepts of the domain. Domain knowledge is distributed across the specialists, dividing the problem into more manageable parts, and organizing the knowledge into chunks that become relevant when the corresponding concepts become relevant during the problem-solving.

Decomposing a domain into specialists raises the problem of how they will coordinate during the problem-solving process. First, the specialists as a whole are organized primarily around the "subspecialist-of" relationship. Each task may specify additional relationships that may hold between specialists. Second, each task is associated with a set of strategies that take advantage of these relationships and the problem-solving capabilities of the individual specialists. The choice of what strategy to follow is not a global decision, but is chosen by the specialists during problem-solving.

2.2 *The diagnostic task*

The diagnostic task is the identification of a case description with a specific node in a predetermined diagnostic hierarchy. Each node in the hierarchy corresponds to a hypothesis about the current case. Nodes higher in the hierarchy represent more general hypotheses, and lower nodes are more specific. Typically, a diagnostic hierarchy is a classification of malfunctions of some object, and the case description contains the manifestations and background information about the object. For example, the Auto-Mech expert system[6] attempts to classify data concerning an automobile into a diagnostic hierarchy of fuel-system malfunctions. Figure 1 illustrates a fragment of Auto-Mech's hierarchy. The most general node, the fuel system in this example, is the head node of hierarchy. More specific fuel-system malfunctions, such as fuel-delivery problems, are classified within the hierarchy.

Each node in the hierarchy is associated with a *specialist* that contains the diagnostic knowledge to evaluate the plausibility of the hypothesis from the case description. From this knowledge the specialist determines a confidence value representing the amount of belief in the hypothesis. If this value is high enough, the specialist is said to be *established*.

The basic strategy of the diagnostic task is a process of hypothesis refinement, which we call *establish–refine*. In this strategy, if a specialist establishes itself, then it *refines* the hypothesis by invoking its subspecialists, which also perform the establish–refine strategy. If its confidence value is low, the specialist *rejects* the hypothesis and performs no further actions. Note that when this happens, the whole hierarchy below the specialist is eliminated from consideration. Otherwise the specialist *suspends* itself and may later refine itself if its superior requests it.

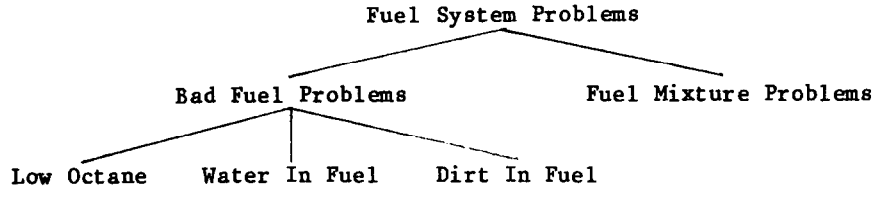

Fig. 1. Fragment of a diagnostic hierarchy.

With regard to Fig. 1, the following scenario might occur. First, the fuel-system specialist is invoked, since it is the top specialist in the hierarchy. This specialist is then established, and the two specialists below it are invoked. Bad fuel problems are rejected, eliminating the three subspecialists of bad fuel from consideration. Finally, the fuel-mixture specialist is established, and its subspecialists (not shown) are invoked.

An important companion to the diagnostic hierarchy is an intelligent data-base assistant that organizes the case description, answers queries from the diagnostic specialists, and makes simple inferences from the data[7]. For example, the data base should be able to infer that the fuel tank is not empty if the car can be started. The diagnostic specialists are then relieved from knowing all the ways that a particular datum could be inferred from other data.

There are several issues relevant to diagnostic problem-solving that we will not address here. The simple description above does not employ strategies for bypassing the hierarchical structure for common malfunctions, for handling multiple interacting hypothesis, or for accounting for the manifestations. Also, additional control strategies are required when many nodes are in a suspended state. For discussion on some of these topics, see Gomez and Chandrasekaran[5]. Test ordering, causal explanation of findings, and therapeutic action do not directly fall within the auspices of the classificatory diagnosis as defined here, but expertise in any of these areas would certainly enhance a diagnostic system. Fully resolving all of these issues and integrating their solutions into the diagnostic framework are problems for future research.

