AI Magazine Volume 7 Number 3 (1986) (© AAAI)

Tom Bylander & Sanjay Mittal

CRSL: A Language for Classificatory Problem Solving and Uncertainty Handling

The ability to map the state of an object into a category in a classification hierarchy has long been an important part of many fields, for example, biology and medicine. Recently, AI research has focused increasing attention on classification (Gomez & Chandrasekaran, 1981; Weiss & Kulikowski, 1984; Clancey, 1985; Cohen *et al.* 1985; and Gordon and Shortliffe, 1985), and has been especially concerned with applying classification to diagnostic problems. One of the problems in classification is that the relationship between observable evidence and categories is often ambiguous. A piece of evidence can be associated with several categories or can occur with a category in an irregular fashion. As a consequence, uncertainty handling is an important facet of classificatory problem solving. In this article, we present and explain a programming language called CSRL, which is intended to facilitate the construction of knowledge-based systems that combine classificatory reasoning with uncertainty handling.

The Need for Special Languages

Languages developed within computer science often embody theories and assumptions about how programs and data should be organized. AI theories, however, are not concerned with programs and data *per se,* but with the organization of knowledge and its use. The problem for AI languages is transforming AI theories into symbolic structures. This pattern can be seen in knowledge representation (for example, semantic nets and KL-ONE [Brachman and Schmolze, 1985]) and in knowledge-based programming (for example, knowledge sources and OPM [Hayes-Roth, 1985]).

Motivations. Why develop "yet another language"? Our desire to transform a particular theory into a language is motivated by the following needs:

Although nearly all programming languages are adequate at the symbol level (being equivalent to Turing machines), the interpretation of their symbols is relatively unconstrained. Although the constructs of these languages can be used to enforce internal consistency of symbols and symbol structures, there is little restriction on the external meanings of symbols. The differences between the language and the real world must be covered by software engineering and programming ability.

Many AI languages do not provide constructs that make the organization of the problem solving explicit. For example, R1 (McDermott, 1982), which is implemented in OPS5 (Forgy, 1981), performs a sequence of design subtasks, each of which is implemented as a set of production rules. However, OPS5 has no organizing construct larger than a single rule, so the grouping of rules and the sequencing from one set of rules to another are achieved by

Tom Bylander is at the Laboratory for Artificial Intelligence Research, Department of Computer and Information Science, The Ohio State University, Columbus, Ohio 43210 Sanjay Mittal is a member of the research staff at Xerox Palo Alto Research Center, 3333 Coyote Hill Road, Palo Alto, California, 94304.

We thank B. Chandrasekaran for his direction and support of the CSRL project. We would also like to acknowledge Jack Smith and Jon Sticklen for many fruitful discussions concerning the design of CSRL. Many improvements in the language are due to Mike Tanner and John Josephson, who, along with many other people, implemented the CSRL specialists in the Auto-Mech and Red expert systems. MDX, the diagnostic system that led to the development of CSRL, and the original theoretical ideas behind MDX resulted from a collaboration between B Chandrasekaran, Fernando Gomez, Sanjay Mittal, and Jack Smith. The language development is funded by a grant from Battelle Memorial Institute's University Distribution Program, and experimentation and application in different domains is supported by AFOSR grant 82-0255 and NSF grant MCS83-05032.

Abstract

In this article, we present a programming language for expressing classificatory problem solvers. CSRL (Conceptual Structures Representation Language) provides structures for representing classification trees, for navigating within those trees, and for encoding uncertainty judgments about the presence of hypotheses. We discuss the motivations, theory, and assumptions that underlie CSRL. Also, some expert systems constructed with CSRL are briefly described.

programming techniques. The point is not that it was difficult or unnatural to use OPS5 for R1, but that R1's method of design problem solving and OPS5's production-rule mechanism are at different levels of organization and problem solving. If the explanation of R1's problem solving were to be automated, knowledge about OPS5 would not be enough. Clancey (1981) noted a similar problem with MYCIN and its rules when he developed a program for explaining MYCIN's knowledge.

Both the generality and the mismatch of organization make it difficult to judge the sorts of problems that these languages can naturally solve. The gap between the languages and the problems is simply too large.

CSRL for Classification The AI community needs to develop languages that are specific to particular ways of organizing knowledge and problem solving. Such languages will be powerful when the problems of a domain match the capabilities of the languages. CSRL is intended to be such a language for developing classificatory problem solvers. Although CSRL doesn't eliminate the need for significant amounts of knowledge engineering, it does provide constructs that encode classificatory knowledge in a variety of ways—from the hypotheses in the classification tree to rules that match on the situation being classified.

Because it is not a general programming language, CSRL should not be viewed as a total solution for developing expert systems but as one of the building blocks for constructing any single expert system. For illustration, we discuss CSRL's role in some expert systems, as well as its general role in diagnostic problem solving, looking at its relationship to other strategies that are diagnostically useful.

Generic Tasks

CSRL's theoretical base comes from Chandrasekaran (1983, 1985), who proposes that expert problem solving relies on a number of elementary organizational and information-processing strategies, which are called *generic tasks*. Each generic task has a particular kind of conceptual organization and a set of problem-solving strategies which take advantage of that organization. The idea is to model an expert as several problem-solving structures, where each structure performs a generic task, and all the structures cooperate to solve the problems presented to them.

The word "task" might be misleading here. The definition of a problem does not directly specify what generic task is appropriate for it, rather a generic task is a strategy that can be adopted for organizing domain knowledge for a specific type of problem solving. More than one generic task might be appropriate for a problem if the domain knowledge can be adapted to the requirements of each generic task.

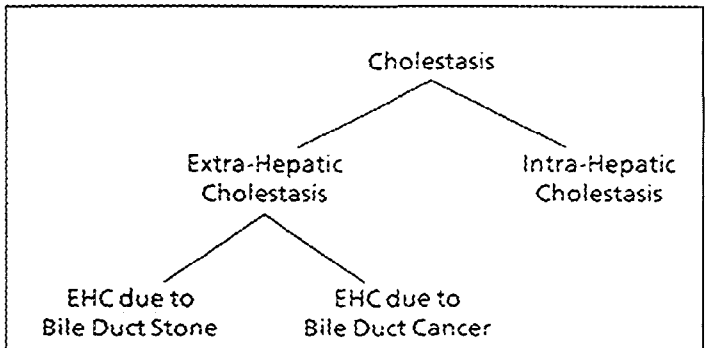
Classificatory Problem Solving

CSRL is intended to express problem solvers of one generic task, called *classification*, which is finding the categories or hypotheses within a classification hierarchy that apply to the situation being analyzed.[1] CSRL generalizes the diagnostic problem solving of MDX, an expert system in the medical domain of cholestasis (the lack of bile flow from liver to intestine)(Chandrasekaran *et al.* 1979; Mittal, 1980; Gomez and Chandrasekaran, 1981; Chandrasekaran and Mittal, 1983).

MDX explicitly organizes and uses knowledge in a way that applies to classification in general. Recently, Clancey (1984, 1985) pointed out that a number of AI systems, diagnostic and nondiagnostic, can be described as performing classification. The work on MDX and CSRL lends more weight to Clancey's historical observation but provides a somewhat different framework for analyzing classificatory problem solving. These differences are discussed in a later section.

MDX has a number of interesting features that, not surprisingly, define the design goals of CSRL:

Explicit Classification Hierarchy. Disease hypotheses are organized as a classification tree in which the children of a node represent subhypotheses of the parent. Figure 1 illustrates a fragment of the MDX tree.



Fragment of MDX's Classification Tree

Figure 1

Local Decision Criteria. The responsibility for calculating the uncertainty of hypotheses and for directing "attention" to and from hypotheses is distributed over the nodes. For example, the procedure for calculating the degree of certainty in cholestasis would be located in the cholestasis node. In addition, the procedure(s) for directing attention away from cholestasis to related hypotheses would also be located there. For this reason and also to make an analogy with the organization of the medical community, these nodes are called *specialists*. In this

[1]This is not to say that all classification is hierarchical but that classification hierarchies are associated with strategies that do not apply to other forms of knowledge organization

metaphor, "directing attention" is realized as transfer of control between specialists.

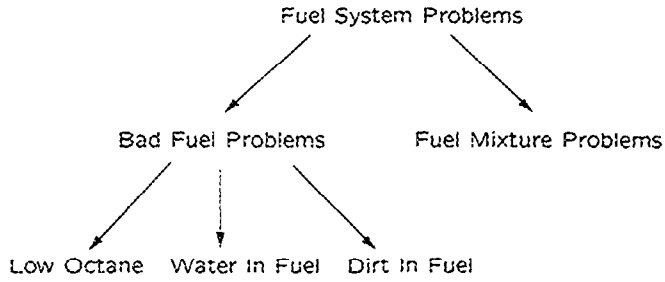
Establish, Refine. Transfer of control is primarily accomplished through a type of hypothesis refinement called *establish-refine*. Simply put, a specialist that confirms its hypothesis (the establish part) invokes its subspecialists (the refine part). A specialist that rules out or rejects its hypothesis also rules out all of the subhypotheses. At other levels of certainty, the specialist relies on domain knowledge and the context of the problem solving to determine what to do next. For systematic search, it is natural to initially give control to the root specialist. The MDX specialists also contain suggestion rules that allow the specialists to perform a data-directed search.

Symbolic Uncertainty Calculation. Calculation of uncertainty is not done by a numerically based calculus but by directly encoding the symbolic judgment of domain experts. We have more to say about this later.

MDX represents a general approach to the classification problem by organizing domain knowledge along the classification hierarchy and integrating it with the establish-refine strategy. In contrast to approaches that make a strict separation between domain knowledge and strategic knowledge, the architecture of MDX permits domain knowledge to directly influence strategic decisions. This approach explicitly recognizes the need for domain knowledge to guide the problem solving as well as the need for domain knowledge to be adapted to the problem-solving strategy.

CSRL

CSRL is a language for representing the specialists of a classification hierarchy and the knowledge within them. Classificatory knowledge is encoded at various levels of abstraction in addition to the nodes in the hierarchy. *Message procedures* describe a specialist's behavior in response to messages from other specialists. These contain the knowledge about how to establish or refine a specialist. *Knowledge groups* are primarily used for uncertainty handling. Knowledge groups are composed of rule-like knowledge that matches the data against specific patterns and, when successful, determines the value of the knowledge group. In the following discussion, we use the classification tree displayed in Figure 2, which is taken from the Auto-Mech expert system (Tanner and Bylander, 1985).



Fragment of Auto-Mech's Classification Tree

Figure 2

Encoding of Classification Trees

In CSRL a classification system is implemented by individually defining each specialist. The superspecialists and subspecialists of the specialist are declared within the definition. Figure 3 is a skeleton of a specialist definition for the bad fuel problems node from Figure 2. The `declare` section specifies its relationships to other specialists. The other sections of the specialist are examined later.

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

Skeleton Specialist for `BadFuel`

Figure 3

Because 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 criterion 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. The subhypotheses should be subtypes of the specialist's hypothesis and usually differ from one another based on a single attribute (for example, location, cause). For further discussion on the design of classification trees, see Mittal (1980) and Bylander and Smith (1985).

Encoding of Local Decision Criteria

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.[2]

[2] A specialist is not allowed to send messages to its superspecialist. However, other message-passing routes are allowed. Specifically, a specialist can 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 to pass the original message downward.