2.3 *Differences from other approaches*

The usual approach to building knowledge-based systems is to emphasize a general knowledge representation structure and different problem-solvers that use that knowledge. One difference in this approach is that the organization of knowledge is not intended as a general representation for all problems. Rather it is tuned specifically for diagnosis. By limiting the type of problem to be solved, a specific organizational technique (classification hierarchy) and problem-solving strategy (establish–refine) can be used to provide focus and control in the problem-solving process.

Another difference is that the specialists in the hierarchy are not a static collection of knowledge. The knowledge of how to establish or reject is embedded within the specialists. Each specialist can then be viewed as an individual problem-solver with its own knowledge base. The entire collection of specialists engages in distributed problem-solving.

3. CSRL

CSRL is a language for representing the specialists of a diagnostic hierarchy and the diagnostic knowledge within them. The diagnostic knowledge is encoded at various levels of abstractions. *Message procedures* describe the specialist's behavior in response to messages from other specialists. These contain the knowledge about how to establish or refine a specialist. *Knowledge groups* determine how selected data relate to various features or intermediate hypotheses that are related to the specialist. The selected data may be the values of other knowledge groups, so that a single knowledge group can "summarize" the results of several others. Knowledge groups are composed of rule-like knowledge that matches the data against specific patterns and, when successful, provides values to be processed by the knowledge group.

3.1 *Specialists*

In CSRL a diagnostic expert system is implemented by individually defining each specialist. The super- and subspecialists of the specialist are declared within the definition. Figure 2 is a

```
(Specialist BadFuel
    (declare (superspecialist FuelSystem)
             (subspecialists LowOctane WaterInFuel DirtInFuel))
    (kgs ...)
    (messages ...))
```

Fig. 2. Skeleton specialist for BadFuel.

skeleton of a specialist definition for the bad fuel node from Fig. 1. The declare section specifies its relationships to other specialists. The other sections of the specialist are examined below.

Since CSRL is designed to use only a simple classification tree, many choices concerning the composition of the hierarchy must be made. This is a pragmatic decision, rather than a search for the "perfect" classification tree. The main criteria for evaluating a classification is whether enough evidence is normally available to make confident decisions. To decompose a specialist into its subspecialists, the simplest method is to ask the domain expert what subhypotheses should be considered next. Usually the subspecialists will differ from one another based on a single attribute (e.g. location, cause). For further discussion on this and other design decisions in CSRL, see Bylander and Smith[8].

3.2 *Message procedures*

The messages section of a specialist contains a list of message procedures that specify how the specialist will respond to different messages from its superspecialist.† "Establish," "Refine," "Establish–Refine" (combines Establish and Refine), and "Suggest" are predefined messages in CSRL; additional messages may be defined by the user. Below, we will examine how Establish and Refine procedures are typically constructed.

Message procedures are the highest level of abstraction for diagnostic knowledge within specialists. Just as in general message-passing languages, messages provide a way to invoke a particular kind of response without having to know what procedure to invoke. Strategies for diagnosis, such as Establish–Refine, are usually easy to translate into a message protocol. However, CSRL does not provide any way to specify and enforce message protocols.

Figure 3 illustrates the Establish message procedure of the BadFuel specialist. "relevant" and "summary" are names of knowledge groups of BadFuel. "self" is a keyword that refers to the name of the specialist. This procedure first tests the value of the relevant knowledge group. (If this knowledge group has not already been executed, it is automatically executed at this point.) If it is greater than or equal to 0, then BadFuel's confidence value is set to the value of the summary knowledge group, or else it is set to the value of the relevant knowledge group. In CSRL a confidence value scale of -3 to $+3$ is used (integers only). A value of $+2$ or $+3$ indicates that the specialist is established. In this case the procedure corresponds to the following diagnostic knowledge.