`Establish-refine` (combines `establish` and `refine`), and `suggest` are predefined messages in CSRL; additional messages can be defined by the user. Later we examine how establish and refine procedures are typically constructed.

Message procedures are the highest level of abstraction for classificatory 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 and allow the receiver of the message (the specialist in this case) to call upon local knowledge to make its decisions. However, the important thing about message passing in CSRL is not that it's useful as a general programming style but that the organization of problem solving can be modeled by identifying certain kinds of messages with specific meaning.

Encoding of the Establish-Refine Protocol

The establish message procedure of a specialist determines the confidence value in the specialist's hypothesis. Figure 4 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 evaluated, it is automatically evaluated 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; if not, 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 classificatory 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.

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

Establish Procedure of `BadFuel`

Figure 4

The refine message procedure determines what subspecialists should be invoked and the messages they are sent. Figure 5 shows a refine procedure which 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 `establish` message.[3] If the subspecialist establishes itself (`+?` tests if the confidence value is +2 or +3). Then the subspecialist it is sent a `refine` message.

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

Example Refine Procedure

Figure 5

CSRL also has a facility for specifying suggestion rules in specialists. This is done by implementing "suggestion" knowledge groups, which are elements of the `kgs` section of the specialist definition. Figure 6 is a suggestion knowledge group that might be part of the `FuelMixture` specialist (Auto-Mech itself does not use any suggestion rules). The `then` part of each rule indicates the specialist(s) that the rule's condition suggests. The value of the knowledge group is the list of specialists that are associated with the rules whose conditions are true. The knowledge group corresponds to the following knowledge:

> A fuel mixture problem should be initially considered if the car's problem occurs only when the engine is hot or cold. Knocking or pinging sounds suggest that the fuel is bad.

```
(fuelProblemSuggestions Suggestion
    (if (Or (Ask-YNU? ''Does the problem occur only
                      when the engine is cold")
            (Ask-YNU? ''Does the problem occur only
                      when the engine is hot"))
        then FuelMixture)
    (if (Ask-YNU? ''Do you hear knocking or pinging
                      sounds")
        then BadFuel))
```

Example Suggestion Knowledge Group

Figure 6

To use suggestion rules, the default refine procedure (Figure 5) must be modified. Figure 7 illustrates how this can be done. The modification is the addition of another loop that invokes the suggested specialists with an `establish` message, conditionally followed by a `refine` message. The second loop is nearly the same as the default procedure except that it avoids reinvoking any of the suggested specialists.

---

[3] For convenience, many of the CSRL control constructs mimic those of INTERLISP; however, these constructs are executed by the CSRL interpreter, not by LISP. LISP code is allowed within message procedures, but only within a construct called `DoLisp` This is intended to allow interaction with other LISP-implemented systems

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

```
(Refine
    (for specialist in fuelProblemSuggestions
       do (Call specialist with Establish)
         (if (+?  specialist)
           then (Call specialist with Refine)))
    (for specialist in subspecialists
       do (if (Not (Member?  specialist
                    fuelProblemSuggestions))
       then (Call specialist with Establish)
         (if (+?  specialist)
           then (Call specialist with Refine)))))
```

Refine Procedure that Uses Suggestion Rules

Figure 7

Encoding of Symbolic Uncertainty

The other knowledge groups in the `kgs` section are used to implement uncertainty handling. Each of these CSRL knowledge groups is intended to correspond to an evidential abstraction underlying the 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, values of other knowledge groups) to some conclusion on a discrete, symbolic scale.

As an example, Figure 8 is the `relevant` knowledge group of the `BadFuel` specialist mentioned earlier. It determines whether the symptoms of the automobile are consistent with bad fuel problems. The expressions query the user (who is the database for Auto-Mech) about 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-valued logic. Each set of tests in the `with` part of the knowledge group is evaluated until one matches. The value corresponding to the rule that matches becomes the value of the knowledge group. For example, the first rule in the figure tests whether the first expression is true (the ? means doesn't matter). If so, then -3 becomes the value of the knowledge group. Otherwise, subsequent rules are evaluated.

The value of the knowledge group will be 1 if no rule matches. This knowledge group encodes the following matching 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.

```
(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)))
```

`relevant` Knowledge Group of `BadFuel`

Figure 8

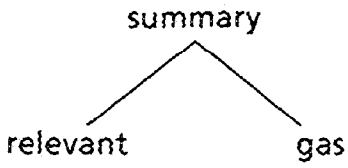
Figure 9 is the `summary` knowledge group of `BadFuel`. Its expressions are the values of the `relevant` and `gas` knowledge groups (the latter queries the user about the temporal relationship between the onset of the problem and the date when gas was last bought). In this case, if the value of the `relevant` 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.

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

`summary` Knowledge Group of BadFuel

Figure 9

This method of evidence combination allows the calculation of uncertainty to be hierarchically organized. In this instance, the hierarchy (illustrated in Figure 10) is very simple. For a more complex evaluation, additional knowledge groups, hierarchy layers, and pattern combinations can be defined as needed.



Hierarchy of Knowledge Groups
Figure 10

Implementation

The current version of CSRL is implemented in INTERLISP-D (Xerox, 1985) and LOOPS (Bobrow and Stefik, 1982; Stefik *et al.* 1983), 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. A graphic interface displays the specialist hierarchy and through the use of a mouse allows the user to easily access, modify, and run any part of the hierarchy. (We would like to note that this version of CSRL is an unsupported public domain program and is available for a nominal cost from Ohio State's AI Laboratory.)

Future versions of CSRL will be substantially different in order to integrate it into a more unified framework of a language specification for each generic task. Because we recently recognized that CSRL's uncertainty-handling capability corresponds to a separate generic task (called hypothesis matching), CSRL will be split into two languages, one for classification and the other for hypothesis matching. The way that classificatory specialists will determine their confidence is to invoke problem solvers which perform hypothesis matching. Languages for other generic tasks are currently being implemented as well as display, explanation, and debugging facilities that will be uniform across all of these languages. This work is being performed at Ohio State's AI Laboratory.