> First perform a preliminary check to make sure that BadFuel is a relevant hypothesis to hold. If it is not (the relevant knowledge group is less than 0), then set BadFuel's confidence value to the degree of relevancy. Otherwise, perform more complicated reasoning (the summary knowledge group combines the values of other knowledge groups) to determine BadFuel's confidence value.

Figure 4 shows a Refine procedure that is a simplified version of the one that BadFuel uses. "subspecialists" is a keyword that refers to the subspecialists of the current specialist. The procedure calls each subspecialist with an English message.‡ If the subspecialist establishes itself (+? tests if the confidence value is $+2$ or $+3$), then send it a Refine message.

CSRL has a variety of other kinds of statements and expressions so that more complicated

```
(Establish (if (GE relevant 0)
               then (SetConfidence self summary)
             else (SetConfidence self relevant)))
```

Fig. 3. Establish procedure of BadFuel.

†A specialist is not allowed to send messages to its superspecialist. However, other message-passing routes are allowed. Specifically, a specialist may send a message to itself, across the hierarchy, and to indirect subspecialists. In the latter case each interconnecting specialist is sent a "suggest" message and decides within its suggest message procedure whether or not to pass the original message downwards.

‡For convenience many of CSRL's control constructs mimic those of INTERLISP; however, these constructs are executed by the CSRL interpreter, not by using LISP EVAL. LISP code is allowed within message procedures, but only within a construct called "DoLisp." This is not intended to let specialists have arbitrary code, but to allow interaction with other LISP-implemented systems.

```
(Refine (for specialist in subspecialists
          do (Call specialist with Establish)
             (if (+? specialist)
                 then (Call specialist with Refine))))
```

Fig. 4. Refine procedure.

strategies can be implemented. For example, a "Reset" statement deletes the confidence value and the knowledge group values of a specialist. This might be used when additional tests are performed, making it necessary to recalculate the confidence value. Also, messages can be parameterized, and message procedures can declare local variables.

3.3 *Knowledge groups*

The kgs section of a specialist definition contains a list of knowledge groups that are used to evaluate how selected data indicate various features or intermediate hypotheses that relate to specialist's hypothesis. A knowledge group can be thought of as a cluster of production rules that map the values of a list of expressions (boolean and arithmetic operations on data) to some conclusion on a discrete, symbolic scale. Different types of knowledge groups perform this mapping differently: e.g. directly mapping values to conclusions, or having each rule add or subtract a set number of "confidence" units.

Knowledge groups are intended for encoding the heuristics that a domain expert uses for inferring features of an hypothesis from the case description. The main problem is that this inference is uncertain—there is rarely a one-to-one mapping from data to the features of the hypothesis. The way that this is handled in CSRL is borrowed from the uncertainty handling techniques used in MDX[9].

Each feature or intermediate hypothesis is associated with a knowledge group. The data that the domain expert uses to evaluate the feature are encoded as expressions in the knowledge group. These are usually queries to a separate data-base system. Each combination of values of the expressions is then mapped to a level of confidence as determined by the domain expert. This set of knowledge groups becomes the data for another knowledge group, which determines the confidence value of the specialist from the confidence values of the features.† By examining the results of test cases, we see that the knowledge groups are relatively easy to debug, since the attention of the domain expert can be directed to the specific area of knowledge that derived the incorrect result.