Expert Systems that Use CSRL

In this section, we briefly discuss a few expert systems that use CSRL.

Auto-Mech

Auto-Mech is an expert system that diagnoses fuel problems in automobile engines (Tanner and Bylander, 1985). This domain was chosen to demonstrate the viability of the MDX approach to nonmedical domains as well as to gain experience and feedback on CSRL.[4] 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 consists of 34 CSRL specialists in a hierarchy whose depth varies from four to six levels. 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 (for example, 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, because 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 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 were 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.

Red

Red is an expert system whose domain is red blood cell antibody identification (Smith *et al.* 1985). 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 can 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 database that 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 (Josephson *et al.* 1984). CSRL is used to implement specialists corresponding to each antibody that Red knows about (about 30 of the most common

[4]Auto-Mech was developed using an early version of the language.

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 the overview subsystem of the plausibility of each antibody. The specialists query the database for information about the test reactions and other patient information and also tell the database to perform certain operations on reaction records.

An interesting feature of Red is how it handles the problem of interacting hypotheses. It is possible for the patient's blood to have practically any number or combination of antibodies, making 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 might only partially account for the data; it doesn't reduce its confidence value if there is extra data to account for. 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 turns out that it is rare to have more than a few antibodies that cannot be independently ruled out. Thus, Red's CSRL subsystem makes the overview's problem solving more manageable because it considerably reduces the amount of search that would otherwise be necessary.

Real-World Use of CSRL

CSRL is being used to develop two commercial systems by the Knowledge-Based Systems group at the Battelle Columbus Institute. WELDEX (Mahalingam and Sharma, 1985) and ROMAD (Mahalingam *et al.* 1985) are diagnostic systems for detecting welding defects and evaluating machinery, respectively. A brief description of WELDEX follows.

WELDEX identifies possible defects in a weld from radiographic data on the weld. Industry standards and regulations require careful inspection of the entire weld and a very high level of quality control. Thus, for industries that rely on welding technology, such as the gas pipeline industry, radiographic inspection is a tedious, time-consuming, and expensive part of their operation.

This problem can be decomposed into two tasks: visual processing of the radiograph to extract relevant features of the weld and mapping these visual features to the welding defects that give rise to them. WELDEX is intended to perform the second task. The current prototype consists of 25 CSRL specialists that are organized around different regions of the weld, taking advantage of the fact that each class of defects tends to occur in a particular region. The knowledge groups in these specialists concentrate on optical contrast, shape, size, and location of the radiographic features. A customer version of WELDEX is currently being developed. Future work is anticipated on developing a visual-processing system whose output would be processed by WELDEX, thus automating both parts of the radiographic inspection problem.

Weaknesses of CSRL

CSRL lacks a good set of classification control primitives for refine message procedures, relying instead on general programming constructs. The result is that it is hard to encode refine procedures in CSRL that combine different priorities, such as invoking suggested specialists before those not suggested, invoking more common specialists before more rare specialists, and refining specialists with higher confidence values before those with lower confidence values. These procedures should also respond to priorities that the superspecialist requests. Additional work is needed to create a set of constructs that more closely match the kinds of operations that are needed for refinement and for cooperative action.

Another problem is the ambiguity of the -3 to +3 confidence scale.[4] For example, +3 was used to mean near certainty in MDX, highly relevant to consider in Auto-Mech, and highly plausible in Red. Future versions of CSRL will adopt more meaningful symbols for confidence values, such as `certain`, `likely`, `plausible`, `problematic`, and `ruled-out`.

A major missing piece in CSRL is that there is no strategy that determines when classification should stop. Currently, the default procedures simply ask the user if the current solution is satisfactory. From work on Auto-Mech and Red, it appears that this decision depends strongly on the goals of whoever uses the CSRL system. In Auto-Mech, the classification should stop when a repair is successful; in Red it should stop when a "best explanation" is reached. The work on Red's overview system is a step in this direction for diagnostic systems, but there needs to be more cooperation between the overview subsystem and CSRL (currently the overview subsystem 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; that is, the diagnosis could concentrate on accounting for the most important manifestation(s).

Issues of Classification

In this section, we further explain our positions on a few issues that influence the design of CSRL.

The Relationship of Classification to Diagnosis

The work on MDX demonstrated that classification is an important strategy in medical diagnosis. Because of this, we speculated that classificatory problem solving was the

[4]The use of numbers also created the false impression that these confidence values could be added and subtracted.

primary component of diagnosis in general, and consequently, what we now call the classificatory task had been called the diagnostic task. However, we have come to realize that diagnosis is a more complex phenomenon in which other problem solvers have major roles, and, depending on the domain, classification can play a minor role.

A useful companion to the classification hierarchy is a problem solver that performs the generic task of *knowledge-directed data retrieval* (Mittal *et al.* 1984). This problem solver can be thought of as an intelligent database that organizes the case description, answers queries from the classification specialists, and makes simple inferences from the data. For example, an intelligent database should be able to infer exposure to anesthetics if the patient has had major surgery or has been exposed to halothane (a type of anesthetic). The classificatory specialists are then relieved from knowing how one datum could be inferred based on its conceptual relationships to other data.

In some applications, such as diagnosing devices that are heavily instrumented with sensors (for example, nuclear power plants), there is almost a direct match between data and malfunctions. In these cases, the intelligent database performs the work of interpreting raw data, and the classificatory problem solving mainly consists of simply associating the processed data with hypotheses.