As an example, Fig. 5 is the relevant knowledge group of the BadFuel specialist mentioned above. It determines whether the symptoms of the automobile are consistent with bad fuel problems. The expressions query the user (who is the data base for Auto-Mech) for whether the car is slow to respond, starts hard, has knocking or pinging sounds, or has the problem when accelerating. "AskYNU?" is a LISP function that asks the user for a Y, N, or U (unknown) answer from the user, and translates the answer into T, F, or U, the values of CSRL's three-

```
(relevant Table
     (match (AskYNU? "Is the car slow to respond")
            (AskYNU? "Does the car start hard")
            (And (AskYNU? "Do you hear knocking or pinging sounds")
                 (AskYNU? "Does the problem occur while accelerating"))
       with (if T ? ?
                then -3
              elseif ? T ?
                then -3
              elseif ? ? T
                then 3
              else 1)))
```

Fig. 5. Relevant knowledge group of BadFuel.

†Actually, any number of knowledge group levels can be implemented.

valued logic. Each set of tests in the if-then part of the knowledge group is evaluated until one matches. The value corresponding to this "rule" becomes the value of the knowledge group. For example, the first rule tests whether the first expression is true (the "?" means doesn't matter). If so, then −3 becomes the value of the knowledge group. Otherwise, other rules are evaluated. The value of the knowledge group will be 1 if no rule matches. This knowledge group encodes the following diagnostic knowledge:

> If the car is slow to respond or if the car starts hard, then BadFuel is not relevant in this case. Otherwise, if there are knocking or pinging sounds and if the problem occurs while accelerating, then BadFuel is highly relevant. In all other cases BadFuel is only mildly relevant.

Figure 6 is the summary knowledge group of BadFuel. Its expressions are the values of the relevant and gas knowledge groups (the latter queries the user about the temporal relationship between the onset of the problem and when gas was last bought). In this case, if the value of the relevant knowledge group is 3 and the value of the gas knowledge group is greater than or equal to 0, then the value of the summary knowledge group (and consequently the confidence value of BadFuel) is 3, indicating that a bad-fuel problem is very likely.

3.4 *Comparison with rule-based languages*

There is nothing in CSRL that is not programmable within rule-based languages such as OPS5[10] or EMYCIN[1]. The difference between CSRL and these languages is that CSRL makes a commitment to a particular organizational and programming style. CSRL is not intended to be a general-purpose representation language, but is built specifically for the classificatory diagnosis problem. It is possible to program in a rule-based language, so that there is an implicit relationship between rules so that they correspond to knowledge groups and specialists. R1, although not a diagnostic expert system, is an excellent example of how one creates implicit grouping of rules in such a system[11]. The central idea underlying CSRL is to make these relationships explicit. The expert system implementor is then relieved from trying to impose an organization on a organizationless system and is free to concentrate on the conceptual structure of the domain. Also, there is a greater potential to embed explanation and debugging facilities that can take advantage of the expert system organization.

3.5 *The CSRL environment*

The current version of CSRL is implemented in INTERLISP-D and LOOPS, an object-oriented programming tool. Each specialist is implemented as a LOOPS class, which is instantiated for each case that is run. The LOOPS class hierarchy is used to specify default message procedures and shared knowledge groups, making it easy to encode a default establish–refine strategy, and letting the user incrementally modify this strategy and add strategies as desired. A graphical interface displays the specialist hierarchy and, through the use of a mouse, allows the user to easily access and modify any part of the hierarchy. Additional facilities for debugging and explanation are being implemented.

4. EXPERT SYSTEMS THAT USE CSRL

4.1 *Auto-Mech*

Auto-Mech is an expert system that diagnoses fuel problems in automobile engines[6]. This domain was chosen to demonstrate the viability of our approach to nonmedical domains, as well as to gain experience and feedback on CSRL.† The purpose of the fuel system is to deliver a mixture of fuel and air to the air cylinders of the engine. It can be divided into major subsystems (fuel delivery, air intake, carburetor, vacuum manifold) that correspond to initial hypotheses about fuel-system faults.

†Auto-Mech was developed using an early version of the language.

```
(summary Table
    (match relevant gas
      with (if 3 (GE 0)
               then 3
             elseif 1 (GE 0)
               then 2
             elseif ? (LT 0)
               then -3)))
```

Fig. 6. Summary knowledge group of BadFuel.