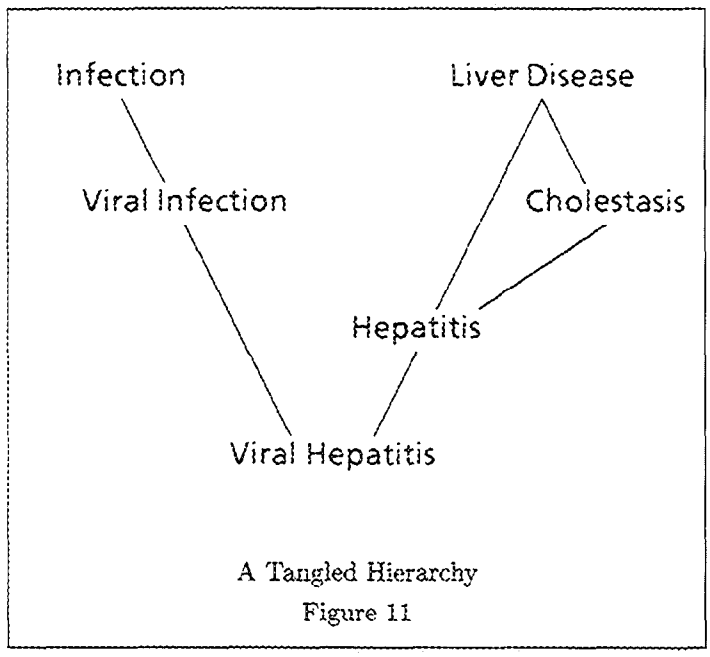
Another important part of diagnosis is accounting for abnormal findings and producing composite hypotheses when one malfunction cannot account for all of them. Although these capabilities have not been needed in some systems (such as MDX and MYCIN), they appear to be necessary in other systems and domains (for example, INTERNIST-I whose domain is internal medicine [Pople, 1977] and ABEL whose domain is acid-base disorders [Patil *et al.* 1981]). Josephson *et al.* (1984) have proposed a generic task called *hypothesis assembly* to perform these actions in coordination with a classificatory problem solver. Roughly, the classifier generates plausible hypotheses and determines the data they can account for. The hypothesis assembler builds and critiques composite hypotheses, taking into account interactions between hypotheses. The Red system discussed earlier implements this combination of problem solvers.

There are several other issues relevant to diagnostic problem solving that need to be considered, such as test ordering, causal explanation of findings, and therapeutic action. Fully resolving all of these issues and integrating their solutions into this diagnostic framework are problems for future research.

Tangled Hierarchies

CSRL, as previously noted, does not allow the representation of tangled hierarchies (hierarchies in which some nodes have more than one parent) and, consequently, is restricted to representing classification trees. Although tangled hierarchies are completely general, trees represent a simpler and cleaner solution for classification. In our experience, tangled hierarchies can often be untangled by a judicious mix of eliminating nodes that provide little classificatory power and carefully analyzing the domain.

For example, consider the tangled hierarchy in Figure 11. Although infection and viral infection are concepts that physicians use, they are not very useful for classificatory problem solving because there is little evidence that distinguishes infections and viral infections from other possibilities. Also, one might make hepatitis (liver inflammation) a subspecialist of cholestasis because hepatitis often causes cholestasis. However, hepatitis is not a subtype of cholestasis, and, furthermore, hepatitis often does not cause cholestasis; thus hepatitis should not be ruled out if cholestasis is ruled out. Figure 12 illustrates an untangled version of this hierarchy. The infection nodes are eliminated because of their weaknesses. A new node, "cholestasis caused by hepatitis," is added to distinguish this cause of cholestasis from other causes. This new node is one kind of composite hypothesis that can be embedded in a classificatory structure. The problem-solving issues of how this hypothesis can take advantage of decisions concerning the hepatitis hypothesis are discussed in Gomez and Chandrasekaran (1981).



A Tangled Hierarchy
Figure 11

Another advantage of trees over tangled hierarchies is that the knowledge of a specialist can be biased to the context of its superspecialist. Because of the establish-refine strategy, the specialist's knowledge can assume that the superspecialist has been established. This simplifies the amount of knowledge that needs to be encoded because decisions can be made in a single context, eliminating the caveats needed to encode other contexts. Also, the specialist can take advantage of knowledge that distinguishes

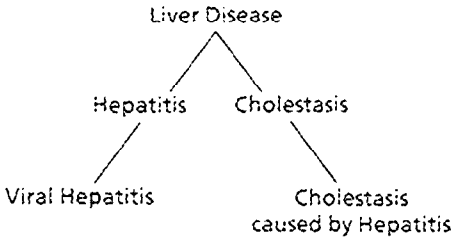


Figure 11 Untangled

Figure 12

the specialist from its siblings. However, there is a danger involved—if the superspecialist produces overconfident decisions, the biasing of knowledge can cause the specialist to also be overconfident.

This should not be taken as an argument that tangled hierarchies are never necessary but that they can often be simplified without loss of problem-solving power. There is a need to discover and investigate those situations in which the complexities of tangled hierarchies pay off in increased problem-solving ability.

Uncertainty in Classification

Uncertainty handling in CSRL is an outgrowth of the technique used in MDX (Chandrasekaran *et al.* 1982). This technique is based on a number of assumptions:

Uncertainty is not a unitary phenomenon. Much of the debate within AI treats uncertainty as if it were a single phenomenon. However, a clear distinction needs to be made between uncertainty as degree of chance versus uncertainty as degree of fit. Rather than assessing the probabilities of hypotheses, the method of uncertainty handling incorporated within CSRL measures the qualitative degree of fit between hypotheses and data. It is interesting to note that this viewpoint has recently been supported by Cohen *et al.* (1985), who cogently argue that degree of fit (called representativeness in their paper) is an appropriate measure of uncertainty in classification.

Uncertainty introduced by the problem solver should be kept to a minimum. In AI uncertainty-handling techniques can introduce uncertainty by making assumptions that are not true of the domain and by requiring knowledge that cannot be obtained, forcing implementors to make guesses. In the case of expert systems, it is important that the expert's reasoning is accurately modeled.