Auto-Mech consists of 34 CSRL specialists in a hierarchy that varies from four to six levels deep. Its problem-solving closely follows the establish–refine strategy. Before this strategy is invoked, Auto-Mech collects some initial data from the user. This includes the major symptom that the user notices (such as stalling) and the situation when this occurs (e.g. accelerating and cold engine temperature). Any additional questions are asked while Auto-Mech's specialists are running. The diagnosis then starts and continues until the user is satisfied that the diagnosis is complete. The user must make this decision because the data that Auto-Mech uses are very weak at indicating specific problems, and, more importantly, Auto-Mech is unable to make the repair and determine whether the problem has been fixed.

A major part of Auto-Mech's development was determining the assumptions that would be made about the design of the automobile engine and the data that the program would be using. Different automobile engine designs have a significant effect on the hypotheses that are considered. A carbureted engine, for example, will have a different set of problems than a fuel-injected engine (the former can have a broken carburetor). The data was assumed to come from commonly available resources. The variety of computer analysis information that is available to mechanics today was not considered, in order to simplify building Auto-Mech.

4.2 *Red*

Red is an expert system whose domain is red-blood-cell antibody identification[12]. An everyday problem that a blood bank contends with is the selection of units of blood for transfusion during major surgery. The primary difficulty is that antibodies in the patient's blood may attack the foreign blood, rendering the new blood useless as well as presenting additional danger to the patient. Thus, identifying the patient's antibodies and selecting blood that will not react with them is a critical task for nearly all red-blood transfusions.

The Red expert system is composed of three major subsystems, one of which is implemented in CSRL. The non-CSRL subsystems are a data base, which maintains and answers questions about reaction records (reactions of the patient's blood in selected blood samples under a variety of conditions), and an overview system, which assembles a composite hypothesis of the antibodies that would best explain the reaction record[13]. CSRL is used to implement specialists corresponding to each antibody that Red knows about (about 30 of the most common ones) and to each antibody subtype (different ways that the antibody can react).

The major function of the specialists is to rule out antibodies and their subtypes whenever possible, thus simplifying the job of the overview subsystem, and to assign confidence values, informing overview of which antibodies appear to be more plausible. The specialists query the data base for information about the test reactions and other patient information, and also tell the data base to perform certain operations on reaction records.

An interesting feature of Red is the way it handles the problem of interacting hypotheses. It is possible for the patient's blood to have practically any number or combination of antibodies, which makes it very hard for a single specialist to determine how well it will fit with other specialists in a composite hypothesis. In Red each specialist is encoded to assume that it is independent—it looks at the data as if no other specialist can account for the same data. The knowledge of how the specialists can interact is left to the overview subsystem. This would be problematic if few specialists could rule themselves out, but it so happens that in this domain it is rare to have more than a few antibodies that cannot be independently ruled out. Thus Red's CSRL subsystem makes overview's problem-solving computationally feasible since it considerably reduces the amount of search that would otherwise be necessary.

5. NEEDED IMPROVEMENTS IN CSRL

The largest flaw in CSRL is that there is no strategy that determines when diagnosis should stop. Currently, the default procedures simply ask the user if the current diagnosis is satisfactory. Some notion of what it means to account for the data needs to be added to the language. The work on Red's overview system is a step in this direction, but there needs to be more integration of overview and CSRL (currently overview starts after the specialists are finished) and a better understanding of what kinds of interactions can occur between two hypotheses. Progress in this area would also help increase the focus of the diagnosis; i.e. the diagnosis could concentrate on accounting for the most important manifestation(s).