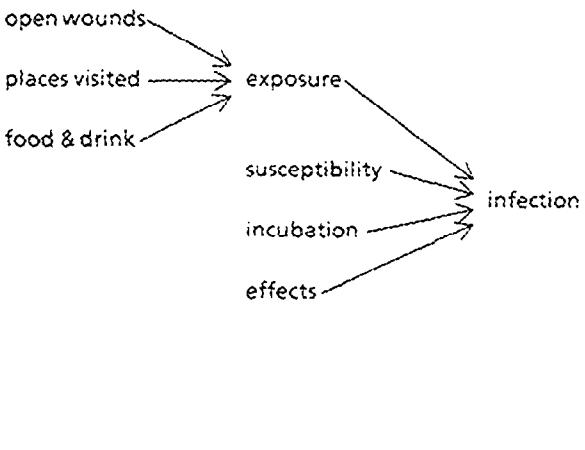
Symbolic uncertainties should be used for expert system reasoning. This and the previous paragraph seem to be contradictory because symbolic uncertainties appear to introduce uncertainty from the start. However, we need to remember that these judgments are coming from people, who simply don't have specific fractions in their heads. Typically, knowledge engineers translate the expert's symbolic judgments into some numeric uncertainty calculus, assume that the methods of inference combination will correspond to the expert's reasoning, and then translate the numbers back into the symbolic scale. Each of these steps introduces uncertainty because they move from the expert's reasoning methods to an artificial system that the expert doesn't use. It would make more sense to use the expert's judgment not only for the "single" inferences, but for the combination of inferences as well.

Our methodology, which we call *hypothesis matching*, allows for a hierarchical evidence structure in which the expert provides symbolic judgments on combinations of data as well as on how these judgments should be combined to determine the confidence in the hypothesis. The following scenario illustrates how the hierarchical organization can be naturally formed from the data.

Suppose that we want to measure the evidence for a particular bacterial infection called X. It is likely that a large amount of data is potentially useful as evidence for or against X. It would be a combinatorial nightmare to match all combinations of data to a judgment on X's presence. However, these data are not evidentially unrelated to one another, but subsets of them are evidence about particular parts of the infection process. One thing we probably want to know is whether the patient has been exposed to the bacteria that causes X. Usually, exposure to bacteria is not a datum given to the system; instead, there are several data that give indications about exposure. For example, open wounds, places the patient has been, what the patient has been eating or drinking, and other activities of the patient are all factors that need to be taken into account. Now we can ask the expert how these data are evidence for exposure to X bacteria. The left half of Figure 13 illustrates this part of the evidence combination. For other data, we ask how they relate to susceptibility, incubation, various effects of the infection, and so forth. After a judgment is made on each of these features of X, we can then ask how combinations of these abstractions (at various levels of uncertainty) relate to X itself. This is illustrated in the right half of Figure 13.

As this scenario indicates, the methodology of hypothesis matching is very simple—data are combined into judgments on evidential abstractions underlying the hypothesis, and these judgments are combined, using as many levels as needed, into a judgment on the hypothesis itself. Here are some other points to keep in mind:

- Because any number of levels of abstractions is permitted, a large body of evidence can be decomposed and combined in a combinatorially manageable fashion.
- These abstractions are not arbitrarily chosen but come from the expert's understanding of the hypothesis.
- Because the expert is providing symbolic judgments



A —> B = A is evidence to consider when evaluating B

Combination of Evidence for Infection

Figure 13

at each level of abstraction, there is no need for an uncertainty calculus, and there is no reason to translate into and out of a continuous probability system.

It turns out that hypothesis matching is useful for more than classification. Both MDX's database (Mittal, 1980; Mittal *et al.* 1984) and AIR-CYL (Brown and Chandrasekaran, 1984), a design expert system, used forms of hypothesis matching to map from situations to decisions. Based on these examples and some further analysis, hypothesis matching is now considered to be a generic task.

This method of evidence combination closely matches the signature table strategy used by Samuel's checker-playing program (Samuel, 1967). Just as Samuel used layers of signature tables to reduce the complexity of decision making in his program, the layers of evidential abstraction eliminate the need to provide one rule for each combination of data and also make the decision making comprehensible to both the knowledge engineer and the domain expert (Chandrasekaran, 1985).

Hypothesis matching allows a simple but powerful debugging technique (see Sticklen *et al.* [1985b] for more explanation and for another application of hypothesis matching). If the expert system determines the wrong level of uncertainty for a hypothesis, the bug can be traced to the abstractions that produced incorrect answers, which can then be appropriately modified.[5] After many debugging sessions, the system becomes tuned to the domain within the conceptual framework of the domain expert.

[5]This can be as simple as adding another "rule" to the abstraction or as difficult as completely reorganizing the abstractions

To make our position clear, we do not mean to advocate hypothesis matching for every classification system. Bayesian probabilities, for instance, should be used if quantitative data on probabilities are available and if the assumptions apply. However, when the idealized conditions of uncertainty calculi are not applicable, which they often aren't for expert systems, they become knowledge engineering obstacles instead of knowledge engineering tools. It makes much more sense to use the expert's judgments and conceptual structures whenever possible and to test and refine them by using real cases.

Comparison to Heuristic Classification

Clancey (1984, 1985) has proposed a description of classification, called "heuristic classification," that is "an attempt to specify what many heuristic programs known as 'expert systems' do." Heuristic classification has three parts:

Data Abstraction: This relation maps observations into data classes or characterizations based on categorical inferences. Clancey gives the examples of inferring low white blood count from a specific white blood count and immunosuppression from the low white blood count (low white blood count is a form of immunosuppression).

Heuristic Match: This relation maps data or data abstractions to hypotheses in the classification hierarchy. This mapping is based on heuristic, but not certain, knowledge.

Refinement: This relation indicates what hypotheses are subhypotheses of other hypotheses.

The processes of heuristic classification then use these relations to find a solution in the hierarchy.

We agree with Clancey that heuristic classification characterizes the problem solving of a considerable number of AI programs and that it is a useful strategy to solve many problems. However, from our perspective, heuristic classification corresponds to a combination of generic tasks rather than being a primitive strategy. This point has practical consequences. Instead of developing a single "knowledge engineering tool designed specifically to perform heuristic classification," our approach leads us to develop separate tools for each generic task and to provide mechanisms that not only allow heuristic classification to be constructed out of several problem solvers but also allow those generic tasks to be used in other ways.

Clancey's data abstraction relation, for example, corresponds to the capability of the intelligent database described previously. This capability, however, is not just useful for classification but for other problem-solving methods as well. We can imagine a designer, that is, an agent who constructs a design plan from specifications, needing to know if some material is metallic or if the temperature of the operating environment will be below freezing. Because simple data inference is a useful function for many tasks, and because it requires knowledge organization and problem-solving strategies different from classifi-

cation, it is natural to think of it as a separate reasoning strategy.

This separation is more clearly seen in programs that do classification but not data abstraction. For example, the DART diagnostic program (Genesereth, 1984) uses classification to find circuit faults, but instead of using data abstraction it generates tests to confirm or reject faults. Although DART's problem solving does not fit into the heuristic classification mold, it can be analyzed as a combination of classification problem solving and test generation based on functional knowledge.

The heuristic match relation in heuristic classification is intended to account for two types of behavior: suggesting hypotheses and confirming or rejecting hypotheses. Our approach accounts for the first type by embedding suggestion rules in individual specialists. This has the advantage of tailoring suggestion knowledge to the classificatory context. Also, the suggestion rules can operate on patterns of data rather than on just a single datum or data abstraction. The apparent disadvantage of efficiency by not executing the rules when the data are entered can be allayed by implementing the suggestion rules as demons. Confirming and rejecting hypotheses within CSRL is done by hypothesis matching, a generic task that we have already covered in some detail.

Finally, we wish to emphasize that classification is useful for more than selecting a single solution from a hierarchy. We have already mentioned one architecture in which classification and hypothesis assembly can be integrated to produce composite hypotheses. Even without hypothesis assembly, some forms of hypotheses interaction can be reasoned about by a classification problem solver (Sticklen *et al.* 1985a).

Conclusion

How does CSRL meet the needs that we listed in the introduction? Currently, CSRL makes some small steps toward constraining the meaning of symbols. Because ruling out a specialist results in ruling out its subspecialists, CSRL's classification tree is more than a simple way to associate hypotheses with one another. The children of a hypothesis must be subhypotheses. The current CSRL implementation does not enforce the meaning of its messages, but given a better analysis of the operations that are needed for each type of message, a message procedure could be appropriately restricted.

CSRL makes the organization of classificatory problem solvers explicit by providing appropriate abstractions for classificatory knowledge: specialists, message procedures, knowledge groups, and rules. The expert system implementor is then relieved from the burden of implementing one level of organization in a language at a different level and is free to concentrate on the conceptual structure of the domain. Also, there is a greater potential to embed general explanation facilities that can take advantage of the organization.

We believe that CSRL provides a good match to a sizable portion of human expertise. Whenever the solution set of a problem (or subproblem) can be characterized by a classification tree and whenever an expert can provide the judgments that map data to confidence in classificatory hypotheses and the evidential abstractions that underlie them, the classification tree and the judgments can be directly represented in CSRL.

From the perspective of the generic task framework, CSRL handles only two (classification and hypothesis matching) of the many types of problem solving that experts use. Besides improving CSRL, special-purpose languages need to be developed for the other generic tasks that have been identified, and these languages need to be integrated for cooperative problem solving. Currently, languages for knowledge-directed data retrieval (IDABLE [Intelligent DAta Base LanguagE]) and another generic task called "plan selection and refinement" (PSRL [Plan Selection and Refinement Language]) (Brown and Chandrasekaran, 1984) have been implemented. In addition, a language for hypothesis assembly is also being developed. This family of languages should be a powerful tool because it will allow implementors to encode different types of problem-solving knowledge, such as that used for diagnosis, in the appropriate form and to integrate the resulting problem solvers into complex knowledge-based systems. For example, note that CSRL together with IDABLE will provide the function of Clancey's heuristic classification while making IDABLE's function available for other situations where a data abstraction capability is called for.

References

Bobrow, D., & M. Stefik. 1982. *The LOOPS Manual.* Tech. Rep KB-VLSI-81-13, Palo Alto, California: Xerox Palo Alto Research Center

Brachman, R J., & J. G. Schmolze 1985 "An Overview of the KL-ONE Knowledge Representation System" *Cognitive Science* 9(2): 171-216

Brown, D. C , & B Chandrasekaran 1984. "Expert Systems for a Class of Mechanical Design Activity" In *Proceedings of the IFIP WG5 2 Working Conference on Knowledge Engineering in Computer Aided Design, Budapest, Hungary* New York: IEEE Computer Society

Bylander, T., & J W. Smith. 1985. "Mapping Medical Knowledge into Conceptual Structures" In *Proceedings of Expert System in Government Symposium, McLean, Virginia.* New York: IEEE Computer Society, 503-511

Chandrasekaran, B 1983 "Towards a Taxonomy of Problem Solving Types" *AI Magazine* 4(1): 9-17

Chandrasekaran, B 1985. "Generic Tasks in Knowledge-Based Reasoning: Characterizing and Designing Expert Systems at the 'Right' Level of Abstraction" In *Proceedings of the Second Conference on Artificial Intelligence Applications, Miami Beach, Florida.* New York: IEEE Computer Society, 294-300.