Another problem is the meaning of the confidence value of a specialist. In MDX this value was directly associated with the amount of belief in the specialist. However, in both Auto-Mech and Red, this meaning had to be slightly altered to fit the purposes of the expert system. In Auto-Mech the confidence value is used to indicate whether the hypothesis was worth pursuing. In Red it is used to indicate the specialist's plausibility, given the independence assumption mentioned earlier. It is not possible in either expert system to confirm a specialist without outside help. In Auto-Mech a repair or highly specific test must be performed, but in Red all the specialists must be considered together. This does not create a problem for the process of establish–refine problem-solving, but makes it difficult to explain what the confidence value means. Any explanation facility must understand the assumptions that are being made in order to make coherent explanations.

6. CONCLUSION

We believe that the development of complex expert systems will depend on the availability of special-purpose languages with organizational and problem-solving tools that match the conceptual structure of the domain. CSRL represents an initial step in this direction. It provides facilities to organize diagnostic knowledge in accordance with the structure of the domain. In particular, CSRL's constructs facilitate the encoding of rule-like and strategic knowledge into appropriate abstractions: knowledge groups, message procedures, and specialists.

Acknowledgments—We would like to acknowledge Jack Smith and Jon Sticklen for many fruitful discussions concerning CSRL's design. Many improvements in the language are due to Mike Tanner and John Josephson, who implemented the CSRL specialists in Auto-Mech and Red., The language development is funded by a grant from the Battelle Memorial Laboratories University Distribution Program, and experimentation and application in different domains is supported by AFOSR grant 82-0255, and NSF grant MCS-8103480.

REFERENCES

1. W. van Melle, A domain independent production-rule system for consultation programs. *Proc. Sixth Int. Conf. on Artificial Intelligence*, pp. 923–925. Tokyo (1979).
2. E. H. Shortliffe, *Computer-Based Medical Consultations: MYCIN*. Elsevier, New York (1976).
3. B. Chandrasekaran and S. Mittal, Conceptual representation of medical knowledge for diagnosis by computer: MDX and related systems, in *Advances in Computers*, pp. 217–293. Academic Press, New York, (1983).
4. B. Chandrasekaran, Towards a taxonomy of problem solving types. *AI Mag.* **4**, 9–17 (1983).
5. F. Gomez and B. Chandrasekaran, Knowledge organization and distribution for medical diagnosis. *IEEE Trans. Syst., Man Cybernetics* **SMC-11**, 34–42 (1981).
6. M. C. Tanner and T. Bylander, Application of the CSRL language to the design of expert diagnosis systems: the auto-mech experience. *Proc. Joint Services Workshop on Artificial Intelligence in Maintenance*. Department of Defense, pp. 131–152, Denver (1984).
7. S. Mittal and B. Chandrasekaran, Conceptual representation of patient data bases. *J. Medical Syst.* **4**, 169–185 (1980).
8. T. Bylander and J. W. Smith, Using CSRL for medical diagnosis. *Proc. Second Int. Conf. on Medical Computer Science and Computational Medicine*. IEEE Computer Soc., Glouster, Ohio (1983).
9. B. Chandrasekaran, S. Mittal and J. W. Smith, Reasoning with uncertain knowledge: the MDX approach. *Proc. Congress American Medical Infomatics Association*. San Francisco (1982).
10. C. L. Forgy, *OPS5 Users Manual*. Tech. Rept. CMU-CS-81-135, Carnegie–Mellon University (1981).
11. J. McDermott, R1: a rule-based configurer of computer systems. *Artificial Intell.* **19**, 39–88 (1982).
12. J. W. Smith, J. Josephson, C. Evans, P. Straum and J. Noga, Design for a red-cell antibody identification expert. *Proc. Second Int. Conf. on Medical Computer Science and Computational Medicine*. IEEE Computer Soc., Glouster, Ohio (1983).
13. J. Josephson, B. Chandrasekaran and J. W. Smith, *The Overview Function in Diagnostic Problem Solving*. Technical Paper, AI Group, Dept. of Computer and Information Science, The Ohio State University (1984).