Chandrasekaran, B. 1986 "From Numbers to Symbols to Knowledge Structures: Pattern Recognition and Artificial Intelligence Perspectives on the Classification Task." In *Pattern Recognition in Practice-II.* Amsterdam: North-Holland

Chandrasekaran, B., F Gomez, S Mittal, & J. W. Smith 1979 "An Approach to Medical Diagnosis Based on Conceptual Structures." In *IJCAI-6.* Los Altos, California: William Kaufmann, Inc , 134-142.

Chandrasekaran, B., & S Mittal 1983 "Conceptual Representation of Medical Knowledge for Diagnosis by Computer: MDX and Related Systems " In M Yovits ed , *Advances in Computers.* New York: Academic Press, 217-293.

Chandrasekaran, B , S. Mittal, & J W. Smith 1982 "Reasoning with Uncertain Knowledge: The MDX Approach." In *Proceedings of the Congress of American Medical Informatics Association, AMIA, San Francisco,* 335-339.

Clancey, W J 1981. "NEOMYCIN: Reconfiguring a Rule-Based Expert System for Application to Teaching " In *IJCAI-7* Los Altos, California: William Kaufmann, Inc.,, 829-836

Clancey, W J 1984 "Classification Problem Solving." In *AAAI-84.* Menlo Park, California: American Association for Artificial Intelligence, 49-55.

Clancey, W J 1985 "Heuristic Classification." *Artificial Intelligence* 27(3): 289-350

Cohen, P., A. Davis, D. Day, M. Greenburg, R Kjeldsen, S Lander, & C. Loiselle 1986 "Representativeness and Uncertainty in Classification Systems." *AI Magazine* 6(3): 136-149.

Forgy, C. L. 1981. *OPS5 Users Manual.* Tech Rep CMU-CS-81-135. Pittsburgh, Pennsylvania: Carnegie-Mellon University.

Genesereth, M R 1984 "The Use of Design Descriptions in Automated Diagnosis." *Artificial Intelligence* 24: 411-436

Gomez, F., & B. Chandrasekaran. 1981 "Knowledge Organization and Distribution for Medical Diagnosis " In *IEEE Trans. Systems, Man and Cybernetics* SMC-11(1): 34-42

Gordon, J., & E. H. Shortliffe. 1985 "A Method for Managing Evidential Reasoning in a Hierarchical Hypothesis Space " *Artificial Intelligence* 26(3): 323-357.

Hayes-Roth, B. 1985 "A Blackboard Architecture for Control " *Artificial Intelligence* 26(3): 251-321

Josephson, J R , B Chandrasekaran, & J. W. Smith. 1984. "Assembling the Best Explanation " In *Proceedings of the IEEE Workshop on Principles of Knowledge-Based Systems, Denver, Colorado.* New York: IEEE Computer Society, 185-190

Mahalingam, S , & D D. Sharma 1985 "WELDEX—An Expert System for Nondestructive Testing of Welds " In *Proceedings of the Second Conference on Artificial Intelligence Applications, Miami Beach, Florida.* New York: IEEE Computer Society, 572-576

Mahalingam, S., D D Sharma, B. Ellis, & B Morrow 1985 "ROMAD: An Expert System for Vibration Analysis " In *Proceedings of the Control West Conference, Long Beach*

McDermott, J 1982 "R1: A Rule-based Configurer of Computer Systems " *Artificial Intelligence* 19(1): 39-88.

Mittal, S , B Chandrasekaran, & J. Sticklen. 1984. "PATREC: A Knowledge-Directed Database for a Diagnostic Expert System " *Computer* 17(9): 51-58

Mittal, S 1980 "Design of a Distributed Medical Diagnosis and Database System." Ph D. diss , Department of Computer and Information Science, The Ohio State University

Patil, R. S., P. Szolovits, & W. B Schwartz 1981 "Causal Understanding of Patient Illness in Medical Diagnosis " In *IJCAI-7* Los Altos, California: William Kaufmann, Inc., 893-899

Pople, H 1977. "The Formation of Composite Hypotheses in Diagnostic Problem Solving: An Exercise in Synthetic Reasoning " In *IJCAI-5.* Los Altos, California, William Kaufmann, Inc , 1030-1037.

Samuel, A. 1967. "Some Studies in Machine Learning Using the Game of Checkers " *IBM Journal of Research and Development* 11(6): 601-617.

Smith, J. W., J R. Svirbely, C. A Evans, P Straum, J R Josephson, & M C Tanner 1985 "Red: A Red-Cell Antibody Identification Expert Module " *Journal of Medical Systems* 9(3): 121-138

Stefik, M , D G Bobrow, S Mittal, & L Conway 1983 "Knowledge Programming in LOOPS." *AI Magazine* 4(3): 3-13

Sticklen, J , B Chandrasekaran, & J. Josephson 1985 "Control Issues in Classificatory Diagnosis " In *IJCAI-9.* Los Altos, California: William Kaufmann, Inc , 300-306

Sticklen, J , B Chandrasekaran, & J W Smith. 1985. "MDX-MYCIN: The MDX Paradigm Applied to the MYCIN Domain." *Computers and Mathematics with Applications* 11(5): 527-539.

Tanner, M. C., & T. Bylander 1985 "Application of the CSRL Language to the Design of Expert Diagnosis Systems: The Auto-Mech Experience." In J. Richardson ed , *Artificial Intelligence in Maintenance,* Park Ridge, New Jersey: Noyes, 149-169.

Weiss, S M , & C A Kulikowski. 1984. *A Practical Guide to Designing Expert Systems* Totowa, New Jersey: Rowman and Allanheld

Xerox 1983 *Interlisp Reference Manual* Pasadena, California: Xerox Artificial Intelligence Group.